Fractal Dimensions and their Relationship to Filtration Characteristics

by

Simon T. H. Brock

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

September 2000

ABSTRACT

Work has been performed to characterise filtration systems according to their fractal properties and to construct agglomerates to mimic the filtration systems under scrutiny.

The first objective was achieved by carrying out experiments examining the dead-end filtration of two separate mineral suspensions, namely calcite and talc. These minerals were chosen to represent typical incompressible (calcite) and compressible (talc) filtration systems, undergoing filtration using a range of pressures. The experimental apparatus produced filter cakes that could be sampled, sectioned and examined under high magnification.

The second objective was met by developing a computer application that could construct simulated particle agglomerates in both two and three dimensions, using a seed agglomeration model as well as simulating filtration by means of a virtual filter cell. A large number of simulations were completed to mimic both the dead-end filtration and other agglomerate models. The computer application was also capable of characterising the fractal and Euclidean spatial nature of both the simulated and experimental particulate systems, using a variety of techniques.

Euclidean spatial attributes such as porosity as well as fractal properties including surface roughness and a number of density fractal dimensions have been measured for both types of system and demonstrate that the conditions under which the trials were performed have a bearing on the final configuration of the structures. Results from both experimental and simulation work show that fractal dimensions offer a valid method of measuring the properties of filtration systems.

Experimental results showed that as the filtering pressure was increased, the density fractal dimension for the system appeared to increase. This change in fractal dimension was also accompanied by a decrease in the porosity of the system (more so for talc than the calcite), confirming the compressibility of the materials under scrutiny. The characterisation of the sampled cakes also showed that the spatial characteristics vary within the individual slices of the sample, in agreement with modern filtration theory.

Results from the simulations show that both the physical and fractal properties of the resulting structures varied with the parameters used to construct them. As a rule, as the particles in the simulations were able to move in a more diffusive manner (akin to Brownian motion), the agglomerates they formed had a more open, rugged structure. The simulation of filtration systems also showed a variation within the individual cake structures. In the case of the filtration simulations, the probability assigned to the particles' sticking to the growing agglomerate was the controlling factor. In addition, it was found that the simulated cakes had similar spatial properties to the experimental systems they were designed to replicate.

ACKNOWLEDGEMENTS

Firstly, I would like to thank *all* my family and friends for their support (and nagging) during the arduous process that has been the creation of this work.

Special mention must go to a number of people who have been more closely involved than others. Without the constant support of Linda, Terry, Lucy and Bob, this work would undoubtedly never have reached completion – my apologies for any sleepless nights I may have caused.

Many thanks also to my supervisor, Dr. E.S. Tarleton, who unwittingly decided my fate and his, back in 1995. Thank you for leading me along such an interesting path, Steve.

My gratitude must also be expressed to past colleagues at the university, with particular thanks to Warren Eagles, Dr. Bruce Jefferson, Jonathan Pascoe, Dr. David Richardson and Mel Smythe – making three years enjoyable as well as educational. One colleague without whom finger would not have touched keyboard is Dr. Daniel Horner, who has directed me in more ways than one along the often rocky path towards the completion of this work. To you all - may the Zulu spirit live on.

Finally, a tribute to my late father, whose unspoken guidance has undoubtedly kept me going through the ups and downs of the last five years.

TABLE OF CONTENTS

1 INTRODUCTION AND LITERATURE SURVEY

This chapter identifies the basics of filtration and fractal mathematics and shows examples of the use of fractal measures in practical applications. The aim of the chapter is to provide an introduction to the main areas of study contained in this work.

1.1 Filtration fundamentals

Liquid filtration is defined [Coulson, 1991] as "*The separation of solids from a suspension in a liquid by means of a porous medium or screen which retains the solids and allows the liquid to pass*".

The suspension of solids is termed a slurry, which is moved toward the medium by a positive or negative (vacuum) pressure or sometimes gravity. There are two fundamental classes of filtration - 'surface' or 'cake' filtration and 'depth' filtration. With surface filtration the suspension is trapped on the surface of the filtering medium and a cake is formed. As the cake grows, the retained solids filter the remaining suspension, leaving the filter medium to act as a support. Depth filtration also involves a medium, but in this instance, the solids being separated are held within the medium rather than on the surface. Surface filtration is usually used for higher concentrations of solids (> 1% v/v) whilst depth filtration is used for lower concentrations [Svarovsky, 1981]. Surface, or cake, filtration has wider applications in the chemical industry than depth filtration and is the mechanism investigated in the work presented.

In cake filtration, as a build up of solids occurs, the suspension is filtered not only through the filter medium, but also through the cake itself. The liquid to be separated flows through interstices formed by the arrangement of particles within the cake. If left to continue in this manner, the filter vessel will eventually fill with cake and will not be able to operate efficiently. Some type of cleaning or washing of the vessel is therefore required as part of the overall filter cycle, particularly with batch-cycle filters.

A typical example of cake filtration is shown in Figure 1.1, with the slurry being introduced into a filter cell from the top and a filter cake forming on the lower surface. This cake is supported by a filter medium, which is in turn held on a rigid support.

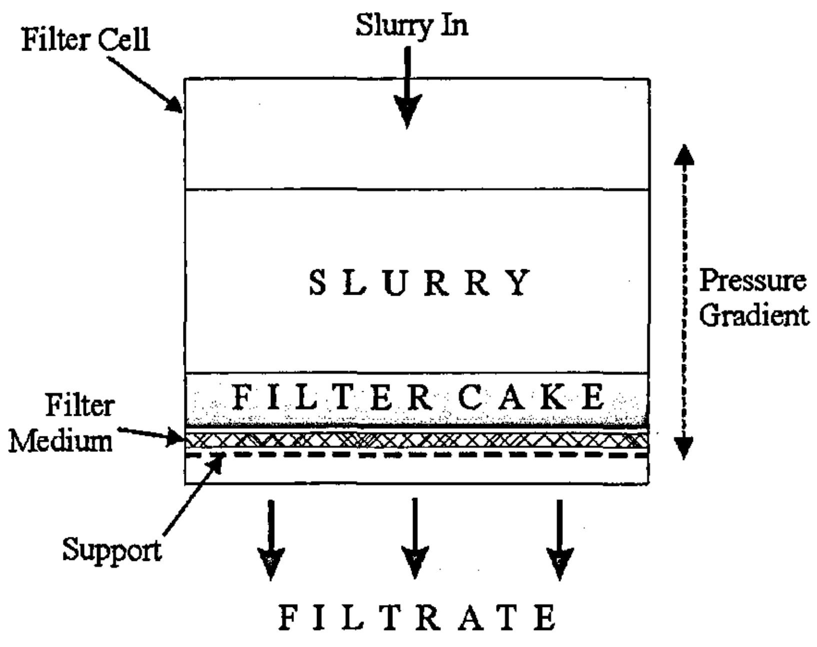


Figure 1.1 - A schematic diagram of dead-end cake filtration

In cake filtration, two types of filter cake can be formed - *incompressible* and *compressible*. In the former case, an essentially homogeneous cake is formed as the filtration progresses, with the cake having a near constant porosity at all points within the cake. The homogeneity in this situation generally means that the cake is easier to study and categorise than a compressible cake. In the latter, as the pressure gradient increases across the cake (i.e. as more cake is built up) the cake tends to squash and become denser as the particles move against one another. This also means that the cake will not be homogeneous throughout, generally being denser at the bottom of the cake where the pressure from above is greatest.

The characteristics of a filter cake are usually described in terms of *cake resistance* and *porosity*. Average specific cake resistance ($\overline{\alpha}$) describes the resistance to flow that the cake offers against the slurry. The average porosity ($\overline{\varepsilon}$) is the amount of space contained within the cake per unit volume and is dimensionless (actually $m^3_{liquid}\ m^{-3}_{total}$). The two can be related thus:

$$\overline{\alpha} = \frac{1}{\rho_s(1-\overline{\varepsilon})k} \tag{1.1}$$

where ρ_s is the density of the solid phase and k the permeability. Other properties that need to be considered for filtration theory are the applied pressure (Δp), the area of the filtering surface (A), the viscosity of the filtrate (μ) and the resistance of the filter medium itself (R_m).

Originally, Darcy's Law [Svarovsky, 1981] stated that the flow of a liquid through a porous medium could be described in terms of the medium dimensions, permeability and pressure thus:

$$Q = -k\frac{\Delta p A}{\mu L} \tag{1.2}$$

where L is the depth of the medium and μ filtrate viscosity, not originally part of the equation.

Thus, the flow through a filter medium can be given by:

$$Q = \frac{A\Delta p}{\mu R_m} \tag{1.3}$$

where R_m is the resistance of the filter medium given by:

$$R_m = \frac{L}{k} \tag{1.4}$$

A modified form of Darcy's equation can be used to describe the flowrate (Q) through the cake and filter medium combined:

$$Q = \frac{A\Delta p}{\mu(R_m + R_c)} \tag{1.5}$$

where R_c is the resistance of the filter cake given by:

$$R_c = \overline{\alpha}\, w \tag{1.6}$$

where w is the mass of cake per unit area.

1.1.1 Incompressible cake filtration

For incompressible cake filtration, substituting Eq. (1.6) into Eq. (1.5) gives:

$$Q = \frac{A\Delta p}{\mu R_m + \mu \alpha w} \tag{1.7}$$

where α and $\bar{\alpha}$ are identical, as the resistance of the filter cake does not change throughout its depth.

If the mass of cake deposited ($w\,A$) is a function of the volume of filtrate passed through the system (V) and the effective feed concentration (c), further substitution into (1.7) can be made. If it is assumed that the amount of solids exiting with the filtrate is negligible, then:

$$w\,A = c\,V \tag{1.8}$$

where c is given by:

$$c = \frac{\rho_s s}{1 - ms} \tag{1.9}$$

where s is the solids weight fraction in the feed slurry and m is the "moisture ratio" given by Eq. (1.10) :

$$m = 1 + \frac{\bar{\varepsilon}\rho_l}{(1 - \bar{\varepsilon}\rho_s)} \tag{1.10}$$

. Substituting Eq. (1.8) into Eq. (1.7) gives:

$$Q = \frac{A\Delta p}{\mu R_m + \mu \alpha\, c(V/A)} \tag{1.11}$$

or

$$\frac{1}{Q} = \frac{dt}{dV} = \frac{\mu R_m}{A\Delta p} + \frac{\mu \alpha c V}{A^2 \Delta p} \tag{1.12}$$

where t is the elapsed filtration time. Assuming Δp is constant, Eq. (1.12) can be integrated to give:

$$t = \frac{\mu\alpha c}{A^2\Delta p}.\frac{V^2}{2} + \frac{\mu R_m}{A\Delta p}.V \qquad (1.13)$$

rearranged, this gives:

$$\frac{t}{V} = \frac{\mu\alpha c}{2A^2\Delta p}V + \frac{\mu R_m}{A\Delta p} \qquad (1.14)$$

For *constant pressure filtration*, if μ, c and A also remain constant then it is possible to calculate α and R_m from experimental data by plotting $\frac{t}{V}$ vs. V, giving a slope of $\frac{\mu c\alpha}{2A^2\Delta p}$ (enabling the calculation of α) and an intercept of $\frac{\mu R}{A\Delta p}$, thereby being able to calculate R_m (Figure 1.2).

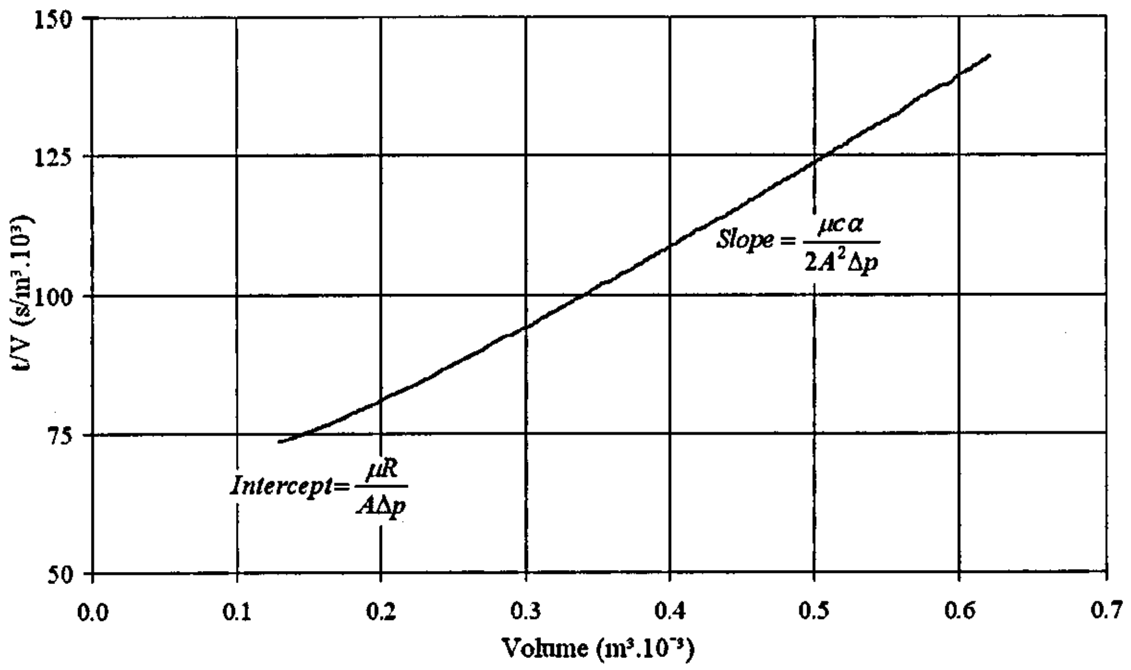


Figure 1.2 - Plot showing typical t/V vs V from experimental data (Calcite)

Similarly, for constant *rate* filtration, whereby the filtrate flowrate (*dV/dt*) is maintained at a constant level, Eq. (1.12) can be rearranged to give Eq.(1.15):

$$\Delta p = \left(\frac{\mu c \alpha}{A^2 t}\right) V + \left(\frac{\mu R_m}{A}\right)\frac{V}{t} \tag{1.15}$$

Eq. (1.15) can be rearranged using the relationship:

$$V = Qt \tag{1.16}$$

so that the pressure required to maintain a constant filtrate flowrate is expressed as:

$$\Delta p = \left(\frac{\mu c \alpha}{A^2 t}\right)\frac{Q}{t} + \left(\frac{\mu R_m}{A}\right) Q \tag{1.17}$$

1.1.2 Compressible cake filtration

If a cake is compressible, i.e. the structure changes with the conditions under which it is formed, for any given pressure, the specific cake resistance will vary throughout the cake depth, with the profile of this change varying both as the filtration progresses and with variation in pressure.

One method for describing the average resistance of a compressible filter cake accounts for the changes in porosity and therefore resistance throughout the height of the cake and is given by Eq. (1.18):

$$\frac{1}{\bar{\alpha}} = \frac{1}{\Delta p_c}\int_0^{\Delta p_c} \frac{P_s}{\alpha} \tag{1.18}$$

However, the value of $\bar{\alpha}$ does not correspond to the true definition of the average cake resistance, rather a function of the applied pressure.

The most commonly used cake resistance for compressible filter cakes is given by Eq. (1.19) [Almy and Lewis, 1912] where α_0 is the cake resistance at unit pressure and n is a compressibility index (zero for incompressible systems). If a series of experiments are performed at different yet constant pressures, a graph of $ln\ \alpha$ vs. $ln\ \Delta p$ can be plotted to determine both α_0 (intercept) and n (slope) - Figure 1.3.

$$\bar{\alpha} = \alpha_0 (\Delta p_c)^n \tag{1.19}$$

where Δp_c is the pressure drop across the cake in question, usually taken to be Δp, the filtration pressure.

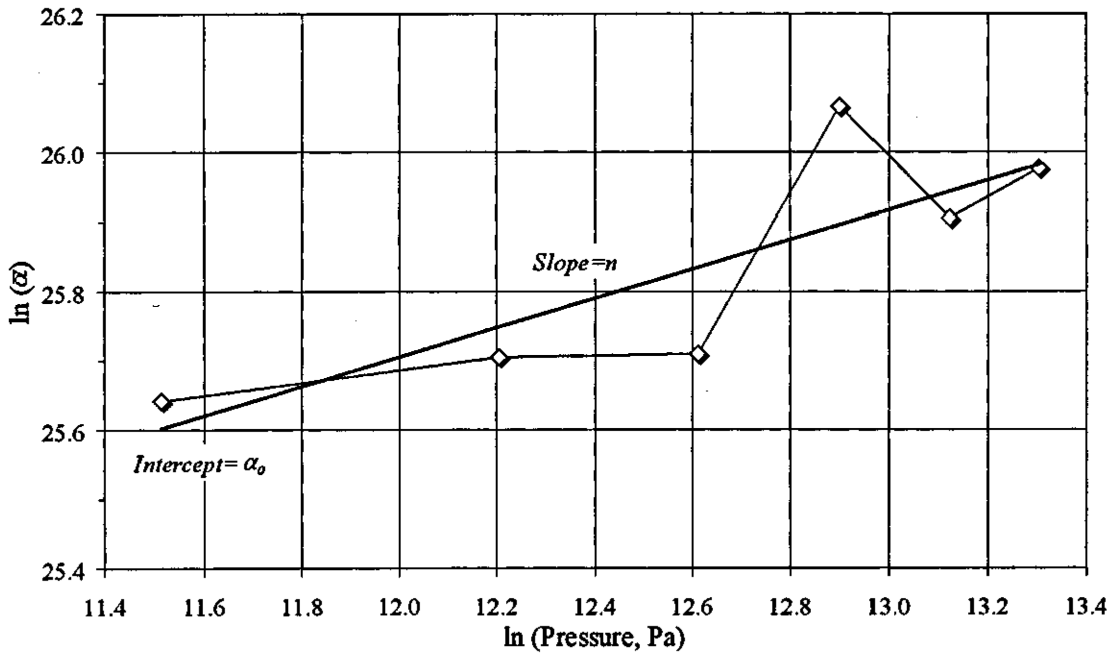


Figure 1.3 - Plot showing typical ln α vs. ln Δp from experimental data (Talc)

Eq. (1.19) is, however, an empirical equation usually valid for a limited range of pressures. The *average* specific cake resistance can better be given as:

$$\bar{\alpha} = (1-n)\alpha_0(\Delta p)^n \tag{1.20}$$

Again, a different mechanism is used for constant rate filtration of compressible cakes, where the filtrate flow-rate is held at a desired level by increasing the pressure driving force for the system. In this case, Eq. (1.11) becomes:

$$Q = \frac{A\Delta p}{\mu\bar{\alpha}\,c(V/A) + \mu R_m} \tag{1.21}$$

with the pressure drop varying with time. This means that the pressure can be related to the flow-rate required in the same way as Eq. (1.17):

$$\Delta p = \left(\frac{\bar{\alpha}\mu c}{A^2}\right)Q^2 t + \left(\frac{\mu R_m}{A}\right)Q \tag{1.22}$$

A plot similar to Figure 1.3 can be used to determine the factors in equation (1.22) giving a slope of $\bar{\alpha}\mu c\frac{Q^2}{A^2}$ and an intercept of $\mu R_m\frac{Q}{A}$.

1.1.3 Advances in filtration theory

Whilst the theory outlined in sections 1.1.1 and 1.1.2 is suitable for obtaining information concerning incompressible filter cakes and *average* values for the properties of compressible cakes, work has previously been performed to characterise some of the dynamic mechanisms of cake filtration. Using, for instance, partial differential equations to give variations in cake properties as functions of time and distance, models have been proposed to predict the cake properties such as porosity, specific resistance, permeability etc. and variations throughout the cake depth.

Tiller [1953] examined a number of physical properties of filter cakes, including the pressure drop with cake depth, filtrate flux with pressure drop, pressure variation with time (constant rate filtration) and the volume / time relationship (constant pressure filtration). Tiller used the assumption that the porosity varied only as a function of the pressure on the solids, given as:

$$P_S = P/(1-\varepsilon) \tag{1.23}$$

where P_s is the pressure on the solids and P is the overall pressure applied.

As a starting point in a number of the studies, Darcy's Eq. (1.2) can be written in the form of a differential equation (1.24) [Wakeman, 1978]:

$$v_{Lr} = -\frac{k_x}{\mu}\frac{\partial P_L}{\partial x} \tag{1.24}$$

where v_{Lr} is the liquid volumetric flux density relative to the solids in the system, and x the distance measured away from the filter medium. Eq. (1.24) gives the liquid flux as a function of the applied pressure throughout the cake and provides the basis for much of 'modern' filtration theory attributed to Tiller and Shirato [Tiller, 1953; Shirato & Sambuichi, 1985].

Shirato *et al* [1969] considered the motion of solids and liquids within the cake to describe the changes in cake characteristics for the filtration of concentrated slurries. A correction factor, J_R, for the specific cake resistance was proposed, where J_R is given by:

$$J_R = \int_0^1 \left[1 - \frac{(\varepsilon_x - \bar{\varepsilon}_x)(m-1)}{(1-\varepsilon_x)\bar{\varepsilon}(1-ms)} s\left(\frac{x}{L}\right) \right] d\left(\frac{w_x}{w}\right) \tag{1.25}$$

and the corrected specific cake resistance,

$$\alpha = J_R \alpha_R \tag{1.26}$$

where α_R is the *average* specific filtration resistance as defined by Ruth [1946].

Koenders and Wakeman [1997] developed a model based on micromechanics that relied on the membrane permeability, effective drag coefficient of the particles (both as a function of the solidosity of the depositing particles) and the stiffness of the solids matrix. However, the model can only be applied to the filtration of so-called "structured" materials.

Landman *et al* [1995] looked at the problem from a rheological viewpoint, considering the hydrodynamic forces exerted on the system during filtration and gave the specific resistance of the cake in terms of a hydrodynamic resistance parameter, $r(\phi)$, the inverse of permeability.

Stamatakis and Tien [1991] suggested a rigorous approach to predicting the dynamics of the build up of a cake on a filter surface, namely the varying solidosity, filtration rate and permeability.

It is seen from these works that modern filtration theory requires the determination of as many empirical parameters as the classical theory and the latter is thus generally used in preference. The modern filtration theory is also often presented in differential equation form (for example Eq. (1.25)), which whilst being able to predict a solids concentration profile within a forming cake, also makes the determination of cake structures more complicated.

The work outlined in this report does not concern itself with the modern developments in filtration theory, but instead will concentrate on the more widely used traditional filtration equations.

1.2 Simulation of particle agglomerates

Due to the transient nature of filtration, it is difficult to define the exact mechanisms that are occurring at any one time. Whilst taking an image of the forming filter cake for a large range of time scales can be achieved through tomography [Tarleton and Hancock, 1996; Willmer et al, 1995], an alternative approach is to simulate the cake-building process. One of the most obvious ways to achieve this is by the use of computer simulation. For the purpose of the current work, it is desirable to build a model of the cake upon which a fractal analysis can be performed. A number of mechanisms for building such a model simulation have been suggested in both two and three dimensions (the latter being more closely related to a filter cake) and a combination of these has been found suitable for the scope of the work presented here.

Possibly the closest mechanism to the processes occurring inside and around a filter cake is agglomeration. Literature shows that considerable work has been done on both aggregation and agglomeration, with the differences between the two sometimes being blurred. The Larousse Dictionary of Science and Technology [1995] defines an aggregate as an "Assemblage of powder particles which are loosely coherent" and an agglomerate as an "Assemblage of particles rigidly joined together, as by partial fusion or by growing together". To avoid confusion, this section of the report uses the terms *agglomeration* and *agglomerate* to describe the structures built, as the particles concerned are more than loosely joined together.

1.2.1 Ballistic Agglomeration

Perhaps the simplest way of building a particle system is to use *ballistic agglomeration.* This involves particles moving in straight lines through a pre-defined space until either contact is made with a central seed particle or another trapped particle, whereupon the arriving particle will stick, or the arriving particle will leave the space and another particle will be released. The releasing and sticking of particles continues until an agglomerate of suitable size has been constructed (Figure 1.4). The easiest way to simulate this type of agglomeration in two dimensions [Fryer and Laurence, 1995] is to define a seed particle at the centre of a circle and randomly select an entry and exit point on the outside of the circle for another particle. If the straight line trajectory followed by these two points passes through the particle(s) already in the agglomerate, then the new particle will stick and the process is repeated. If the trajectory does not take the new particle on a path towards the agglomerate, new entry and exit points are chosen (again at random) until an impact occurs. In this way, large agglomerates (in the magnitude of 10^5 particles are not uncommon) can be built and analysed.

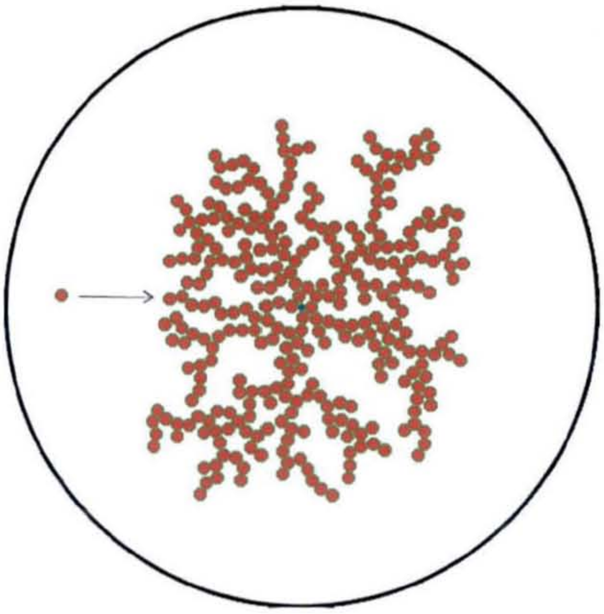


Figure 1.4 - An example of ballistic agglomeration in two dimensions

1.2.2 Diffusion Limited Agglomeration

Another mechanism for agglomerate growth is *Diffusion Limited Agglomeration*, or "DLA", as proposed by Witten and Sander [1981]. This has been described [Kaye, 1994] as similar to 'a drunk staggering home at night' because the particle is sent on random trajectory that may or may not cause it to collide with the agglomerate (shown in Figure 1.5). A DLA agglomerate takes more computing time to construct as the possibility of collision must be checked after each step taken, and (in principle) a particle might never reach the agglomerate at all. Such a process produces 'critical' agglomerates with 'scale-independent correlations over an arbitrarily large range of distances', in other words, self-similar fractals.

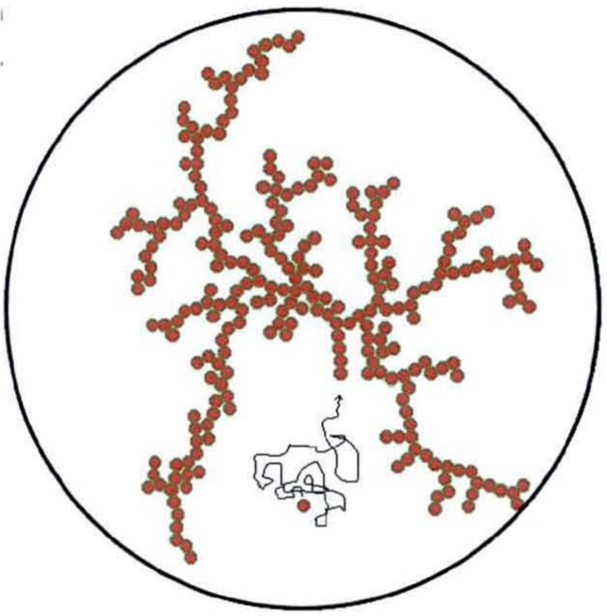


Figure 1.5 - An example of diffusion limited agglomeration in two dimensions

1.2.3 Mixed Agglomeration

A third suggested mechanism is to use a combination of DLA and ballistic agglomeration. This can be achieved in two ways. The first is to have successive steps of DLA followed by a ballistic step, with the second possibility having random step lengths for the DLA, thus allowing the particle to take larger steps in various directions.

In addition to the ballistic, DLA and mixed models for agglomeration, more complicated simulations have been used in an attempt to more closely mimic particle systems, in both this and other works.

1.2.4 Monte Carlo Simulation

Giona and Patierno [1993] describe a rigorous approach to agglomerate simulation called the 'Monte Carlo simulation' in which a number of parameters are taken into consideration when building an agglomerate.

The agglomerates were built by releasing particles from a 'bulk vapour phase' and allowing them to move in a controlled manner until they made contact with the growing agglomerate. Taking the current work as an example, the bulk vapour phase can be thought of as the suspension of mineral particles in the fluid, whilst the growing agglomerate is the filter cake forming at the base of the filtration cell. Whilst perhaps not an ideal approach to the simulation of such systems, the Monte Carlo method does allow for the characterisation of simulations in order to relate their physical properties to the mechanisms by which they were constructed. Giona and Patierno's work provides the basis for the filtration simulations described in Chapter 2.

The parameters used by Giona and Patierno include a 'downward probability', an 'occupation probability' and finally a 'sticking probability'. These three parameters can be linked to physical quantities, respectively the Peclet number, *Pe*, bulk phase concentration, *c*, and Boltzman factor, k_B. The relationships for these parameters are:

1. *Downward Probability, P_D*

The downward probability is the probability that the released particle will travel *towards* the agglomerate (ballistic motion) rather than in a random trajectory (diffusive motion).

$$Pe = \frac{3}{2}\frac{4P_D - 1}{1 - {P_D}^2} \tag{1.27}$$

The downward probability is set to simulate either DLA or ballistic mechanisms, or a mixed mechanism somewhere between the two. For DLA, P_D is set to *1/2d* where *d* is the Euclidean dimension of the simulation (2 or 3). Thus, for a two-dimensional lattice (*d*=2), the Peclet number would be set to zero. For ballistic agglomeration, the downward probability is set to one (all particles will move toward the agglomerate), giving a theoretical Peclet number of infinity.

2. *Occupation probability, f_m*

The occupation probability is related to the concentration of the system thus:

$$f_m = \frac{c}{c_{\max}} \tag{1.28}$$

where

$$c_{\max} = \frac{1}{\sigma^{3/2} N_A} \tag{1.29}$$

where σ is the particle cross-sectional area, N_A is Avogadro's number and $c_{\max}$ is the maximum concentration in the bulk phase.

The occupation probability will determine how many particles are moving out of the bulk phase at any one time. As the simulation is carried out on a square lattice, no two particles can occupy the same site. This means that as the number of particles in the agglomeration region increases, clusters are likely to be formed as the possible sites for particle attachment decrease. The values for f_m range between zero (single particle in the bulk phase) to one (maximum concentration of particles in the bulk phase). The concentration influence has also been studied [Pencea and Dumitrascu, 1995] for cluster-cluster agglomeration to show how fractal dimension varies with varying concentrations.

3. *Sticking probability, s*

The sticking probability dictates whether a particle will attach to the main sediment, and can be related to the Boltzman factor thus:

$$s = \exp\left(-\frac{E_a}{k_B T}\right) \tag{1.30}$$

where E_a is the activation energy for the agglomeration and T the absolute temperature. The values for s are set between zero (no particle in contact with the agglomerate will stick) and one (every particle will stick to the agglomerate if in contact). Giona and Patierno were concerned only with a sticking probability of one, so the influence of this parameter has not been demonstrated.

Whilst the Monte Carlo system is widely recognised as a method of creating random-event situations, these factors have been chosen by Giona and Patierno to mimic the specific system under study. This particular Monte Carlo simulation was chosen for this work due to its close resemblance to filtration structures.

The Monte Carlo simulations of Giona and Patierno showed that the agglomerate density and therefore the fractal dimension (high-density structures generally have lower perimeter fractal dimensions) are mainly dependent on the downward probability (Peclet number). However, when the value of downward probability is low, the mechanism is more diffusion controlled and the density is more sensitive to the concentration influence.

Using the three parameters described above, it is possible to simulate agglomerate growth on a square lattice as illustrated in Figure 1.6.

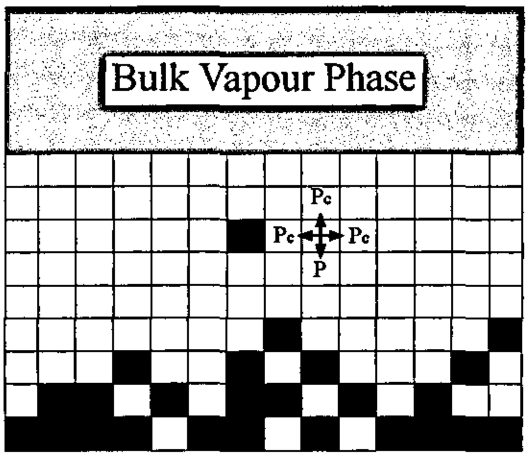


Figure 1.6 - Grid used for 'Monte Carlo' simulation of a particle agglomerate [Giano and Patierno, 1993]

A particle is released from a random point in the bulk vapour phase, with the frequency of release determined by the value of f_m. As the released particle enters the agglomeration zone (the grid), it is able to move into adjacent squares on the grid; the movement being controlled by a given downward probability, P_D. The probability that the particle will move into any given square is given by p_c where $p_c=(1-P_D)/3$. When the particle finally reaches the growing structure at the base of the grid, the sticking probability, s, determines whether it will stick to the structure, or move away from it back into the phase above.

Definite changes in the structure of agglomerates are seen when the various parameters are altered. The main contributing factor in the agglomeration is the downward probability factor (Peclet number). Giona and Patierno found that when Pe is high, the influence of bulk concentration is negligible. On the other hand, for low values of Pe (indicating DLA is the controlling factor), the agglomeration is more sensitive to f_m. Figure 1.7 shows how the structure of the agglomerate varies with f_m and Pe.

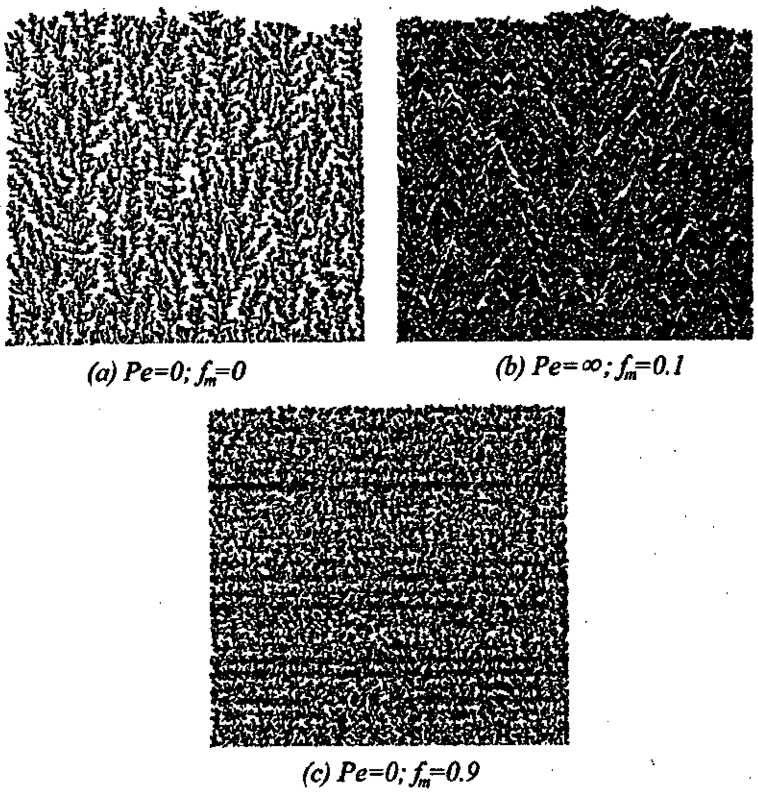


Figure 1.7 - Structure of simulation shown varying with construction parameters [Giano and Patierno, 1993]

The way an agglomerate forms can be described by a density factor, φ which varies with Pe and f_m as shown graphically in Figure 1.8. The three sets of data relate to the three models in Figure 1.7.

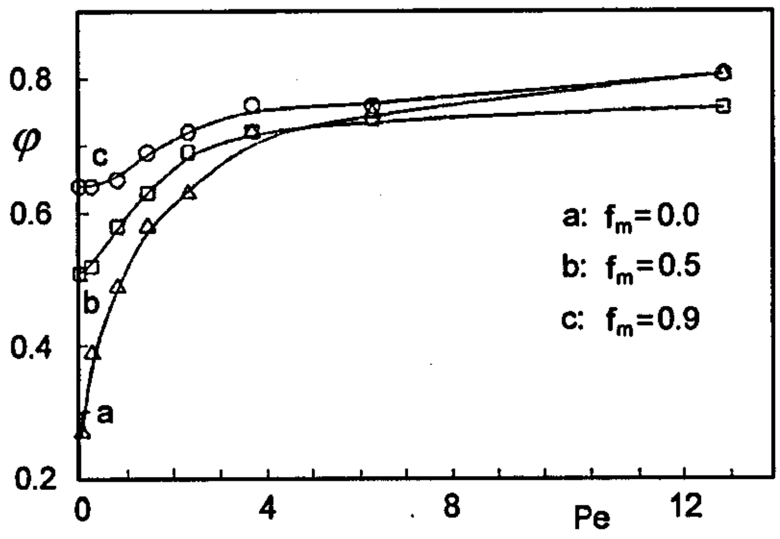


Figure 1.8 - Density factor, φ of simulations in Figure 1.7 vs. Pe [Giano and Patierno, 1993]

The Monte Carlo model developed by Giona and Patierno is a good example of particles moving in a confined system, and therefore has been chosen as the basis for the simulations discussed in the work presented here.

1.2.5 'Hit Rate'

A method has been suggested [Meakin, 1988] to reduce the degree of 'randomness' in the DLA model (that is to increase the structured appearance of the simulations). The method involves introducing a factor that will only allow a particle to join the agglomerate if the site of contact touched has been 'hit' a certain number of times (m). Removing the randomness is reported to give clusters the overall appearance of clusters comprising many more particles (for instance, $m = 2$ forms a structure similar to that containing around 100 times more particles). It has been suggested, therefore, that large values of m could be used to explore agglomerates where the number of particles approaches infinity. Figure 1.9 shows how the structure of the agglomerates varies with m. Agglomerates (a)-(d) have values of m of 2, 3, 5 and 100 respectively. As m is increased, so the structure takes on a more cross-like appearance. This shape is due to the restrictions of the lattice upon which the agglomerate was built, and would suggest that the lattice restricts the movement of particles in such a way as to make the agglomerates appear far from natural.

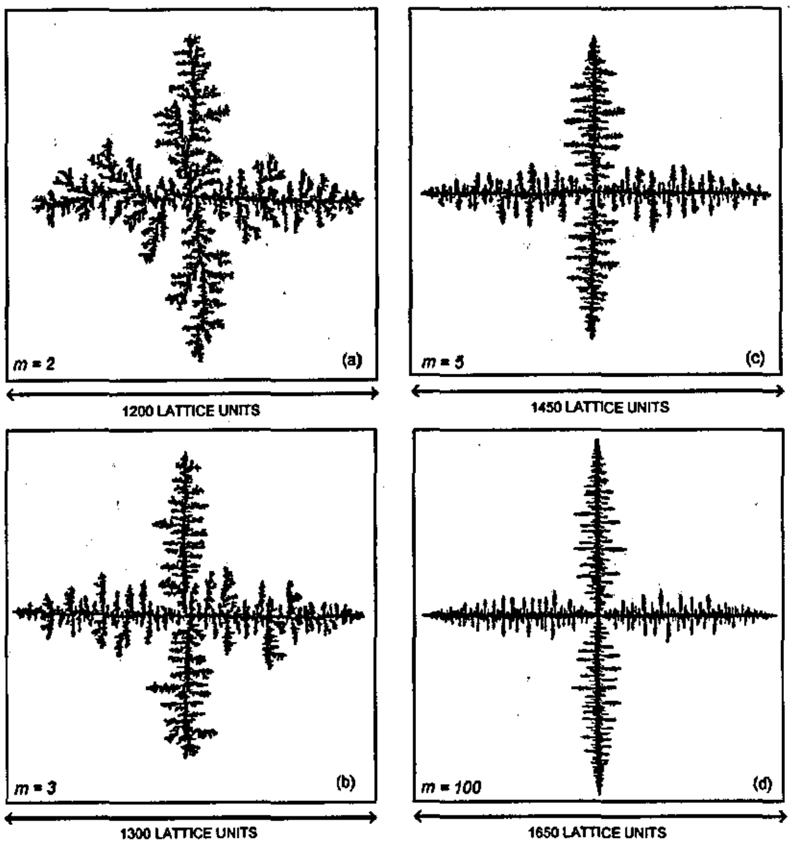


Figure 1.9 - Variations of agglomerate structure with m *[Meakin, 1988]*

1.2.6 Cluster-Cluster Agglomeration

Cluster-cluster agglomeration (CCA) models were invented independently in 1983 by Kolb *et al.* and Meakin and use a more complex procedure than simple DLA. As the name suggests, CCA involves particles forming clusters, which then agglomerate with each other, before reaching the main agglomerate structure. This is probably a more realistic simulation than others, as it would be difficult to imagine particles not interacting with each other in an agglomeration situation. Unfortunately, it requires more computing power than single particle models. There exist a number of different models, most of which work on some form of brownian motion principle, as with DLA. Some processes that have been suggested include a 'D' variable CCA model [Jullien, 1994] which encompasses Diffusion-Limited, Ballistic and Chemical-Limited cluster-cluster agglomeration, and a model by Meakin [1986] which allows clusters to rotate around their initial point of contact until a firm, fixed contact is made.

Figure 1.10 [Meakin, 1986] shows three examples of structures built using cluster-cluster agglomeration. Figure 1.10 (a) shows a typical individual cluster of 16,384 particles. The model allows clusters to stick together on first contact, but then they are allowed to rotate in a random direction (clockwise or anti-clockwise) until they become joined by two or more of the particles in the clusters. Once the clusters have joined together, they are considered to be one cluster, and can then join with other similar groups of particles, or individual particles in the structure. Figure 1.10 (b) and (c) show the results of two different models used to produce the agglomerates. Figure 1.10 (b) shows the structure formed when two of the clusters from Figure 1.10 (a) are joined and one cluster or the other (selected at random) is rotated about the point of contact. Figure 1.10 (c) is similar to Figure 1.10 (b), but allows reorganisation within the cluster itself.

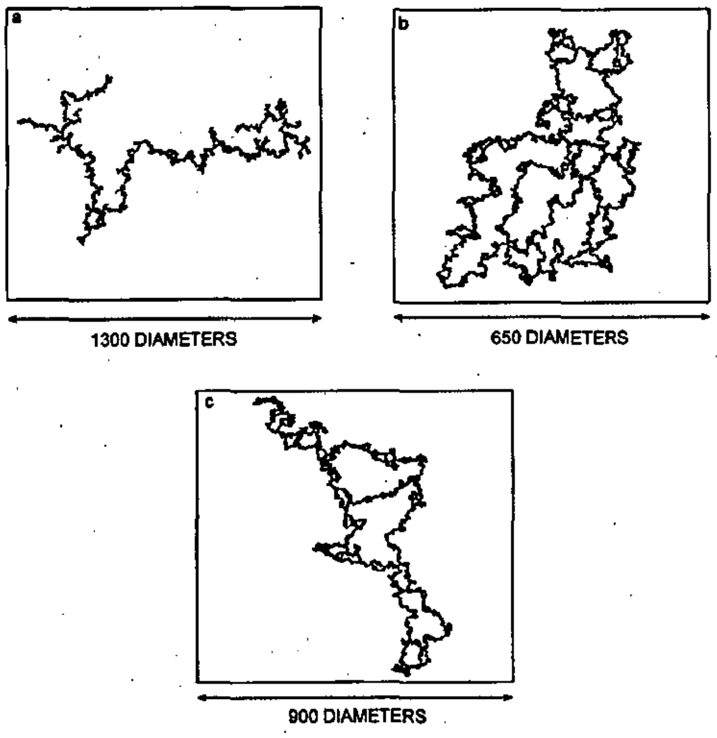


Figure 1.10 - Examples of cluster-cluster agglomeration showing the variation in shape with and without reorganisation [Meakin, 1986]

1.2.7 Other models

Other models have been suggested which include some combination or permutation of the previously described mechanisms. One of the earliest models considered in this literature survey is attributed to Vold [1963] and can be considered a ballistic agglomeration model ('successive addition of primary particles, taken as spheres of unit radius, to the growing agglomerate in a completely random manner'). After this, Sutherland [1970] proposed the formation of agglomerate chains, which had been observed in colloidal systems. The model of Gutsch *et al* [1995] involved monomer-cluster coagulation under brownian motion (DLA). Meakin and Skjeltorp [1993]. Kudoh *et al* [1993] developed a model for the sol-gel transition of SiO_2, which included reaction mechanisms in the CCA simulations. On-lattice simulations (described in Section 2.3) were carried out varying the 'ratio of the cross monomer' as the simulations moved from reaction limited to diffusion limited CCA, the density fractal dimension of the system was seen to fall from 1.53 to 1.42. Xiong and Pratsinis [1993] used a particle size distribution model to predict agglomeration by coagulation and sintering in order to determine size and shape characteristics of the particles involved in the process. Titus [1989] formed a one-dimensional general model of CCA. Marner and Schmickler [1989] produced an on-lattice model for two-dimensional clusters and measured the radial density of the structures formed.

The current work concentrated on single particle addition, rather than any type of cluster addition. Cluster simulations, whilst valid, would have required more computational power and thus created fewer simulations in the time available. The later work is based upon a model similar to the Monte Carlo simulations of Giona and Patierno [1993], including downward and sticking probabilities, as this seems a model that can be considered to be closer mimicking physical systems. The 'hit rate' model used by Meakin [1988(a)] was not used, as the agglomerates built appear to bear little resemblance to natural systems, especially filtration systems. Also, whilst perfectly valid for a number of systems, no reaction mechanisms were necessary for the simulations in the current work.

1.3 Fractal Structures

The fractal dimension can be thought of as a *fractional* dimension. Objects are normally considered to occupy one, two or three dimensions (line, plane or solid respectively). These are known as Euclidean or topological dimensions. However, it is also possible for a line, if tortuous enough, to fill 2-dimensional space. Figure 1.11 is an example of a single line being drawn in a repeated, non-overlapping pattern in order to fill a two-dimensional space. This particular structure is called a *Triangle Sweep* due the fact that the line sweeps across the plane in a triangular shape.

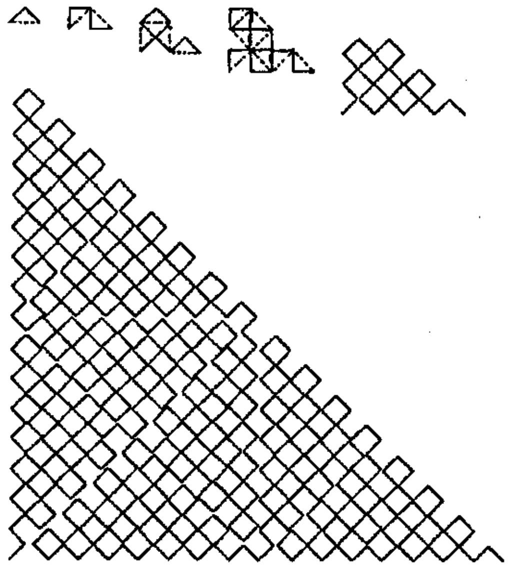


Figure 1.11 - Example of a triangle sweep, showing the first five steps and resultant structure after a number of iterations [Feder, 1998]

In such cases as the *Triangle Sweep*, the line can be described as having a dimension greater than one, but less than two - a fractional dimension. It is the fractal dimension that describes its space-filling properties or tortuosity. The more tortuous a line, the greater its fractal dimension. It can be seen in Figure 1.12 that the final (twisted) line fills more space than the straighter lines preceding it. The same tortuosity can also be applied to planes that occupy a certain amount of 3-dimensional space, although these are harder to measure and categorise.

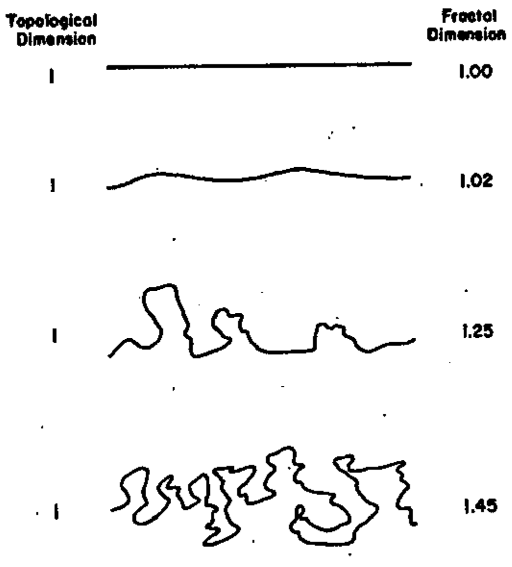


Figure 1.12 - Tortuous Lines and their Fractal Dimensions [Kaye, 1994]

There exist a number of mathematical fractals, which are said to have an infinite perimeter, due to the equations used to produce them. These include the *Koch Triadic* and *Quadric* curves or islands and *Sierpenski's Gasket* and *Carpet.* The first of these (the Koch Triadic curve) is formed [Feder, 1988] by starting with a straight line and continuously expanding it to 4/3 its original length, as shown in Figure 1.13.

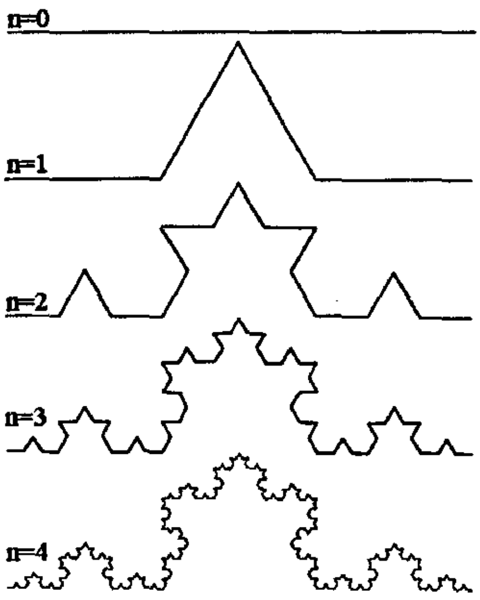


Figure 1.13 - Koch Triadic Curve [Feder, 1998]

The *Koch Triadic Curve* has a fractal dimension of ln 4 / ln 3 ≈ 1.2628, due to the nature of its construction. If the curves are made with an equilateral triangle (Triadic) or square (Quadric) as their respective starting points, *Triadic* and *Quadric Islands* are formed. Figure 1.14 shows the first three stages of formation of these islands.

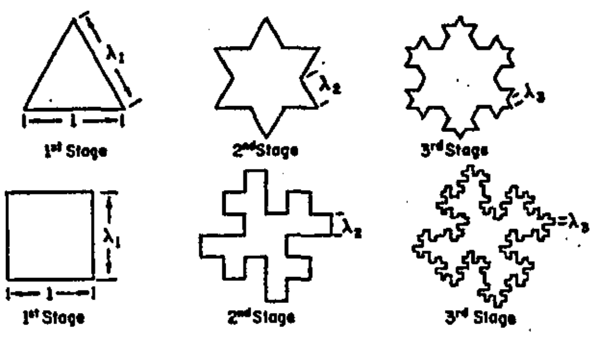


Figure 1.14 - Triadic and Quadric Islands [Kaye, 1994]

Sierpenski's Gasket (Figure 1.15) is also formed using an equilateral triangle as a starting point, but in this case, a central triangle is removed from the structure, this is continued "to infinity" forming the shape shown below (fractal dimension ln 3 / ln 2 ≈ 1.58):

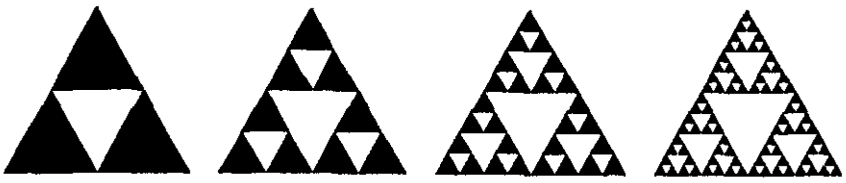


Figure 1.15 - Sierpenski's Carpet [Feder, 1998]

Similar to the Carpet is Sierpenski's Gasket (Figure 1.16), which follows the same mechanism, but using squares instead of triangles to give a fractal dimension of ln 8 / ln 3 ≈ 1.89:

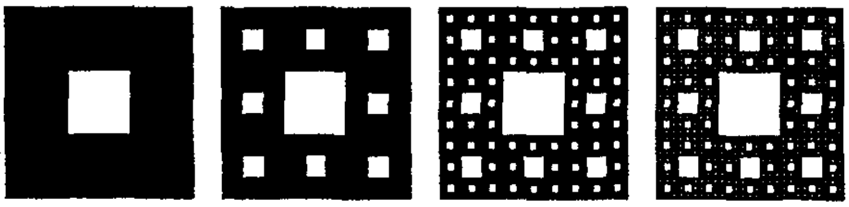


Figure 1.16 - Sierpenski's Gasket [Feder, 1998]

A more complicated fractal, formed in three dimensions is the '*Menger Sponge*' (Figure 1.17). The construction of this shape is more complicated than that of the 2-dimensional shapes. It involves starting with a solid cube, and removing the centre cores in each of the three planes and repeating the process for the cubes then formed within the original. The process involved is best explained with a diagram of the sponge itself.

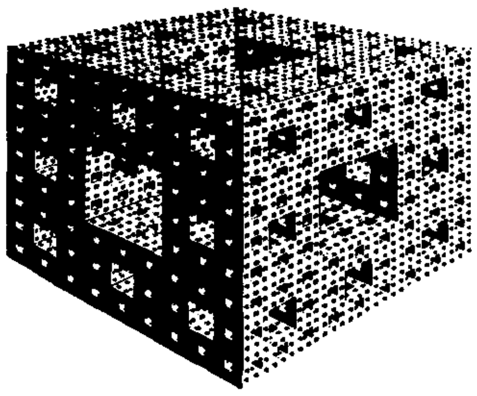


Figure 1.17 - Menger Sponge [Kaye, 1994]

The examples above also show one of the definitions of a fractal that they are self-similar at any level of scrutiny. In other words, any section should be a reduced replica of the whole. Whilst this is true for mathematical fractals, due to the way in which they are formed, it is not strictly speaking true for natural fractals. This means that one might encounter two different fractal dimensions for the same shape depending on the level of scrutiny applied. For instance Clark *et al.*, 1992 stated that it is possible to have a macroscopically rough shape consisting of microscopically smooth particles. The Richardson plot (discussed in Section 1.4.1.2) would show a high fractal dimension for the low 'magnification', but low fractal dimensions for the closer observation. It is then necessary to decide which of the values (if any) is more relevant and act accordingly upon that decision.

The current research is more interested in generating 'random' fractal shapes that are more likely to occur in a filtration system. This also means that varying fractal dimensions are a possibility.

1.4 The Fractal Dimension

Various methods have been employed in the measurement of fractals, both real and simulated. For real, or naturally occurring fractals (e.g. groups of particles or structure surfaces), software is usually employed to initially obtain an image of the system under scrutiny and subsequently to prepare and analyse this image. The use of these types of software is the basis for the majority of fractal measurements, although the precise methods used vary according to the type of structure being analysed and to a certain extent, the mechanism preferred by the author(s) involved. For 'simulated' fractals, the measurement of the perimeter is much the same, but the actual fractal structure is built up from constituent parts, usually by computer. A number of algorithms exist for this purpose, and these are discussed in Section 1.4.1.

1.4.1 Measuring Fractal Dimensions

The *general* relationship for a fractal structure is that the mass of the body is related to its length such that:

$$M \sim l^{D} \tag{1.31}$$

where D is the fractal dimension.

A number of methods for measuring fractal dimensions have been proposed by various authors and these are briefly described below. The fractal dimensions obtained using the different methods will vary due to the nature of measurement. It is therefore useful to decide upon the most suitable method(s) and only compare results between like techniques, otherwise, the results can become meaningless.

1.4.1.1 Feret Diameter

For all the methods which require a grid to be constructed, or a measure repeated and changed (for example structured walk step length), a normalised measure is used. This measure is usually some fraction of the maximum *Feret diameter* of the shape in question. The maximum Feret diameter is calculated [Kaye, 1994] as the greatest projected length of the profile of the structure. Likewise, the minimum *Feret diameter* is the shortest projected length of the profile. Figure 1.18 shows a talc particle with its maximum and minimum Feret diameters superimposed. The Feret diameters[1] are equally valid for agglomerates or other structures, as well as individual particles as shown.

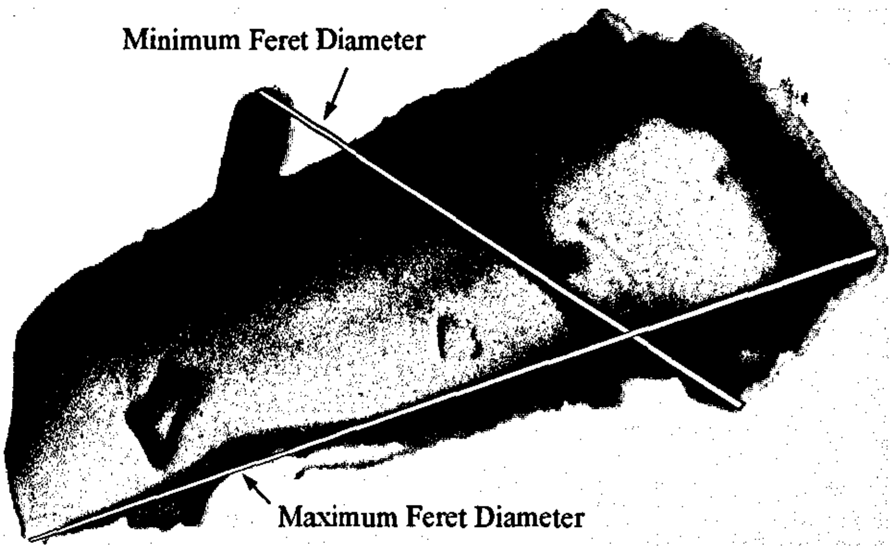


Figure 1.18 - Talc particle showing maximum and minimum Feret diameters

[1] The Feret diameters used throughout this work have been calculated using binary images and a Turbo Pascal ® program written specifically for the project.

When constructing grids or using structured walks or circle covering techniques (e.g. Minkowski's second sausage logic, Section 1.4.1.3), it has been found [Kaye, 1994] that generally, the optimum grid sizes, stride lengths or circle diameters lie between 0.08 and 0.32 times the maximum Feret diameter.

1.4.1.2 Structured Walk

One of the most widely used methods for determining fractal dimension is the structured walk technique [e.g. Trottier, 1987]. This method involves choosing a starting point on the perimeter of an object - a classic example is the coastline of Great Britain - and stepping around with a real or virtual pair of dividers to evaluate the perimeter, P, where:

$$P = n\lambda \tag{1.32}$$

or, more correctly,

$$P = (n + \alpha)\lambda \tag{1.33}$$

where λ is the step length (width of the dividers), n is the number of steps of that length taken around the perimeter and α is a fractional step length that must be taken to complete the perimeter measurement. As the step length gets smaller (tends towards zero), so the perimeter of the agglomerate becomes larger (tends towards infinity).

Three types of structured walk can be used (see Figure 1.19)- *outswing*, *inswing* or *alternate* (a combination of the two). As the name suggests, the *outswing* method involves moving the dividers in an arc outwards from the perimeter until they touch the next outside edge. The *inswing* method is similar, but in this case, the dividers are [2]moved in an arc on the inside of the shape, until they meet the next inside edge. The *alternate* method involves alternating the inswing and outswing methods. All three methods give similar results, with the *alternate* method being probably the most reliable, but also the most difficult to automate via a computer algorithm[2].

[2] The current work has used the *outswing* method as it is more reliable than the *inswing* (and therefore *alternate*) method for measuring the perimeter of solid (or near-solid) objects as opposed to the more open (or perimeter-only) type of structure seen in Figure 1.19.

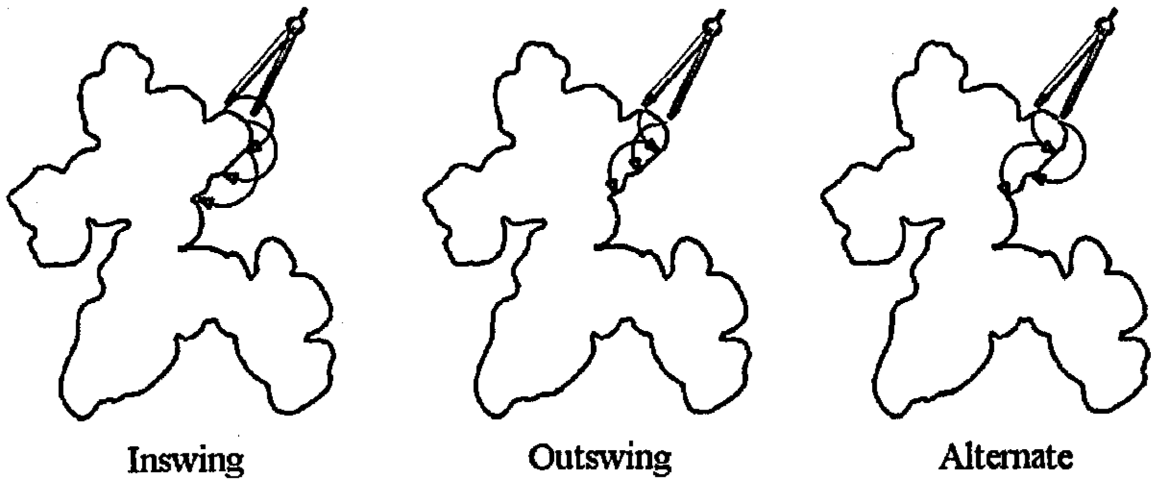


Figure 1.19 - Examples of the three Structured Walk methods, as used on a carbon black particle [particle used by Kaye, 1994]

There are however, problems with the structured walk method of perimeter measurement. It is possible for the dividers to become 'stuck' inside the shape, whereby the arc drawn cannot move outside of a particular feature. Another concern is that it is possible for the walk to miss features such as 'gullies' that appear in the structure by walking over their 'entrance'. These types of features would markedly increase the structured walk fractal dimension of a shape. Missing out gullies is not actually a problem, as the detection of surface properties at varying levels of scrutiny is the main feature of measuring the fractal dimension of a structure's perimeter.

The fractal dimension, D_s, is calculated by plotting the measured perimeter against step length on a double logarithmic scale known as a Richardson plot (Figure 1.20).

If the system being studied has a fractal (that is, fractional), rather than Euclidean (integer) dimension, the relationship between perimeter and step length can be expressed as:

$$P = \lambda^{(1-D_s)} \tag{1.34}$$

Thus, taking logarithms of both sides of Eq. (1.34) yields:

$$\ln P = (1-D_s) \ln \lambda \tag{1.35}$$

and a straight line on the Richardson plot with a slope of $1-D_s$.

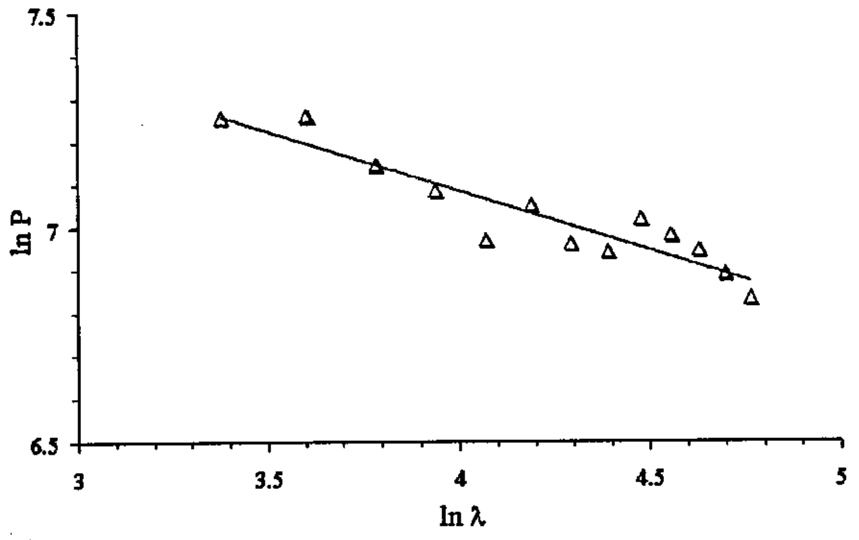


Figure 1.20 - Typical Richardson plot for two dimensional seed agglomerate; D=1.278

It should also be noted that there can often be two different slopes on a Richardson Plot. Figure 1.21 is a typical example of a shape with two distinct fractal dimensions. This indicates that the system being studied is not completely 'self-similar' and the fractal dimension changes below (or above) a certain level of scrutiny.

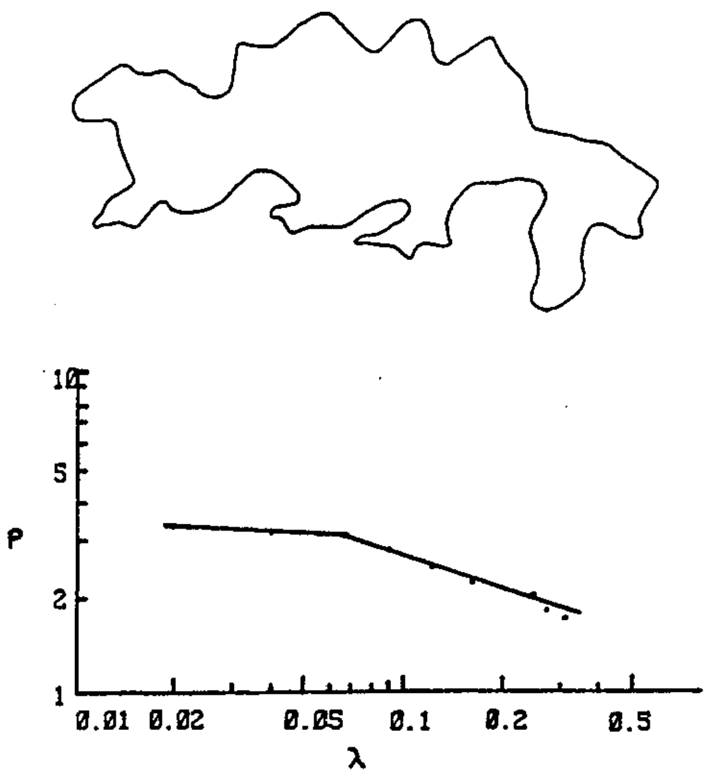


Figure 1.21 - A Typical Irregular Shape Exhibiting Varying Fractal Dimension [Trottier, 1987]

1.4.1.3 Minkowski's "Sausage" Logic (first and second)

So-called because the shape of the system resembles links of sausages, these methods use interlinking (first) or touching (second) circles of a known radius (equivalent to the divider step length) drawn along the line. Once the line has been covered by these circles, the area covered is measured and with the known circle radii, a theoretical length for the line can be calculated. The circle radius is then changed and the operation repeated, in much the same way as the step length with the structured walk technique, until enough circle sizes have been used to yield the fractal dimension. The perimeter or length of the line will increase with decreasing radii as the larger circles do not show as much detail as the smaller ones. The data obtained from this method are treated in the same way as that form the structured walk, plotting the logarithms of the perimeter against circle radii.

1. *Minkowski's First 'Sausage' Logic*

This process requires the placing of *interlinking* circles over each point on a perimeter to form a 'ribbon' around the shape being studied. Figure 1.22 shows a ribbon superimposed over the line to be studied. For a circle of radius ε, a ribbon of width 2ε will be formed along the line. Without knowing the number of circles used for the overlaying, this ribbon can be stretched out and its area computed. The area is then divided by 2ε to give a value for the perimeter.

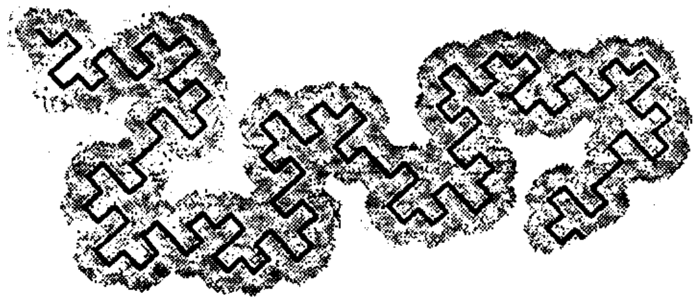


Figure 1.22 - Minkowski's First "Sausage Logic" [Kaye, 1994]

2. *Minkowski's Second 'Sausage' Logic*

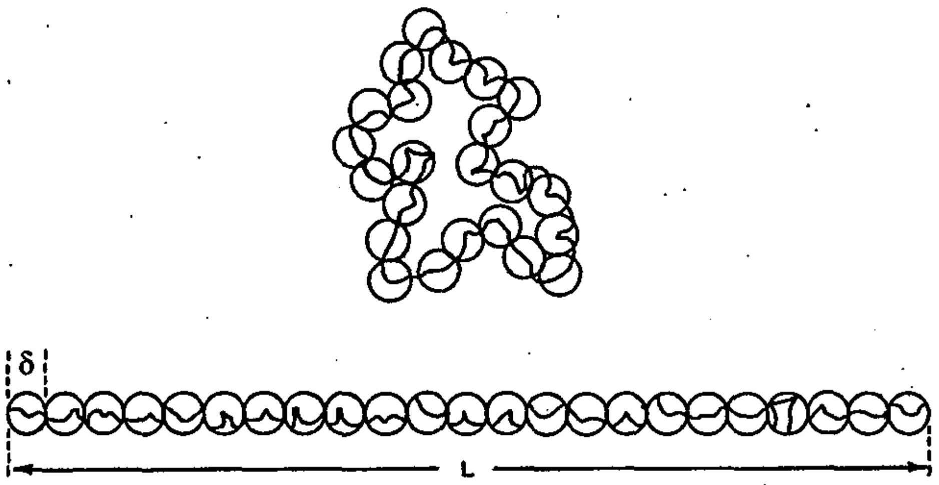


Figure 1.23 - Minkowski's Second 'Sausage' Logic [Kaye, 1994]

Minkowski's *second* sausage logic method uses *touching* circles of a uniform diameter, δ, to cover the perimeter of the shape under scrutiny (see Figure 1.23). Once the perimeter has been completely overlaid with circles, a 'ribbon' replaces the circles. The area of this ribbon is then divided by the diameter of the unit circle to give a perimeter estimate. The circles are then decreased in size and the procedure repeated. The perimeters can then be plotted against the circle diameters to which they relate on a Richardson plot and the fractal dimension calculated.

1.4.1.4 Dilation Principle

The dilation method involves taking the structure under scrutiny and dilating its boundary. Trottier [1987] used this principle to ascertain the length of the coastline of Britain and assessed the accuracy of the results by comparing the fractal dimensions of known objects, namely Koch's Triadic and Quadric islands. Figure 1.24 shows the dilation principle applied to the coastline of Great Britain. It can be seen that as the perimeter is dilated, less of the detail is apparent, in much the same way as with increasing the steplength during a structured walk measurement. The fractal dimension is calculated as the slope of the plot of the perimeter of each dilated shape against the width of the perimeter on double logarithmic axes. The results obtained from these measurements compared favourably with the theoretical values (1.24 cf. 1.26 for the Triadic curve and 1.48 cf. 1.50 for the Quadric curve).

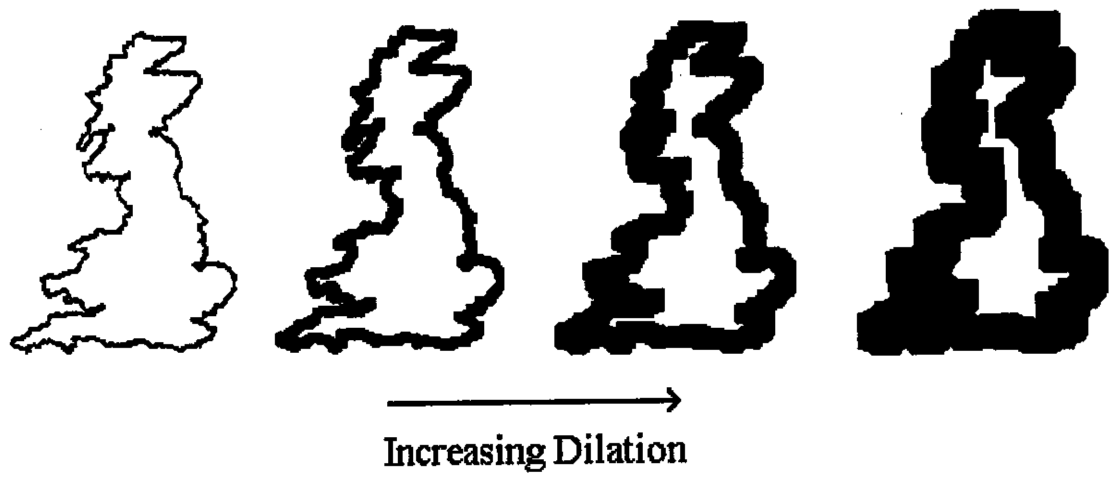


Figure 1.24 - Coastline of Britain undergoing dilation technique [Trottier, 1987]

1.4.1.5 Equipaced Method

Kaye [1994] describes another method for measuring the perimeter of a random shape. The method employed is to mark out equi-spaced points on the perimeter of the profile, and form each side of a regular polygon by creating a line between a set number of these points. The number of points between each side is then varied, as for the step length in the structured walk method, and a Richardson plot produced to measure the fractal dimension. This method is illustrated in Figure 1.25 as applied to the outline of a carbon black agglomerate (a). The results (b) again show a change in the fractal dimension as the scale of scrutiny (number of points per line, λ) is altered.

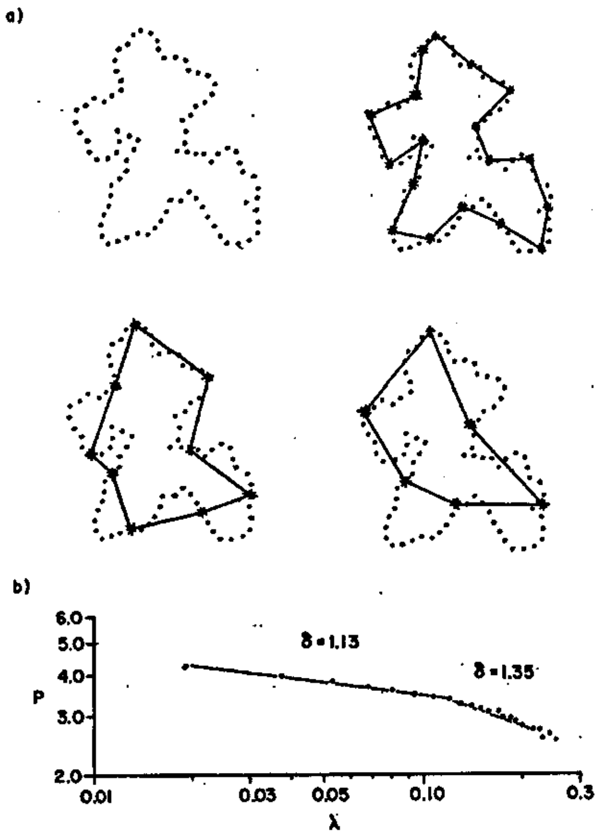


Figure 1.25 - Use of Equipaced Method as applied to Carbon Black silhouette [Kaye, 1994]

1.4.1.6 Co-ordinate Averaging

Co-ordinate averaging [Adler and Allen, 1994] involves digitising a perimeter image and mapping each of the co-ordinates in that image. Each of the co-ordinates is then 'averaged' with an increasing number of neighbouring co-ordinates using a computer algorithm, with new shapes being created as more iterations are performed. The neighbouring co-ordinates are found by using two linear arrays of x and y co-ordinates in a 'wrap-around' list. For each shape, the perimeter (P) is measured by counting the neighbouring co-ordinates using the computer (NOT with any of the other methods of perimeter measure, e.g. Structured walk) and the step length (L) is then calculated from Eq. (1.36):

$$L = P.\, Av / C \qquad (1.36)$$

where Av is the number of neighbours used to average each of the co-ordinates and C is the number of co-ordinates for the perimeter. The numbers shown inside the shapes in Figure 1.26 are the number of neighbours for each co-ordinate (Av).

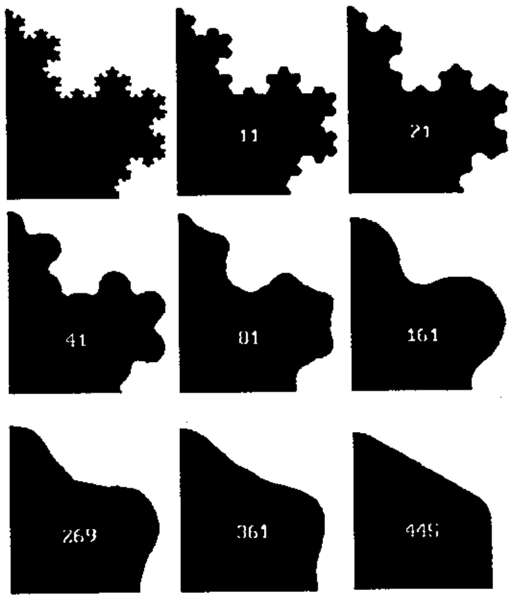


Figure 1.26 - Co-ordinate averaging of a quadratic Koch Island [Adler and Allen, 1994]

1.4.1.7 Geometrical Probability

The geometrical probability method uses the intersection of a (fractal) structure with parallel or concentric lines to determine the fractal dimension. It can be shown [Kaye *et al.*, 1992] that a fractal boundary will intersect these lines more frequently than a Euclidean boundary. By using different spacing of these parallel (or concentric) lines, the probability of intersection compared with an equivalent Euclidean shape (as in Figure 1.27) can be calculated. The probability of these intersections shows a good correlation with the boundary fractal dimension (measured using the equi-paced method) 'with an accuracy of not less than 1%'.

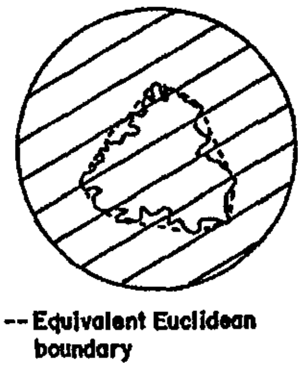


Figure 1.27 - Example of Geometric Probability technique, showing intersection of both fractal and Euclidean boundaries [Kaye, 1992]

1.4.1.8 Box counting technique (Mosaic representation)

The *box counting method* [Trottier, 1987; Víquez *et al*, 1995] involves laying a grid over the perimeter or object to be studied, as shown in Figure 1.28. The number of squares of the grid that intersect the perimeter are counted, and the fractal dimension calculated in a similar way to the structured walk technique, the difference being that the stride length is equivalent to the width of the individual boxes in the grid, and the number of strides equivalent to the number of boxes. The grid size is then changed, and with a decreasing box width, it is possible to analyse the perimeter or object in more detail.

The fractal dimension is obtained from the relation:

$$N(b) = b^{-D_b} \tag{1.37}$$

where N is the number of boxes of size b, and D_b is the fractal dimension.

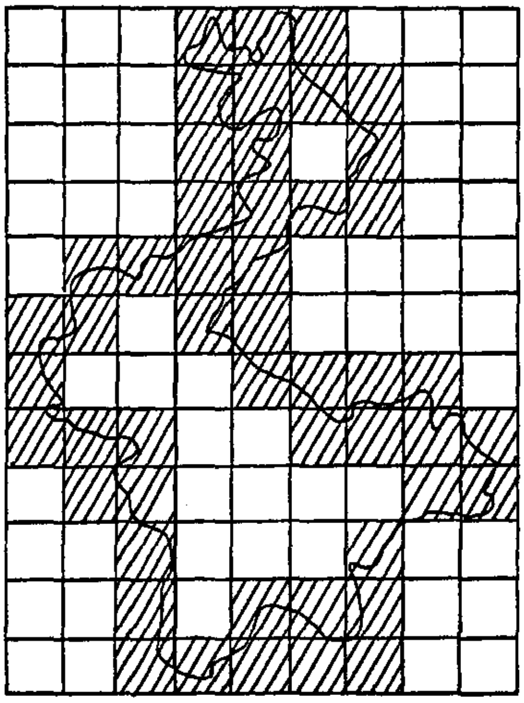


Figure 1.28 - Example of the box counting technique [Trottier, 1987]

The box counting method is suitable for retrieving the mass fractal dimension of two dimensional objects, as it is able to describe how the mass is distributed within the object in question. If the mass is distributed in large, discreet areas, the object will have a low fractal dimension, as the mass will drop rapidly with decreasing box size, whereas if the mass is distributed more evenly, the mass will not fall off as much with decreasing box size.

1.4.1.9 Radius of Gyration

Another method that can be used to calculate the mass fractal dimension of a system is the *radius of gyration*. This is calculated by comparing the size (mass) of a particle and its radius of gyration, R_g, which is calculated as:

$$R_g = \sqrt{\frac{I_g}{A_t}} \tag{1.38}$$

where I_g is the moment of inertia around the centre of gravity (for the agglomerate) and A_t is the total area (mass) of the agglomerate. The radius is then plotted against the total area for varying numbers of particles in the system, again on double logarithmic axes, with the fractal dimension, D_{rg} being equal to the reciprocal of the slope.

1.4.1.10 Enclosing Circles / Spheres

As the name suggests, the enclosing circles (two dimensions or '2-D') or spheres (three dimensions or '3-D') techniques require the structures under scrutiny to be enclosed by circles or spheres (all referred to as circles for brevity) of progressively increasing radius. These circles have their centres at the centre of gravity of the object and extend from close to the centre out to the boundary of the fractal object in a uniform way. The mass contained within each circle is then plotted against the circle size on double logarithmic axes, and the fractal dimension, D_e calculated as the slope of the resulting plot. Figure 1.29 shows a typical 2-D agglomerate measured using the enclosing circle technique.

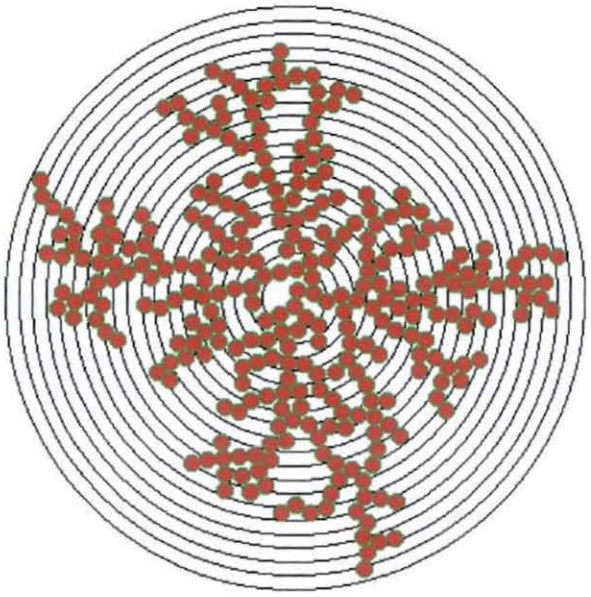


Figure 1.29 - Enclosing Circle Technique applied to a Typical 2-D Agglomerate

1.4.2 Fractal Surfaces

The methods described in Section 1.4.1 are primarily concerned with measuring the space filling property of a line (one dimensional analysis), but it is also useful to be able to quantify the roughness (and hence fractal dimension) of a surface (two dimensional analysis). One method of analysing a surface is to form a 'coastline' around the object being measured and calculate the fractal dimension of this coastline. The fractal dimension is then given as $D+1$ where D is the dimension of the coastline [Russ, 1992]. This method, however, has not gained wide acceptance. Other methods include [Dubuc *et al*, 1989] the 'covering blanket', difference statistic and variation methods, which attempt to measure the surface directly. The work presented here, however concentrates only on the measurement of the fractal dimension of two-dimensional projections of three-dimensional objects, as the algorithms required for the computation of three-dimensional fractal analysis were too involved for the resources available.

1.4.3 'Fractal Rabbits'

The term *Fractal Rabbits* refers to false fractals generated during a structured walk around a boundary or perimeter. The results for certain Euclidean shapes seem to show a fractal dimension for large step lengths. Fractal Rabbits are usually seen when measuring shapes with a large aspect ratio (long and thin) such as ellipses. These false fractals can be seen in Figure 1.30. The diagram shows that as the shapes become less circular and more ellipsoidal (A-D), a false fractal dimension emerges from the analysis of the shapes' perimeters.

Fractal Rabbits are so called because the fractal dimension appears from nowhere, similar to a magician's 'rabbit in a top hat' trick.

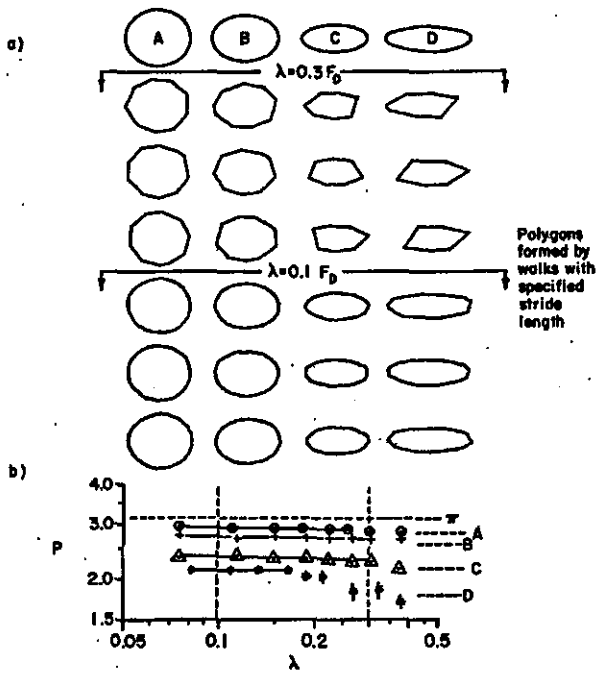


Figure 1.30 - Examples of 'Fractal Rabbits' for ellipsoidal shapes [Kaye, 1994]

Unfortunately, there is no easy method by which it is possible to determine whether the measured fractal dimension is a true dimension or a 'Fractal Rabbit' without operator interference, comparing the shape with its likely fractal dimension. 'Fractal Rabbits' are more likely to appear when analysing small shapes or using a structured walk technique with a step length greater than the recommended [Kaye, 1994] maximum.

1.4.4 The Application of fractal measurements

Fractal dimensions have been used in a wide range of applications. The applications can be broken down into the characterisation of particles, aggregates or agglomerates, surfaces and porous media. A fifth category describes the measurement of structures that do not fall into one the above.

1.4.4.1 Characterisation of particles

A number of authors have categorised individual particles by their fractal dimension, usually through the structured walk or a similar technique.

Xie and Bhasker [1993] characterised pulverized materials using both size distribution fractal dimensions and surface fractal dimensions.

The size distribution fractal dimension was calculated as three minus the slope (D = 3 - slope) of the size distribution plot (Rossin-Rammler or Gaudin-Schuhmann distributions), whilst the surface fractal dimension was calculated from by:

$$s = G_0 x^{2-D_s} \tag{1.39}$$

where G_0 is the surface area of a smooth particle of size x and s is defined by:

$$s = K_S x^2 \tag{1.40}$$

where K_S is the 'surface shape factor', or the surface area of a particle of size x. Thus, if the surface fractal dimension, D_S, is equal to the Euclidean dimension (i.e. the particle is smooth), $s = G_0$. The size distribution fractal dimension of the pulverised material ranged from 1.350 to 1.770, whilst the surface fractal dimension ranged from 2.058 to 2.584. The size distribution fractal measure is of little relevance to the work discussed here, whilst the surface roughness technique might be applied to the minerals investigated.

Ludlow and Vosen [1993] used the fractal dimension to describe the surface roughness of synthetic coal-char particles. The fractal dimension was calculated using a 'molecular tiling technique' in which the surface under scrutiny had decreasing sizes of molecules adsorbed onto it. The relationship between the coverage of these molecules and their size gave the fractal dimension (D_a):

$$V_m = \kappa \sigma^{(-D_a/2)} \tag{1.41}$$

where V_m is the monolayer coverage, σ is the cross-sectional area of the molecule and κ the 'lacunarity' (constant). The measured fractal dimensions for the agglomerates varied from 1.81 to 2.85, but with errors of up to ±0.72 (29%).

Zerda *et al* [1991] also used an adsorption technique to measure the fractal dimension of carbon black particles. They found that the fractal dimension could be calculated in much the same way as for Ludlow and Vosen [1993], using Eq. (1.42) to give the number of molecules (*N*) needed to cover a monolayer of the agglomerate under scrutiny.

$$N = S\sigma^{-D/2} \tag{1.42}$$

where σ is the diameter of the adsorbent, *S* is the Hausdorff measure of the surface and *D* is once again the fractal dimension. The results of Zerda *et al* gave the surface fractal dimension of carbon black particles as 2.2 ±0.1, giving a result with less margin of error than Ludlow and Vosen [1993].

The surface of carbon black particles was also measured using a fractal dimension by Xu *et al* [1996], again using the surface adsorption technique. In this instance, the fractal dimension was determined as 2.0 ±0.1, much the same as the result obtained by Zerda *et al* [1991] previously.

Hancock *et al* [1993] used the fractal dimension to differentiate between wear and contaminant particles. They used the dilation principle to determine the fractal dimension of individual particles taken from the system under examination and found that contaminant particles exhibited different fractal properties from wear particles, with the wear particle generally having a higher fractal dimension for the same size distribution.

1.4.4.2 Characterisation of aggregates and agglomerates

Again, considerable work has been done to characterise aggregates and agglomerates made up of a wide variety of sub-units. The fractal characterisation usually takes the form of a mass or volume to size relationship.

Jiang and Logan [1996], in an industrial application used the fractal dimension to classify aggregates. They used a number of fractal dimensions (perimeter-length, area-length and volume-length), giving fractal measures in one, two and three dimensions respectively to characterise aggregates formed in shear devices with a paddle mixer. Results showed that the perimeter-length fractal dimension varied between 1.08 and 1.13, with low variance, whilst the area- and volume-length fractal dimensions varied between 1.77 and 1.87, showing good agreement with each other, but with the volume-length fractal dimension having a higher variance than that measured using the area-length relationship.

Logan and Kilps [1995] measured the fractal dimensions of aggregates formed in different fluid mechanics environments, using a number of different techniques in order to obtain the fractal dimensions. The methods employed were two power-law techniques (size-porosity and size-area) and a size distribution technique. A number of different fractal dimensions were observed depending on the type of the structure studied, and the system in which the particles existed.

Herd *et al* [1992] used fractal dimensions to classify carbon black aggregates, this time employing both a perimeter-area and mass fractal dimension. Values for the perimeter-area fractal dimension ranged from 1.05 to 1.23 and the mass fractal dimension, relating the mass of the aggregate to its radius varied between 2.47 and 2.85.

Namer and Ganczarczyk [1994] measured the fractal dimension of sludge aggregates and found that the fractal dimension for inorganic flocs was higher (1.59-2.85) than that for activated sludge (1.4-2.0). Both size-density and mass-length fractal dimensions were used in the work.

Kim and Scarlatos [1993] used Minkowski's sausage logic to give the perimeter at various levels of scrutiny, and thus the fractal dimension of aggregated sediments. They found that the perimeter fractal dimension varied between 1.03 and 1.69.

Li *et al* [1993] investigated the fractal dimension of aggregates formed from construction materials by calculating the box-counting fractal dimension of their perimeters, after having obtained the perimeters through edge detection methods. Results showed that the fractal dimension varied between 1.05 and 1.15.

Kruis *et al* [1994] characterised agglomerated and aggregated particles in an aerosol by using a variation of Minkowski's sausage logic, covering the image of the aggregate with circles of a known diameter. They found the average fractal dimension of the system under scrutiny to be 1.59, with a distribution from 1.1 to 1.9.

1.4.4.3 Characterisation of fractal surfaces

One of the more difficult fractal measures to perform is the determination of the fractal dimension of surfaces. A number of techniques have been proposed, some more robust than others. These include the split-island technique, which involves measuring the perimeter of islands formed at certain levels of the structure (much like contour lines) and the examination of surfaces by adsorption of various molecules. The latter method can also be applied to the surface of individual particles (as in the work of Zerda *et al* [1991] and Xu *et al* [1995], above).

Fracture surfaces were analysed by Li *et al* [1995] by both 'roughness' and fractal methods. The surface roughness of a material (R_S) is defined as

$$R_S = \frac{S}{A} \tag{1.43}$$

where S is the true fracture surface and A the *apparent* fracture surface. The roughness (R_L) of the surfaces' profiles were obtained from observing the profiles of the fractures, and determining the ratio of true perimeter length to apparent length, as shown in Eq. (1.44):

$$R_L = \frac{L}{L_0} \tag{1.44}$$

where L is the true length of the projected perimeter, and L_0 is the apparent length. The perimeter roughness was related to the surface roughness of the material by Eq. (1.45):

$$R_S = \overline{R_L \omega} \tag{1.45}$$

where ω is the 'profile structure factor'.

The profile fractal dimension, D_L, and surface fractal dimension, D_S, were found to be related to the respective roughnesses by Eqs. (1.46) and (1.47):

$$R_L(\eta) = C_1 \eta^{-(D_L - 1)} \tag{1.46}$$

$$R_S(\eta) = C_2 \eta^{-(D_S - 2)} \tag{1.47}$$

where C_1 and C_2 are constants and η the size of the measuring unit (level of scrutiny) used. The profile fractal dimensions ranged from 1.037 to 1.096. whilst the surface fractal dimensions varied between 2.038 and 2.094.

Pimienta *et al* [1994] measured the surface fractal dimension of titanium bone implant material using Dubuc's [1989] covering blanket method to determine the variation in surface roughness with plasma-coated titanium plates. They found the surface fractal dimension of the material to range from 1.09 to 2.32, with the coarsely and finely coated surfaces having a higher fractal dimension than the uncoated surface. Whilst this method seems to be robust and gives reproducible results, it is difficult to code as a computer algorithm.

Antonucci *et al* [1996] investigated the surface of electrodeposits, using atomic force microscopy (AFM) to determine the surface fractal dimension. This was achieved by taking the perimeter-area fractal dimension, obtained by a split island technique. The surface fractal dimension, D_S was said to be D +1, where D is the perimeter-area fractal dimension. Results gave D_S ranging from 2.3 to 2.5 (perimeter fractal dimension of 1.3 to 1.5). This method for the calculation of a surface fractal dimension is not widely believed to be a useful or accurate technique.

1.4.4.4 Characterisation of porous media

Porous media are one application where the shape of the internal structure is important to their functionality, and therefore lend themselves to examination using the fractal dimension.

Bayles *et al* [1989] used a structured walk technique to give a perimeter fractal dimension from the islands formed by sectioning a porous media (split island technique). This technique failed, however, as the Richardson plot showed a constantly changing slope, therefore making it impossible to determine a fractal dimension. After the initial failure, the fractal dimension was successfully measured using a size-area function, giving a mass fractal dimension ('fragmentation fractal'), with the areas being measured using an erosion-reconstruction technique. This technique gave fractal dimensions ranging from 1.295 to 1.843 for those measured from computer generated porous systems and 1.56 to 1.82 for those measured from the size distribution from images of the media itself. Bayles *et al* concluded that the split island technique could not define the system in question, whereas the fractal dimension measured using a size-area relationship could help to predict the behaviour of the system, in particular the permeability.

Kolb [1990] measured the fractal dimension of a model porous medium using both monolayer and multilayer adsorption technique. The fractal dimension for the generated system was found to be 1.33 in both cases. It was concluded, however, that the monolayer adsorption method was more reliable than multilayer adsorption.

Lenormand and Zarcone [1989] measured the fractal dimension of a porous system (capillary fingering) by using a form of the box-counting technique, plotting the number of invaded pores against the scale of scrutiny. Their results gave a fractal dimension ranging from 1.83 for the slowest capillary fingering action down to 1.80 for the faster action.

Gottsleben and Hesse [1992] attempted to measure the fractal dimension of porous structures using molecular adsorption, but their results seemed to show that the porous media they were investigating did not show a fractal nature, with fractal dimensions ranging from 2.90 to 5.2

1.4.4.5 Characterisation of other systems

Fractal dimensions have also been used to characterise systems that do not readily fall into the above categories.

Banik *et al* [1993] investigated the fractal dimension of mine airways, in which a fractal measure was applied to determine the roughness of the airway surface. They noted that the profile of the airways exhibited a fractal nature, and that the variance of the thickness of the profile could be described with a fractal dimension by Eqs. (1.48) and (1.49):

$$\mathrm{VAR}(h_{i+1} - h_i) = \sigma^2 |\delta t|^{2H} \tag{1.48}$$

where h is the height of the i th strip of the airway under scrutiny, δt is the thickness of the strip and H is the characteristic roughness parameter related to the fractal dimension (D) by:

$$D = 2 - H \tag{1.49}$$

It was found by Banik *et al* that the friction factor (f) for the airway varied with the fractal dimension according to the relationship:

$$f = 0.826 - 1.011D + 0.314D^2 \tag{1.50}$$

Jaroniec *et al* [1993] investigated the correlation between the microporosity of active carbons and their fractal dimension. The fractal dimension (D) in this case was calculated using the relationship between the pore size, x a the pore size distribution function, $J(x)$ according to Eq. (1.51):

$$J(x) = x^{\frac{C}{D-2}} \tag{1.51}$$

where C is again a constant. From these calculations, the fractal dimension was found to correlate with the average micropore size ($\bar{x}$) according to the relationship

$$D = 6.44 - 6.17\bar{x} \tag{1.52}$$

1.4.5 Choice of Fractal Measurement Systems

The choice of which fractal measurement to use is an important one. The measurement technique must be suitable for the type of system being measured (2-D, 3-D, solid, perimeter etc.) and must also be able to yield statistically significant results over the range of structures under scrutiny.

The majority of the work presented in the thesis used the structured walk technique. This gave a range of perimeter fractal dimensions that could be correlated against the parameters used to build the fractal structures. Structured walks have been used on both images of individual particles and simulated agglomerates to give a value for the surface roughness of the structure under scrutiny. Both seed agglomerates and filtration simulations have been constructed in the current work (see Chapter 2) and the methods used to measure the fractal dimensions have proved successful in both cases.

In initial work, both radius of gyration and enclosing circles / spheres techniques were used to give a mass fractal dimension, which again could be correlated against set parameters. The radius of gyration technique, in particular gave good, reproducible results for the mass fractal dimension of structures built using the seed agglomeration method. Neither the radius of gyration nor the enclosing circle techniques can be applied to obtain an overall fractal dimension of the filtration simulation agglomerates, but the enclosing circle technique has been used to measure the mass fractal dimension of the interstitial spaces within two dimensional simulated cakes.

The most useful classification for the filter cakes (SEM images) was the box counting or mosaic representation technique. This was due to the starting shape of the structure (rectangular) and the fact that there was no 'age' information as far as the individual particles within the cake are concerned (necessary for the radius of gyration technique). Whilst useful when applied to a given system, the remaining methods of measuring fractal dimensions did not easily lend themselves to the research programme.

1.5 The fractal dimension of filtration systems

Relatively little work has been carried out to correlate filtration characteristics with the fractal dimension of the system, a topic which is discussed in this thesis.

Bayles *et al* [1987,1988,1989] analysed the fractal dimension of coal filter cakes by taking the boundary fractal dimension of particles in the filter cakes. The method created a virtual intersection of the surface of the filter cake with a plane and measured the fractal dimension of the resulting boundary or 'coastline'. The surface fractal dimension was then taken as the boundary fractal dimension + 1. The *absolute* slope of the Richardson Plot used to determine the fractal dimension was plotted on a double logarithmic scale against two known characteristics of filter cakes - mean hydraulic diameter and specific surface area. The second of these two plots showed a good correlation that was taken to mean that the fractal dimension might be related to the permeability of the cake. Another plot showed the ratio of absolute slope and sample porosity against porosity for the cake, again on a double logarithmic scale. This plot showed a better fit compared to the first. However, the slopes of the Richardson plots varied along their length and the gradients used for the plots were only measured at one steplength value. This means that a true fractal dimension was difficult to determine and therefore correlate with cake characteristics.

The work of Schmidt [1995] with solid/gas systems involved filling the pores of limestone dust filter cakes from bag filters with epoxy resin, taking planes through the cakes, polishing and etching them to enable the local structure to be studied through an electron microscope. This enabled the structure of the cake to be analysed by using the box-counting technique - grids with varying side lengths were laid over the micrographs and those squares containing part of the cake plotted against the length scale of the grid. The fractal dimension was then plotted against the distance from the filter surface, but no correlation was seen between the two properties, even with varying pressure (from 50 to 2000 Pa). A similar method has been used in the current work to analyse incompressible and compressible cakes formed from the filtration of solid/liquid mixtures.

1.6 Conclusion

As there is as yet no unifying description of the filtration mechanism, this work attempts to use the fractal dimension as a new descriptor of the internal structure of filter cakes, through both simulation and the characterisation of samples taken from physical filter cakes. It has been shown in the past that fractal dimensions of one form or another can be used as a discriminatory measure to characterise a wide range of systems.

However, a great deal of care must be exercised when choosing the measurement technique in order to avoid obtaining results that are not a true representation of the system being described.

Once a measurement technique has been chosen, a rigorous approach must be taken to ensure that reproducible results are obtained and that the results have a logical basis.

2 SIMULATION AND MEASUREMENT OF FRACTAL STRUCTURES

This chapter describes the method by which structures of varying type were both built and measured using computer simulation. A number of steps were taken towards the final goal of a filtration simulation, and likewise a number of methods, including fractal analysis were used to measure the properties of these agglomerates once constructed.

2.1 Introduction

A Turbo Pascal® computer program was created to perform a number of functions concerning the agglomeration of particles and subsequent analysis of the resulting structures. The main part of the program allows the simulation of particle agglomeration onto a central seed particle and the simulation of particle movements within a virtual filter cell, both of which can be performed in either two or three dimensions. After completion, the created structures could be analysed in a number of different ways, including fractal and structural analysis. As well as the ability to analyse computer simulated agglomerates (including those constructed in other environments), the program was also able to perform both structural and fractal analysis on SEM images.

Described here are the main functions of the program, including the construction and analysis of both seed agglomeration and filtration models, as well as the analysis of digitised images. The major equations governing the building and analysis are given, if not already covered in the first chapter. Also shown are a number of flow sheets, showing the outline of the user controllable parts of the program. The algorithms for the mathematics within the program are not shown, however, as described above, these are mainly covered by the equations shown. A complete listing of the program created for the project is given in *Appendix A*.

The first stage in designing the computer program was to develop models and computer algorithms to simulate the growth of seed agglomeration models. Once this was achieved it was possible to ascertain whether the parameters used to control the growth of the structures had an effect on their physical and/or fractal properties. When the seed agglomeration models and coding were completed, the next stage of the process was to establish models and computer algorithms to simulate the growth of a filter cake on a filtering surface. As with the seed agglomeration models, this was first completed in two dimensions, before moving on to the more complicated model in three dimensions.

Analysis procedures to examine the structures in a number of different ways were then coded to complete the program.

The program's operation was constantly checked for both mathematical and coding errors by introduced a number of "de-bugging" routines that allowed the programmer to visualise and/or slow down the computer's processing in order to check for errors. Once each of the routines were checked and completed, these de-bugging routines were removed in order to conserve space within the program.

2.2 Program structure

The program was arranged in many small modules, so that different procedures could share program code and avoid excessive repetition. Turbo Pascal® uses various components within a program. Units are the main blocks of the program, and are stored under unique file names. Procedures and functions are smaller sub-sections within these units, and are the parts of the program that perform the necessary calculations.

The flow sheets in Figures 2.1 to 2.5 show a simplified version of the program and the way in which simulated structures are created and analysed. Names in italics (e.g. "*Simulate*") refer to a procedure or function name, whilst the name in capitals within parentheses (e.g. "AGGLOM.PAS") refers to the filename of the holding unit.

Figure 2.1 shows the main menu routine for the program. The main routine initialises the majority of the global variables and where necessary, sets initial values for these variables; global variables are variables used by more than one procedure within the program. After initialisation, a menu procedure is called and the start up menu displayed, with the options described. *Simulate*, *Display*, *Analyse* and *Setup* are procedures that are described in Figure 2.2 - Figure 2.5. *Trace_Menu* is used to change settings for animating Ray Traced images of the simulated agglomerates and filter cakes. *DosShell* provides a DOS shell from within the program to enable disk functions etc. to be carried out without the need to shut down the program. Finally, *Ren* is used to rename the automatic files that can be generated by the program.

Figure 2.2 shows the flowsheet for the procedure S*imulate* which controls the simulation of both the seed agglomeration and filtration models. Using this algorithm, agglomerates of various types (seed agglomerates, Monte Carlo simulations and the potential others) can be built by employing various simulation parameters.

The *Carlo* and *Agglomerate* procedures called from this routine control the particle-by-particle growth of the structures. Both procedures operate from within procedure S*imulate* and are called each time a new structure is to be built. They in turn use all the geometrical and other procedures needed to construct the agglomerates.

Agglomerate is the routine used to control the growth of agglomerates built around a seed particle. The attaching particle's motion is determined by a number of different parameters explained later in this chapter (Section 2.3.1).

Carlo is the routine that controls the growth of the virtual filter cakes. Again, the movement of a descending particle is controlled by a number of parameters, discussed in Section 2.3.2.

Once all the required agglomerates are built, the particle data are stored on disk so as to be accessible for later analysis.

Figure 2.3 shows the flow sheet for the D*isplay* procedure, which displays previously constructed agglomerates without analysing them. The *Display* routine also has the capability to rotate three-dimensional seed agglomerates, both in the vertical and horizontal planes, and to save the agglomerate data again, with the rotated particles' co-ordinates.

Figure 2.4 is a very simplified flow sheet of the *Analyse* procedure, which calls the individual procedures used to analyse all the simulated structures and SEM information. To start, the *Analyse* procedure initiates a user menu to input the file name(s) of interest. The procedure then checks the type of file inputted, and automatically decides upon the type(s) of analyses possible; the type of analysis is subsequently chosen from these possibilities. After all the relevant information has been inputted, the *Analyse* procedure performs the selected analysis on the file(s). In the case of single files, the analysis results are simply displayed on screen for the user. In the case of multiple files, the results of the individual analyses are stored on disk, along with statistical information and an average value for the results obtained. The resulting file is in ASCII format, which can be readily imported into spreadsheet or other similar applications. As well as analysing the agglomerates generated by the simulation procedures, the program can also analyse data from other agglomeration simulations and digitised images of, for example, filter cake sections.

Figure 2.5 shows the *Setup* algorithm for setting the basic parameters for the entire program. These are:

- Setting the number of dimensions for a simulation (two or three dimensions).
- Setting the shape parameters of the particles in the simulations - currently inactive: only circular (spherical) particles with a variable size distribution can at present be used in the simulations.
- Altering the settings of the random number generator. The three possible states are:
    - On – a new random seed is generated for each simulation
    - Off – the random seed is set to zero for all simulations
    - Set – the random seed can be set to any value based on the computer's clock. Setting the random seed to a given value allows exact replication of simulations for, amongst other purposes, debugging
- Visual debugging – when set to 'On' this will display the path of particles in the simulations and show extra details in the analysis modules
- Visual delay – slows parts of the program down by a set amount to allow the user to view particle motion etc.
- Manual / Automatic control – allows multiple simulations with the same simulation parameters to be performed, without user interference
    - Number of automatic files – the number of simulations to perform when Automatic control is selected
- Data directory – default directory for simulation information
- Local / Network graphics directory – allows the program to be run on a networked computer, where necessary graphics 'drivers' can be found

The flow sheets shown in Figures 2.1 to 2.5 are the top level of the program, each part of which is explained in greater detail later.

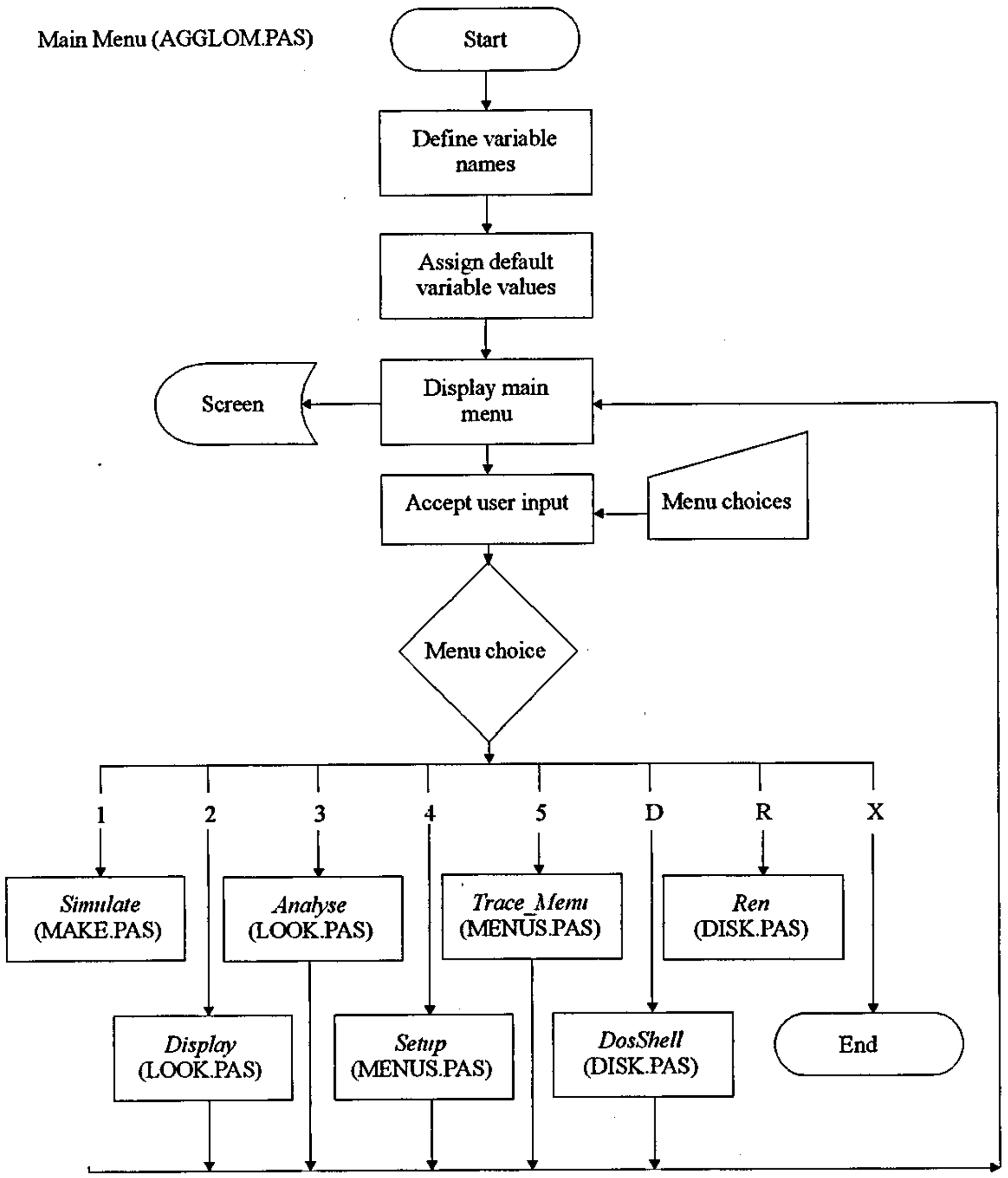


Figure 2.1 - Flow sheet for main menu

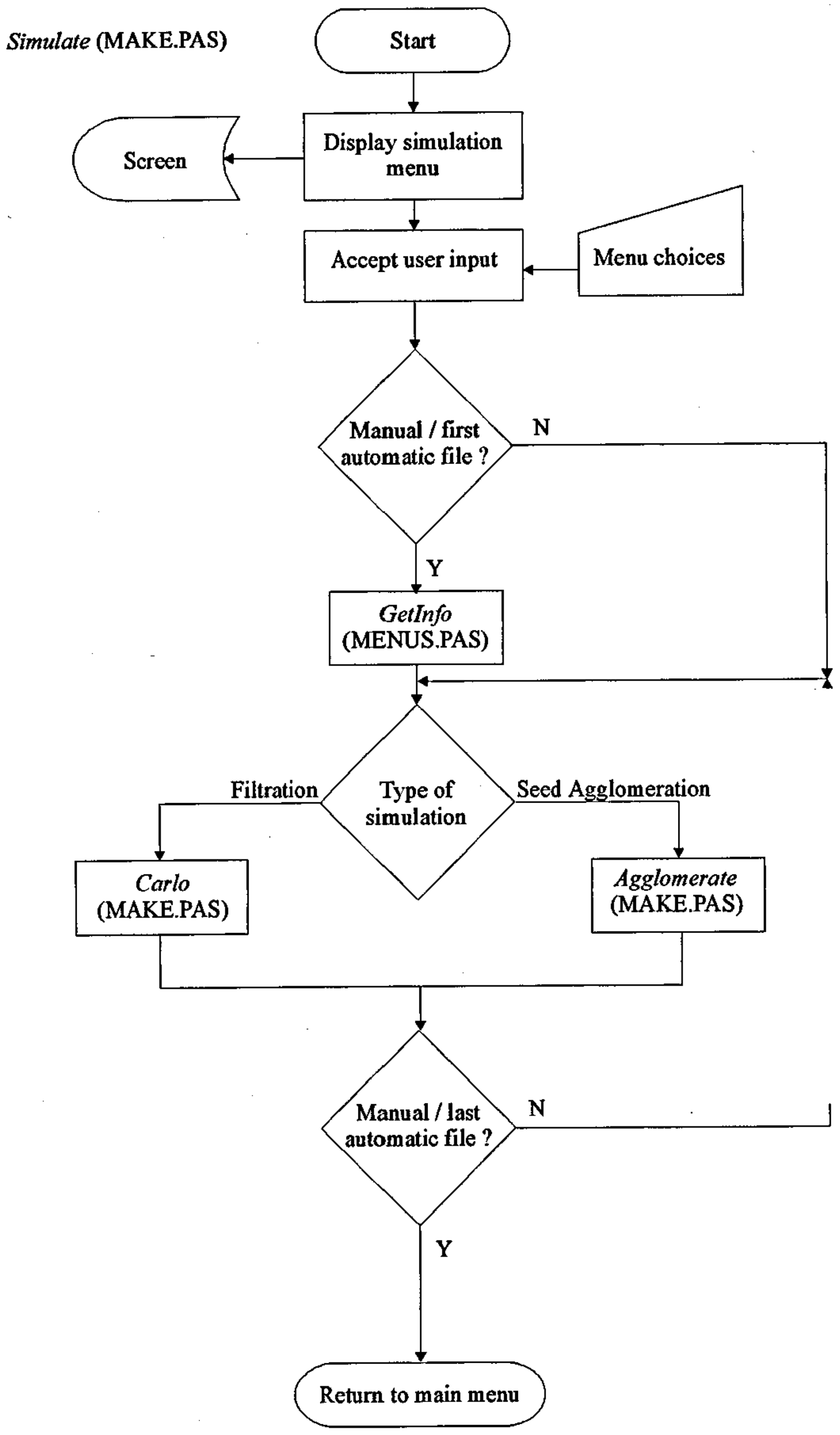


Figure 2.2 - Flow sheet for simulation procedure

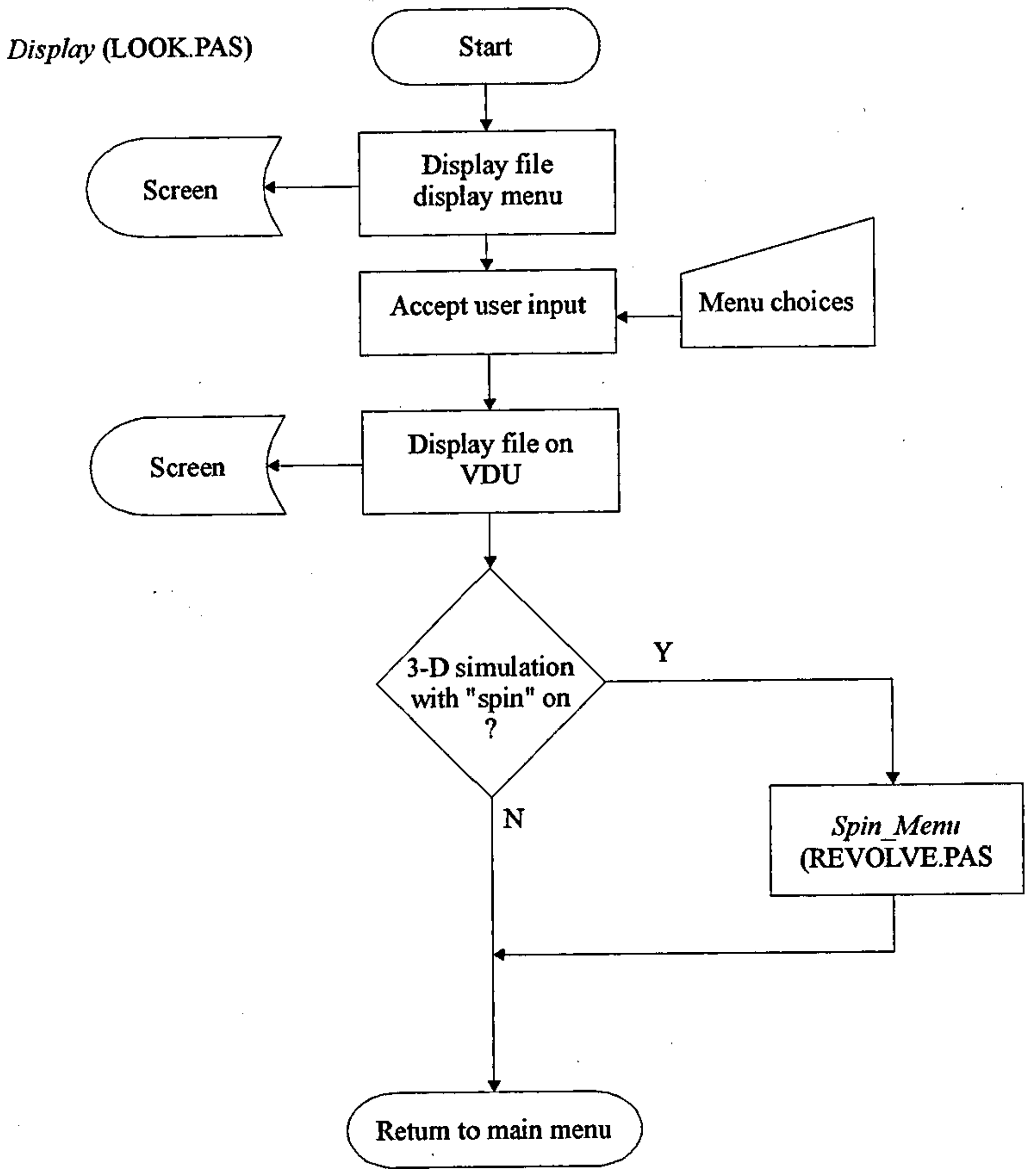


Figure 2.3 - Flow sheet for displaying and rotating agglomerates / filtration simulations

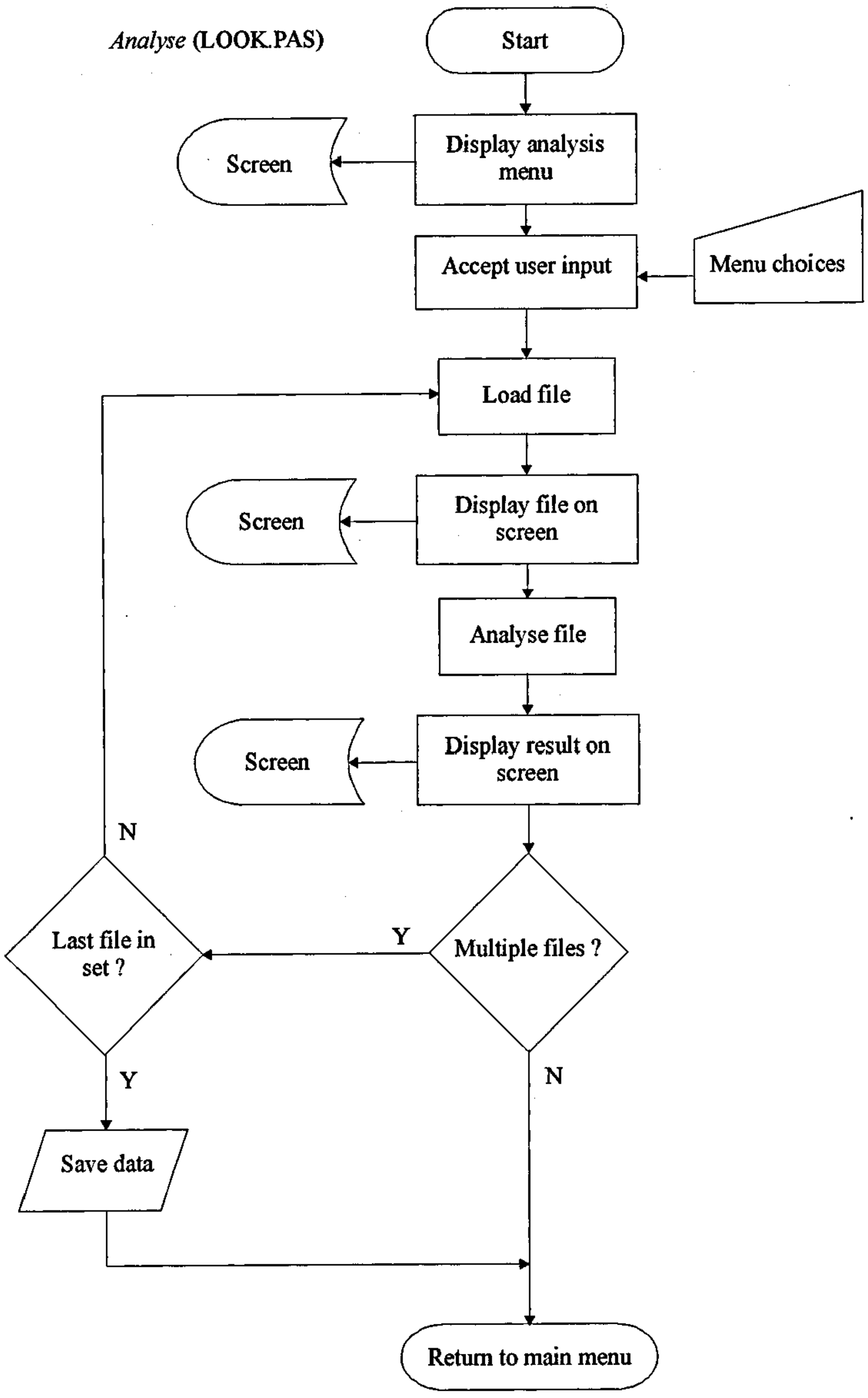


Figure 2.4 - Flow sheet for analysis of structures

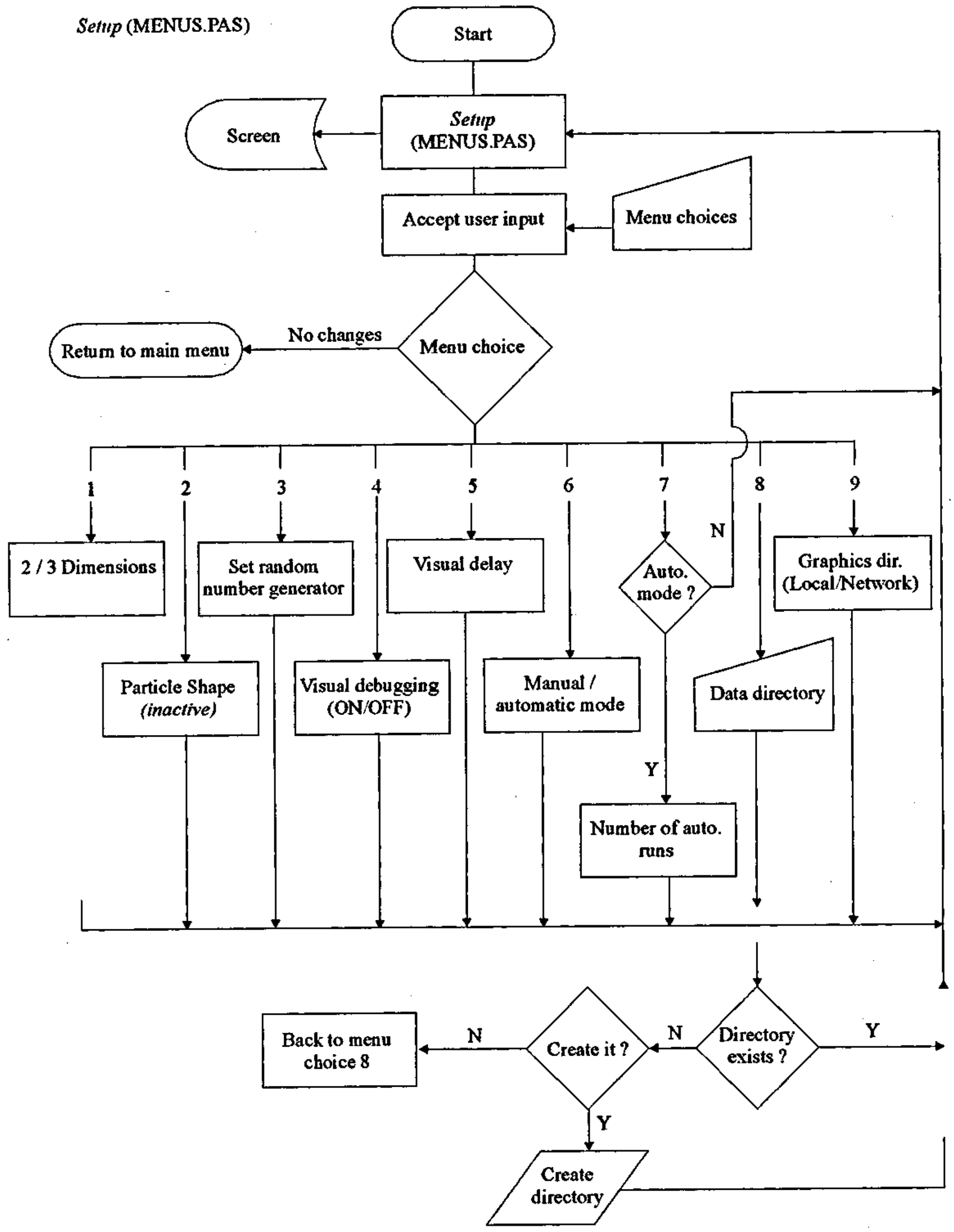


Figure 2.5 - Flow sheet for user settings

2.3 Agglomerate building

Agglomerates were built in two or three dimensions using various mechanisms by attaching particles to either a seed in the centre of a circle (or sphere) or attaching particles to the base of a virtual filter cell. Both types of agglomerate are built using an off-lattice model, which means that the particles in the system are free to move anywhere in a 360° arc around their current position, rather than being restricted to adjacent pixels, as is the case with on-lattice simulations.

All the simulations use the same co-ordinate system - the positive *x* direction is to the right, the positive *y* direction is down the screen (*not* up as with traditional axes) and the positive *z* direction is 'into' the screen. For the *x* and *y* axes, a zero position is automatically defined by the software as the top left hand corner of the screen. The *z* axis has no natural zero position. The zero position for these simulations was defined as an arbitrary point in space from which all measurements were taken. The co-ordinate system is illustrated in Figure 2.6. The co-ordinate system for two dimensions is identical to that for three dimensions, without the *z* axis.

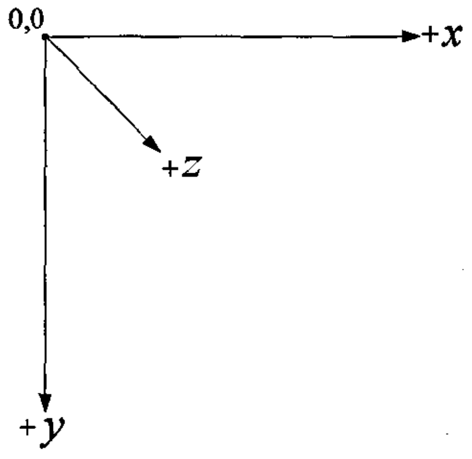


Figure 2.6 - Co-ordinate system for three dimensions

Figure 2.7 shows the difference between on- and off-lattice simulations. For a typical off-lattice simulation, a particle comprising a single pixel has its movement restricted to adjacent horizontal and vertical pixels (i.e. the particle is allowed to travel only in the *x* or *y* direction, one pixel at a time). For an off-lattice simulation there are much less stringent restrictions imposed on the movement of the particle's size, shape or movement.

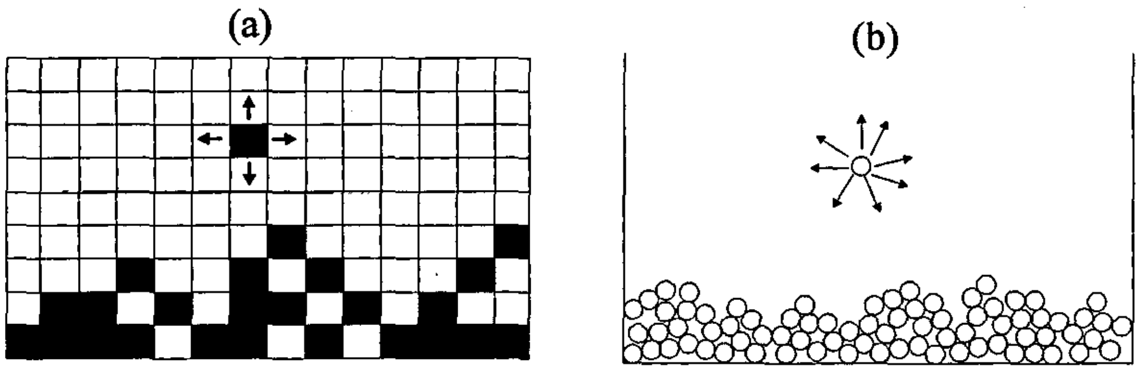


Figure 2.7 - Examples of on-lattice (a) and off-lattice (b) simulations [On lattice example from Giano and Patierno, 1993]

2.3.1 Seed agglomeration

In the case of seed agglomeration, the agglomerates consisted of a pre-determined number of particles (either circular or spherical, depending on the number of dimensions in which the simulations were performed) with a pre-determined size range.

The maximum number of particles any system is capable of storing depends on a number of factors. The operating system, controlling software and program code all affect the amount of data storage available. Each particle requires co-ordinate and size information to be held within the memory of the computer. For two-dimensional simulations, this requires three pieces of information: x and y co-ordinates and size. For three dimensional simulations, the z co-ordinate information must also be stored. In the application discussed here, each of these pieces of information is stored as an integer, occupying a pre-determined amount of space both in the memory of the computer (when the simulation / measurement is being performed) and the disk storage space.

Once the agglomerates had been built, they could be analysed in their respective dimensions by the three different methods discussed in Section 2.4.

Initially, for the seed agglomeration models, a boundary circle or sphere, depending on whether the simulation was being performed in two or three dimensions, was set up on the computer screen. To avoid excessive repetition in the following sections, the circle or sphere will be referred to simply as the circle.

After the boundary circle was drawn, individual particles were assigned random entry and, if required, exit angles (and thus point(s)) around the perimeter of the circle. In the case of two dimensions, an x and y co-ordinate were determined from a one-angle start point, and in the case of three dimensions, x, y and z co-ordinates were determined from a two-angle start point. This was a relatively simple operation as both the radius and centre of the circle were given as constants in the program. An example of the technique used is shown in Figure 2.8. In this case, $x_1 = x + r\sin\alpha$ and $y_1 = y + r\cos\alpha$. The code for defining the entry point of a particle can be found in the *Start* procedure (MOVER.PAS) in Appendix A, whilst the calculation of an exit point (required only for the 'ballistic' and 'mixed' motion models) can be found in the *Ballstep* procedure in the same section.

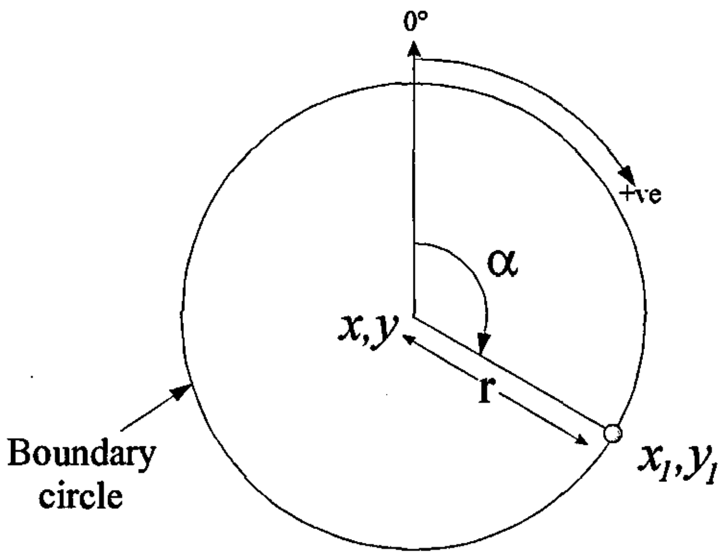


Figure 2.8 - Example of calculation of start-point definition

After the entry and exit (if required) points were defined, the particle moved inside the boundary circle depending on the controlling mechanism(s) used for the particular simulation. The two controlling mechanisms were ballistic and diffusive motion, and varying degrees of each could be defined in any simulation (i.e. from 0% to 100% diffusion in 5% intervals).

2.3.1.1 Ballistic interception

After the entry point had been defined (as described above), ballistic interception required only an exit point to be defined, again from a random angle around the boundary circle. For each particle in the simulation, a trajectory was calculated based on the entry and exit points and the possibility of collision was calculated using the perpendicular distance between the moving particle and the growing agglomerate. This was achieved by checking the distance between the moving particle and each of the particles within the agglomerate. The perpendicular distance is given by Eq. (2.1):

$$h = \frac{2}{a}\sqrt{\left|d(d-a)(d-b)(d-c)\right|} \tag{2.1}$$

where *a, b, c & h* refer to the length of the lines shown in Figure 2.9 and *d* is defined as ½(*a*+*b*+*c*):

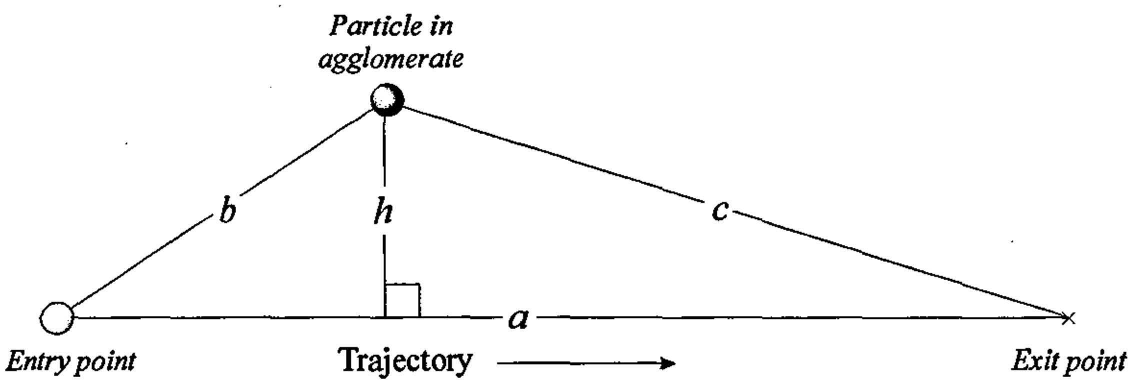


Figure 2.9 - Calculation of perpendicular distance

If the perpendicular distance, *h,* is less than the combined radii of the moving particle and the particle already attached to the agglomerate, then the moving particle will attach to the agglomerate. If, on the other hand, the perpendicular distance is greater than the combined radii of the two particles, then the moving particle will miss the agglomerate. In the latter case, another pair of random entry and exit points are defined, and the same procedure applied to check whether the moving particle will hit, and therefore attach to the agglomerate. In Figure 2.9, the moving particle would obviously miss the particle in the agglomerate, as the perpendicular distance is far greater than the combined radii of the two particles under scrutiny. The calculations for perpendicular distance can be found in the *Test2D* and *Test3D* procedures (MOVER.PAS) in Appendix A.

If a particle was deemed to be on a trajectory that would intercept the agglomerate, the motion of the moving particle would continue in small increments along the line between entry and exit points until interception had occurred. If interception occurred between two of these increments, a fraction of an increment was subtracted from the previous step in order to avoid overlapping any of the particles. This fraction was calculated as the ratio of the difference between the combined radii of the two particles and the current separation distance to the difference between the previous and current separation:

$$\text{Fraction} = \frac{\text{Combined Radii - Current Separation}}{\text{Previous Separation - Current Separation}}$$

The attachment algorithms can be found in the *Attach2D* and *Attach3D* procedures (MOVER.PAS) in Appendix A.

If more than one particle in the agglomerate was in the trajectory of the moving particle, the straight line distance from the moving particle to each of the potential target particle was calculated, with the moving particle hitting the closest target particle.

Once the allotted number of particles were attached to the agglomerate, the structure's information was stored on disk in the form of *x* and *y* co-ordinates (*x*, *y* and *z* in the case of three dimensions) and the size of each particle. This enabled the program to later read the agglomerate structure from disk and perform any necessary analyses.

Figure 2.10 shows two typical agglomerates built using the ballistic interception model. The agglomerate on the left was built in two dimensions, whilst the one on the right was constructed in three dimensions. The apparently unconnected particles in Figure 2.10 and others are caused by the shadows of other particles (generated by the ray tracing calculations) falling over the connecting particles.

The two dimensional structure is shown as a 'screen shot' (direct screen representation) from the application used to create it. The three dimensional agglomerate has been 'rendered' using POV-Ray®, commercially available ray tracing software. This software was used to visualise the agglomerates in three dimensions in order to check the accuracy of the models used. As well as still images such as this, it was also possible to create an animation of the agglomerates, by combining a number of different views around the structure.

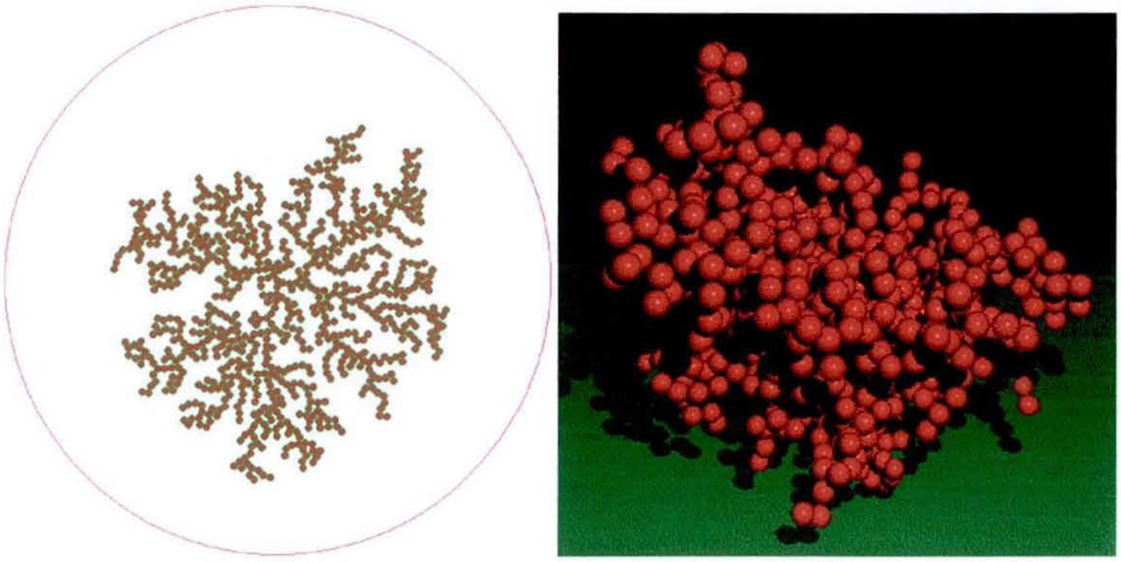


Figure 2.10- Example of 2-D and 3-D ballistic model simulations

2.3.1.2 Diffusion (brownian motion) model

The diffusion model was similar to the ballistic interception model in that a boundary circle was defined and particles released one at a time into the circle. However, it differed in that instead of the particles having a prescribed trajectory, they were allowed to move in a random motion within the boundary circle. If a particle's next move was to take it out of the circle, then a new random direction of movement was calculated that would keep it inside. Due to the nature of the model, the agglomerates built using a diffusional mechanism took longer to construct than those built using the ballistic interception model. In three dimensions, the extra simulation time required the use of an progressively-increasing sphere radius around the growing agglomerate from which the particles were released. The capture mechanism for the diffusion controlled model was similar to the ballistic model where a perpendicular distance determined whether or not the moving particle contacted the growing agglomerate. The difference was that the distance a in Figure 2.9 (i.e. the step movement taken by the moving particle) was equal to the moving particle's radius rather than the distance to the boundary circle.

Examples of agglomerates built using the diffusion model are shown in Figure 2.11

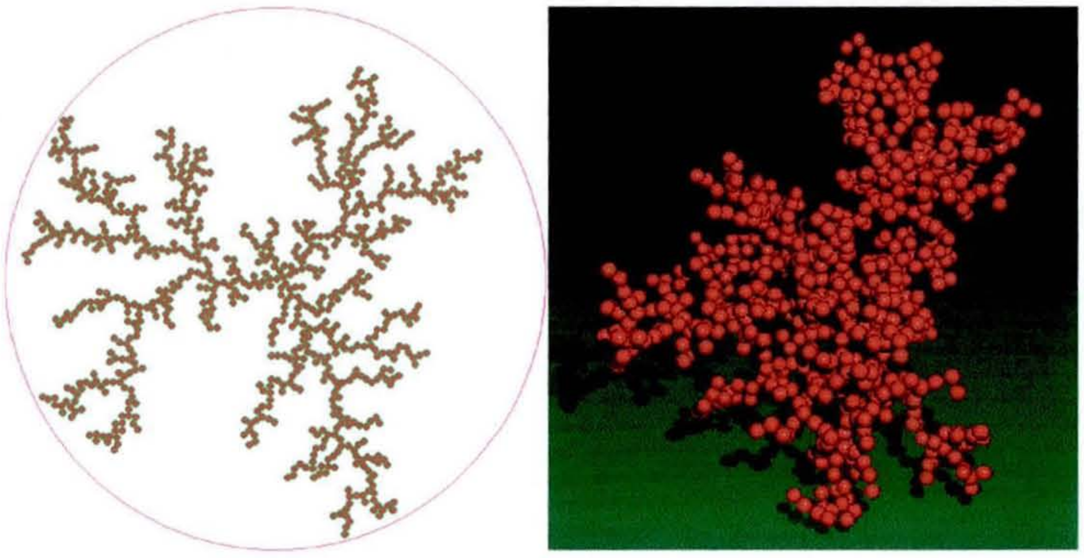


Figure 2.11 - Examples of 2-D and 3-D simulations with 100% diffusion

It can be seen by comparing Figure 2.10 and Figure 2.11 that an increase in the diffusion influence has a pronounced effect on the structure of an agglomerate. The exact effect is quantified later. There are as many particles of the same size distribution in the three dimensional agglomerate on the right of Figure 2.11 as there are in that of Figure 2.10; the particles appear smaller and more numerous due to the variation in structure between the two. The ray tracing software was given a point of view (camera position) to enable it to see the entire agglomerate without either the structure overlapping the edges of the image, or the structure being too 'far away' from the point of view so as to make it difficult to view.

2.3.1.3 Mixed mechanism model

As well as the ballistic and diffusion controlled models, agglomerates were also built using varying degrees of ballistic and diffusive steps. As for the ballistic model, an entry and exit point were defined for a particle and this set the particle's trajectory and original direction. The particle then moved a given number of ballistic steps (in the original direction), followed by a number of diffusive steps (in a random direction). Once again, if the next diffusive step of the particle would take it outside the boundary circle, another direction was calculated to keep the particle inside. If a ballistic step (along the original trajectory) that would require the particle leaving the boundary circle, this was allowed to occur, and a new particle released. Particle capture was determined as before.

The amount of diffusive influence on particle motion could be varied from 0% to 100% in 5% increments for any simulation. The model used these 5% increments to calculate the number of diffusive compared to ballistic steps. For instance, 95% diffusion would give 19 random walk steps and one ballistic step. The algorithm for the mixed mechanism model can be found in the *Agglomerate* procedure (MAKE.PAS) in Appendix A. Figure 2.12 shows two typical agglomerates, again in two and three dimensions, built with 50% diffusion influence (ten ballistic steps followed by 10 diffusion steps).

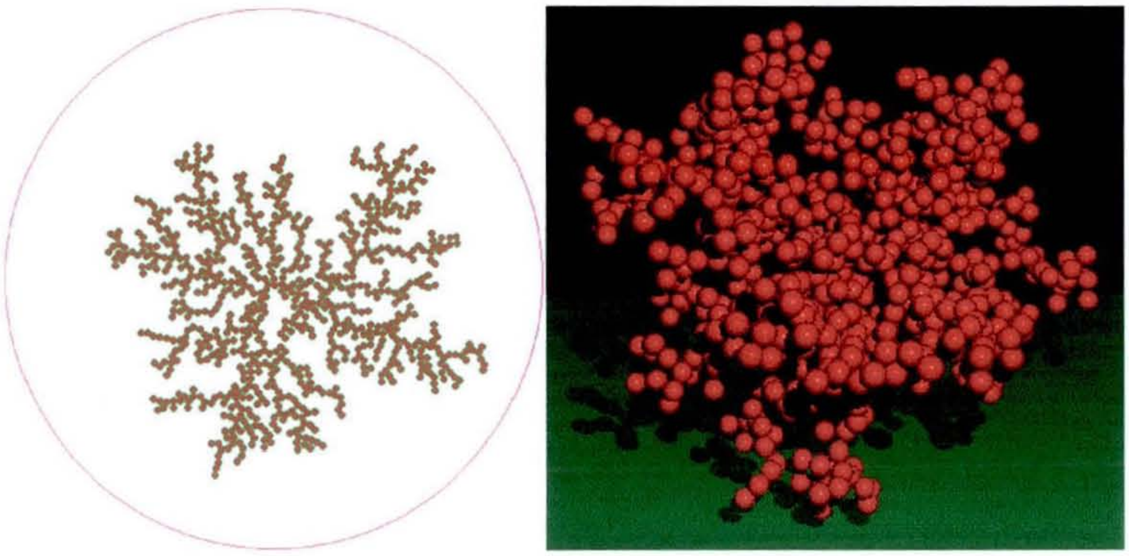


Figure 2.12- Examples of 2-D and 3-D simulations with 50% diffusion

2.3.2 Filtration simulation (Monte Carlo)

Unlike the seed agglomeration, the Monte Carlo simulation used in this work builds a structure up from the surface of a virtual filter cell. These simulations were constructed by imagining individual particles to be released from the top of the cell and allowing them to move in a pre-defined trajectory until they either contact the base of the cell or another particle already forming part of the agglomerate.

These simulations were designed to mimic the basic mechanisms or particles descending through a fluid under the force of gravity until they contacted a surface from where they were unable to descend further. As the filtration experiments carried out as part of this work were dead-end filtration with the cake forming at the bottom of the filter cell, the particles could also be said to move in a downwards direction, as described by these simulations. This work has not attempted to include any other forces as part of the particles' motion, although the way in which the code for the simulation has been written allowed for the inclusion of such additions. Due to the processing power available at the time, this simulation also only releases particles individually through the 'fluid', thus no particle interactions are accounted for whilst the particle is descending. Further additions to the program could include for multiple particles to be released, with the particle interacting as they descend towards the base of the filter cell.

The start point of a particle for a two dimensional simulations was a random value representing a position between the left and right hand sides of the cell (*x* co-ordinate). For the three dimensional simulations, the start point was defined by setting a random value for the *x* and *z* co-ordinates, whilst ensuring that the resulting point lies within the boundaries of the cell.

Particle motion was controlled by two simulation parameters, namely the downward and sticking probabilities. The downward probability governed the particle's motion within the cell. The higher the downward probability the more likely the particle was to move down inside the cell. A downward probability of 100% forced a particle to move vertically downwards inside the cell in a manner akin to the ballistic motion model mentioned previously. A value of 0% allowed a particle complete freedom of movement within the cell, more akin to the diffusive motion model above. The value of downward probability could be set anywhere between these two limits. Eq. (2.2) shows the degrees of freedom of movement of the descending particle:

$$D_F = 360\left(\frac{100 - P_D}{100}\right) \qquad (2.2)$$

where P_D is the downward probability and D_F the freedom of movement in degrees. According to Eq. 2.2, a particle can move anywhere within an arc of *F/2* degrees either side of vertically downwards. The actual direction that a particle moves is given by Eq.(2.3):

$$\theta = \Psi(D_F) - (D_F / 2) \qquad (2.3)$$

where θ is the direction (degrees from vertical) that the particle takes and $\Psi(F)$ is a function used to give a random value between zero and D_F. Although the downward *probability* is fixed at the beginning of each simulation, the direction in which a particle moves is re-calculated for each step, with the constraints previously applied. As with the diffusion controlled seed agglomeration model, if the moving particle's next step would take it outside the boundaries of the cell, a new direction is calculated to retain it within the cell.

The sticking probability governed the particle's behaviour when it made contact with another particle that already formed part of the structure within the cell. As with the downward probability, the sticking probability could have any value between 0% and 100%. A value of 0% infers that a falling particle rolls over any particles in the structure until coming to rest (i.e. the particle does *not* initially stick to the structure). On the other hand, a value of 100% forced a particle to stick immediately on contact with the structure.

Each time a descending particle came into contact with the structure at the bottom of the cell, a random number between zero (0%) and one (100%) would be created. If this random number was less than the sticking probability for the simulation, the particle was deemed to be at rest on the structure. If the random number was greater than the sticking probability, then the descending particle continues to roll over the structure until it came to rest (a position of lowest 'potential energy'). The criteria for a particle coming to rest are described below.

As would be expected, a high sticking probability leads to cakes with a more open structure, whereas cakes built with low sticking probability have denser structures. These structural differences are quantified in Chapter 3.

The rolling mechanism for a descending particle is shown in Figure 2.13. Figure 2.13 (a) shows the falling particle (2) contacting a particle already forming part of the structure at the base of the cell (primary target particle, 1) and rolling to rest on the surface of the virtual filter medium (the base of the cell). Figure 2.13 (b) shows a falling particle (3) contacting a target particle (1) and rolling to rest on top of a second particle (secondary target, 2). Finally, Figure 2.13 (c) shows the falling particle (2) contacting a target particle (1) and rolling to rest against the wall at the side of the filter cell.

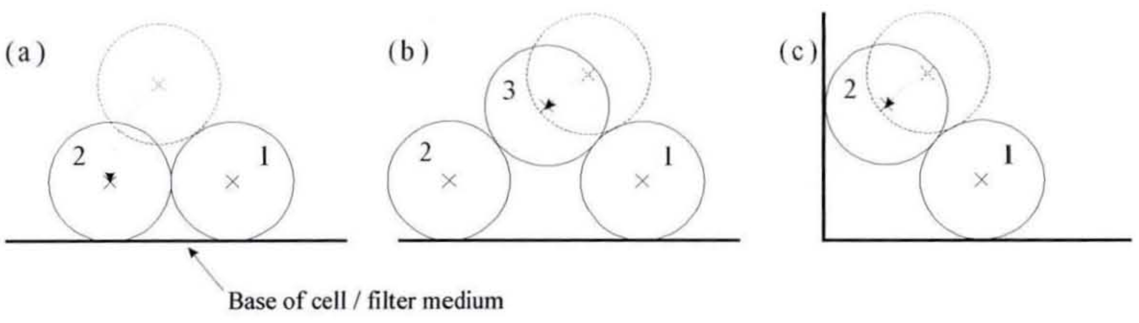


Figure 2.13 - Particles rolling to rest at the base of the 2-D virtual filter cell

The rolling mechanism used a number of calculations to allow the descending particle to move in a way that would be expected for two circles or spheres contacting each other under gravitational forces.

For two dimensions, the attaching particle rolls in a simple arc about the centre of the particle attached to the structure, and a radius equal to the combined radii of the attached and descending particles. Initially, a contact angle was calculated in order to ascertain the direction in which the descending particle would roll. Once this function has been completed, the angle between the two particles was reduced or increased in increments of 0.005 Radians (depending on whether the direction of the roll was anti-clockwise or clockwise) to simulate rolling of the descending particle. With each change in angle, the new position of the rolling particle was re-calculated.

With this small step change in position it was relatively easy to allow the particle to roll until contact was made with either another particle, the base of the cell, the wall of the cell or a combination of the three (see Figure 2.13). The equations for particle roll (both for two and three dimensions) can be found in the ROLLING.PAS unit in Appendix A. Both the x and y co-ordinates at each step in the roll were calculated from the change in angle, and then the distance to each particle in the agglomerate as well as the base and wall. If the distance to another particle (other than the primary target particle) was equal to the combined radii of the two, the rolling particle would either be deemed to be at rest (if its centre lay between the centres of the two particles - see Figure 2.13 (b)) or continue to roll over the secondary target particle (if the descending particles centre was no longer between the two particles). If the descending particle rolled to a position where its centre was on the same horizontal plane (equal y co-ordinates) as the target particle, the descending particle was released into the bulk 'fluid'. In this situation, the descending particle was given a new movement step in a direction away form the target particle (in order for the program not to register the descending particle as a new contact), from where it moved back into the open space surrounding the agglomerate, subject to the previous motion characteristics as defined earlier in the simulation. This mechanism can be found in the *Freefall* procedure (CARLOMOV.PAS) in Appendix A.

Figure 2.14 shows two examples of cakes built using the Monte Carlo simulation in two dimensions. The structure on the left was built with 0% sticking probability, whilst the right hand image was built with a sticking probability of 100%. Both agglomerates contain 1000 particles and were constructed with the same downward probability of 50%. The difference in structure between the two simulations can clearly be seen.

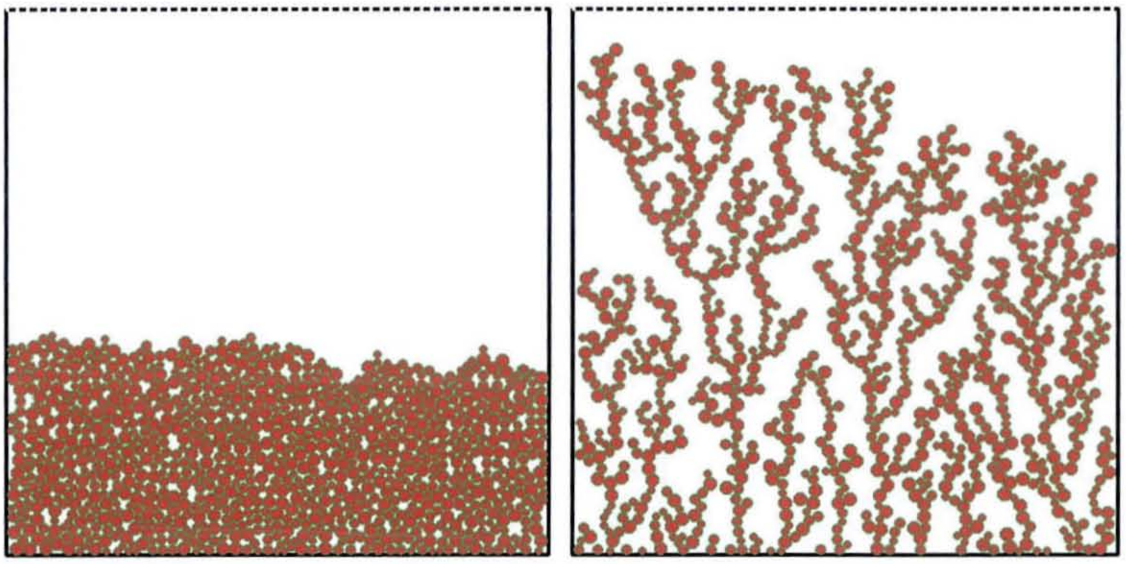


Figure 2.14 – Filtration model simulations built with sticking probabilities of 0% (left) and 100% (right) and 50% downward probability, particles randomly distributed between three and five pixels radius

For a particle to roll in three dimensions, it was not just a case of a simple arc between two circles. The particles in question were considered to be spheres and as such, it was found that the descending particle would roll in an arc similar to that for two dimensions, but with the added dimension of depth. Thus, the particle was able to roll 'forwards' or 'backwards' (positive or negative movement in the z direction) in addition to right and left (positive or negative movement in the x direction). Once the descending particle had contacted more than one target particle, it was only considered to be at rest if all three centres were in a straight line (no impetus for movement). If, however, this were not the case, then the descending particle would be allowed to roll in a plane perpendicular to a line connecting the two target particles. Whilst difficult to visualise in two dimensions, Figure 2.15 shows how a descending particle rolls across two target particles. Again, if the descending particle were to move past the horizontal position relative to the target particles, it would move into a 'free-fall' state, with a three dimensional direction vector being assigned in order to move the descending particle away from the target particles, followed by the re-application of the original motion characteristics (downward and sticking probabilities). The descending particle was deemed to be at rest if it contacted the base of the cell. As well as this case, an 'infinitely sticky' wall was used, which meant that if the descending particle contacted the wall whilst rolling over a target particle, it would stick. The third case for a particle to be deemed at rest was if the descending particle contacted a third target particle whilst rolling over two target particles.

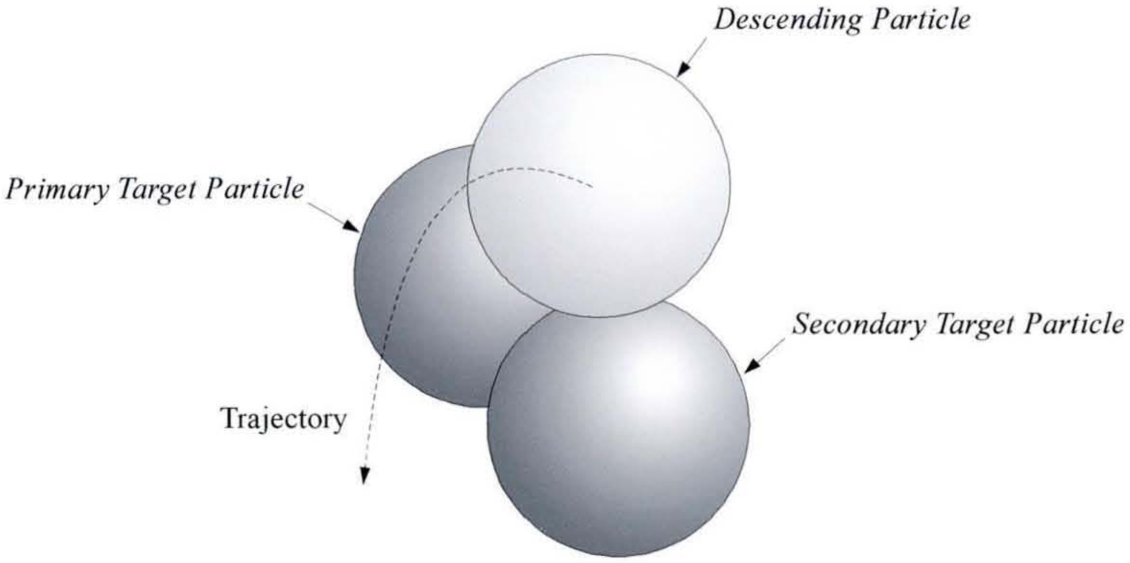


Figure 2.15 - Descending particle rolling across primary and secondary target particles

Figure 2.16 shows a typical structure built using the three dimensional model with particles randomly distributed between eight and ten pixels radius. The circle at the base of the structure represents the filter medium at the base of the virtual filter cell. This structure contains 1000 particles and was built using 100% downward probability and

Figure 2.16 to that in Figure 2.17 (built with 100% downward probability, but 100% sticking probability) that the parameters controlling the formation process affect the final appearance of the structure. Both structures contain 1000 particles, but the example in Figure 2.17 appears larger due to the more open structure. It can be seen from Figure 2.17 that it is possible to create structures that bear little or no resemblance to naturally occurring systems (with particles seemingly hanging in mid air with no apparent means of support). It is therefore necessary to balance the two system parameters used in order to generate a structure that does resemble a filtration system.

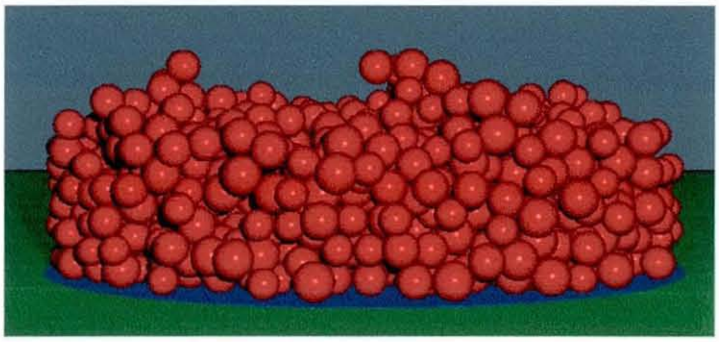


Figure 2.16 – Typical 3-D filtration simulation constructed with downward probability = 100% and sticking probability = 0%

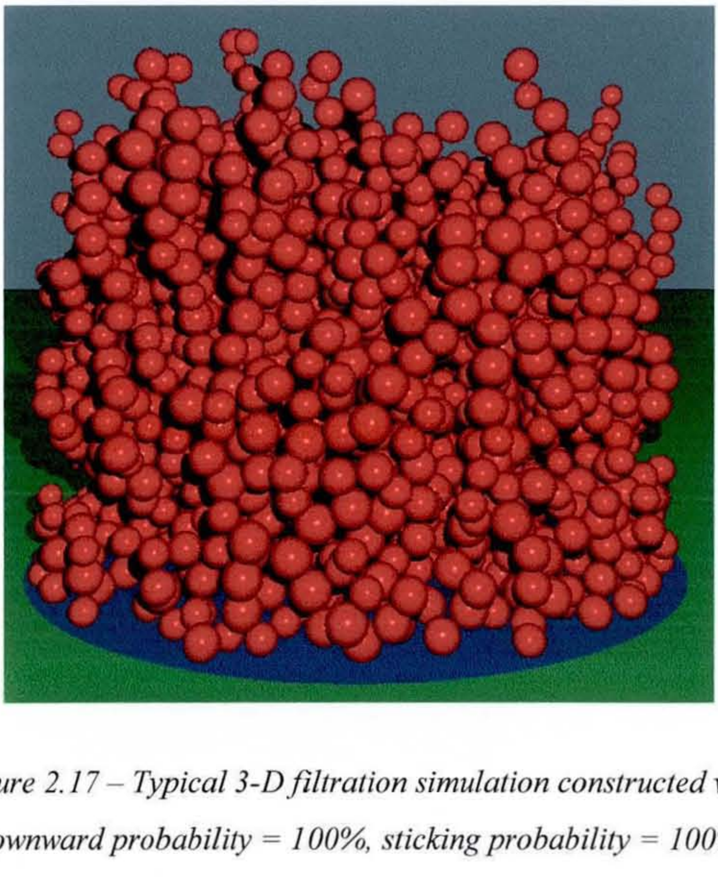


Figure 2.17 – Typical 3-D filtration simulation constructed with downward probability = 100%, sticking probability = 100%

2.4 Fractal and physical analysis of structures

As well as the ability to analyse the properties of structures created within the software, the program written by the author could also measure the fractal and physical characteristics of a range of different image and agglomerate types. This section describes both the measurement techniques available and the types of data to which they could be applied.

2.4.1 Structured walk analysis

The initial analysis type that was used on the agglomerates (both seed agglomeration and filtration simulation models) was the structured walk method. As explained previously, this method gives a value for the surface roughness of a structure.

In the case of the seed agglomeration models, the structure was first drawn on screen in one colour (blue) in order that the perimeter could be found against the background of the screen. After this task, the program then calculated the centre of gravity of the agglomerate. The equation for the centre of gravity is:

$$\text{Centre of gravity } (x) = \frac{\text{Sum of moments } (x)}{\text{Sum of area}} \tag{2.4}$$

where the sum of the moments (in the x axis) is calculated by Eq.(2.5):

$$\text{Sum of moments } (x) = \sum_{i=0}^{n} \Delta x . \pi r_i^2 \tag{2.5}$$

where Δx is the distance in the x direction between the centre of the screen and the x co-ordinate of particle i for n particles. By applying the same equation in the y axis (also the z axis for three dimensions) it was possible to ascertain the co-ordinates for the centre of gravity of a structure. The next stage was to determine the position of the particle furthest away from the centre of gravity. This was the particle used as a starting point for the structured walk measurement. By splitting the agglomerate into four quadrants - top right, bottom right, bottom left and top left, and ascertaining in which quadrant the outermost particle lay, a starting angle for the structured walk could be defined. Use of the quadrant procedure ensured that the first attempt at a step started from an angle in a direction normal to the mass of the agglomerate. Figure 2.18 shows a typical agglomerate split into the four quadrants with the outermost particle of the agglomerate marked.

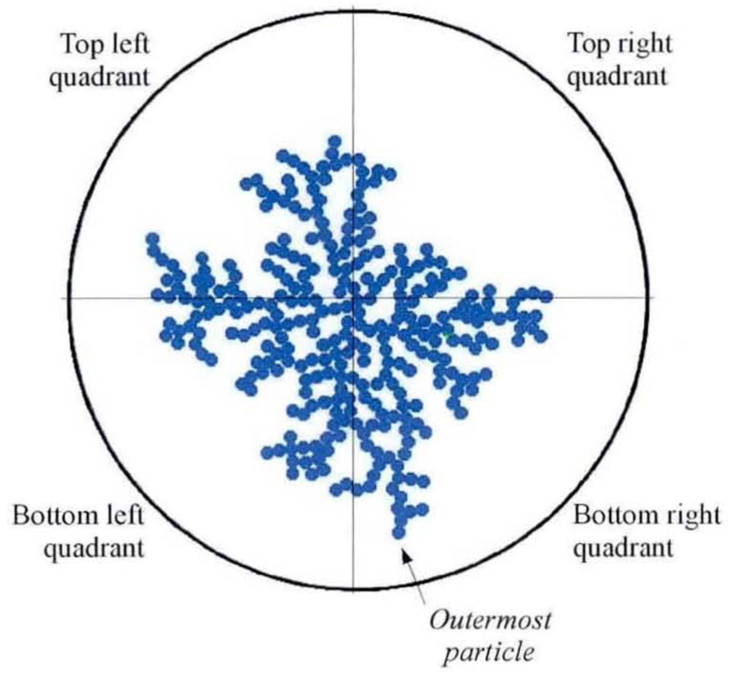


Figure 2.18 - Typical agglomerate showing quadrants and outermost particle

The algorithm for the calculation of centre of gravity can be found in the *CofG* function (FUNCS.PAS), whilst the quadrant containing the outermost particle is determined in the *AggWalk* procedure (SWALK.PAS), both in Appendix A.

Before the structured walk could be applied to an agglomerate, the steplength was defined as a fraction of the maximum Feret diameter of the structure. This in turn required the calculation of the Feret diameter. The Feret diameter was calculated by measuring the maximum separation of two parallel lines brought in towards the shape from a circle with a radius slightly larger than the distance between the outermost particle and the centre of gravity. The lines in question were tangential to the circle and were moved in towards the agglomerate from all 360° around the structure. Figure 2.19 shows the technique in a graphical form.

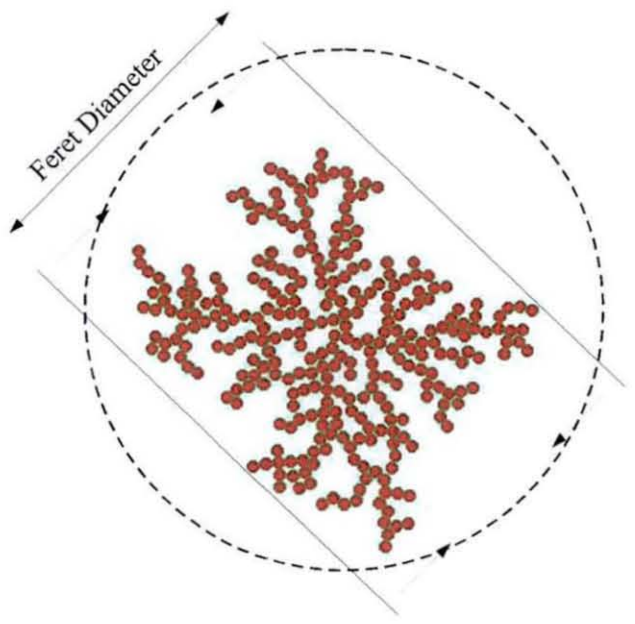


Figure 2.19 - Graphical representation of calculation of Feret diameter for a two dimensional agglomerate

The maximum Feret diameter was given by the maximum separation of the parallel lines when in contact with the agglomerate, whilst the minimum Feret diameter was given by the minimum separation of the same pair of lines. The calculation of the Feret diameter for seed agglomerates can be found in the *AggFerets* procedure (SWALK.PAS) in Appendix A. Once the Feret diameter had been determined, a steplength was calculated as a fraction of the Feret diameter (generally between 0.08 and 0.32 times the Feret diameter).

After the starting point, starting angle and steplength had been calculated, the walk was allowed to proceed around the perimeter of the structure. Virtual 'dividers' moved in a clockwise direction around the perimeter, using the outswing method, as described in Chapter 1. The structured walk was performed by plotting the position of the end point of an arc with a pre-defined centre and radius (the structured walk steplength). Checks were initially made to ensure that the start of the step was in the empty space around the outside of the agglomerate. If this was not the case, the start point of the step was moved backwards (anti-clockwise) until the start point entered space. Once a satisfactory start point had been found, the 'dividers' would continue moving in a clockwise direction until the perimeter of the agglomerate was found. Both the start point and perimeter checks were carried out by inspecting the pixel colour at the end of the formed arc - a black pixel indicated open space while a blue pixel indicated part of the structure. A colour change from blue to black or vice versa indicated the perimeter of the agglomerate. Both the start point calibration and actual steps were moved in 1° intervals. The walk continued in this way until either of two criteria were satisfied:

- Distance from end of steplength to original start-point was less than the defined steplength. In this case, if more than four steps were carried out, the walk was deemed a success and the perimeter at that steplength stored. Conversely when less than four steps were performed, there is a high possibility that the walk started on the edge of one particular feature of the agglomerate and simply walked around that feature before returning to the start point. In this case, the walk was abandoned, and the perimeter not included in the calculation of perimeter fractal dimension.
- The total perimeter of the structure was more than five times the maximum Feret diameter. In this case (only applied when using multiple steplengths for the calculation of the fractal dimension), the 'dividers' had usually been trapped inside a feature of the structure. Again, if this error occurred, the walk was abandoned and the perimeter discounted from the calculation of fractal dimension.

The algorithm for the measurement of the perimeter fractal dimension using the structured walk method can be found in the *Steps* procedure (SWALK.PAS) in Appendix A. It should be noted that the *Steps* procedure is also used to measure the fractal dimension of the perimeter of particles scanned and digitised from an SEM.

A typical structure walk (at 0.10 times the Feret diameter) is shown in Figure 2.20. The agglomerate has been lightened in colour to emphasise the perimeter line. Figure 2.20 also shows the maximum and minimum Feret diameters of the agglomerate and the outermost particle. For the given example, the maximum Feret diameter is represented by the line that passes through the outermost particle of the agglomerate.

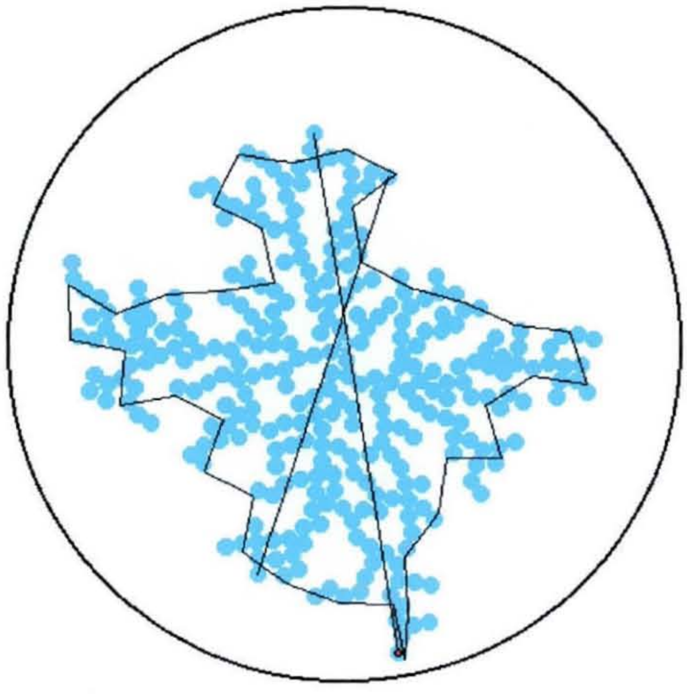


Figure 2.20 - Example of structured walk applied to a 2-D agglomerate

Once the perimeter was measured at all the desired steplengths, the data was passed to another two procedures (*GraphIt* and *Regression* - both in FUNCS.PAS). The *Regression* procedure was used to calculate a least squares fit for the data, including a given confidence interval, and confidence of linearity. Data files containing the regression data from a statistical analysis booklet [White et al. 1994] were used to provide the figures required to produce the confidence data. The *GraphIt* procedure was then used to display the data on screen, including the best fit line and the final result. The *GraphIt* procedure could handle a wide variety of data, depending on the type of information passed to it.

The fractal dimensions of the surface of filtration simulations were measured in much the same way as for the seed agglomeration models, with the exception of two techniques:

- The maximum Feret diameter was taken as the width of the filter cell divided by π. In this way, the width could be better equated to the diameter of a circle.
- The technique for finding the start point of the structured walk varied in that to find the start point, the highest particle within one steplength of the cell wall was found, and this deemed to be the level at which to begin the walk.

- The walk is performed in two directions - right to left (clockwise outswing method) and left to right (anti-clockwise outswing method) – and an average of the two results taken.

The data from the structured walk measurements were again stored and passed on to the *Regression* and *GraphIt* routines.

The perimeter fractal dimensions for 3-D simulations (both filtration simulations and seed agglomeration models) were obtained by measuring the fractal dimension of the perimeters of 2-D projections of the structures at varying orientations. It was possible within the program to rotate the structures in steps of one degree and save the new orientation for further analysis. However, due to the way in which the program stored particle information (with the particle co-ordinates held as integer-type data), rotation of this nature tended to disrupt the structures with the recalculation of particle positions. A more stable method of rotating the structures for analysis of their projections was to rotate the structures through 90° along the *x*, *y* and *z* axes. This could be effectively achieved by swapping the values of one set of co-ordinates with another. For example, to rotate a structure through 90° around the *y* (vertical) axis required the *x* co-ordinates and *z* co-ordinates of the structure to be swapped. In this way, no trigonometric mathematics were performed on the structures.

The interstitial spaces within two dimensional filtration simulation structures were also characterised using the structured walk technique. The first step of the procedure was to isolate the individual pore under scrutiny. This was achieved by moving a cross-hair cursor around the screen until it rested inside the desired pore space. At this point, the pore space was filled with a different graphical colour from the main body of the agglomerate and thence isolated by means of removing the remaining structure (including pore spaces). The isolation algorithm can be found in the *Isolate* and *Instruct* procedures (SCREENIN.PAS) in Appendix A. Once the pore was isolated, its pixel data was stored in a separate file to enable later retrieval. The main difference between measuring the surface fractal dimension of a simulated structure and a pore space was that the pore space data is stored on a pixel-by-pixel basis, rather than having larger sub-units, such as particles.

The measurement of the fractal dimensions of the perimeters of digitised images was identical to that of the pore spaces, as the SEM information was also stored as pixels, rather than particle units. Due to the size of the SEM images, the measurement of the Feret diameters was slower (calculation pixel by pixel), but with no difference in the manner of final measurement of fractal dimension.

2.4.2 Enclosing circle analysis

The enclosing circle analysis technique was used primarily to give a mass fractal dimension for the 2-D and 3-D seed agglomeration models. Although the technique was slightly modified for the 3-D models by using an enclosing sphere rather than a circle, it was essentially the same. Again the technique is described using the term 'circle' to encompass both two and three dimensional analysis.

Initially, the centre of gravity of the structure was found using the same method as for the structured walk technique. The furthest particle was again used as a reference point for the analysis, this time to define the maximum diameter of the enclosing circles. The number of data points required for the analysis was given as a user input, and thus the size of each circle calculated (simply the maximum circle size divided by the number of data points required). Any particle with its mass wholly within the enclosing circle would have its mass added to the total. The algorithms for both the two and three dimensional analyses can be found in the *Enclose* procedure (ENCLOSING.PAS) in Appendix A. As described in Chapter 1, the data sent to the *Regression* and *GraphIt* procedures was the size of the circle employed and the mass contained therein.

The enclosing circle technique was also used to analyse the pore spaces within the two dimensional filtration simulation models. The difference between the seed agglomeration and pore space techniques was that, again, the data for pore spaces was held on a pixel-by-pixel level rather than particulate information.

When applied to the individual pore spaces, the enclosing circle fractal dimension was used to describe the way in which mass was distributed around the centre of gravity, rather than the density distribution of the structure, as was the case for the seed agglomeration models. The right hand side of Figure 2.21 shows a typical pore space being measured using the enclosing circle technique. The circular indentations in the structure are the particles surrounding the pore. The centre of the circles represents the centre of gravity of the object.

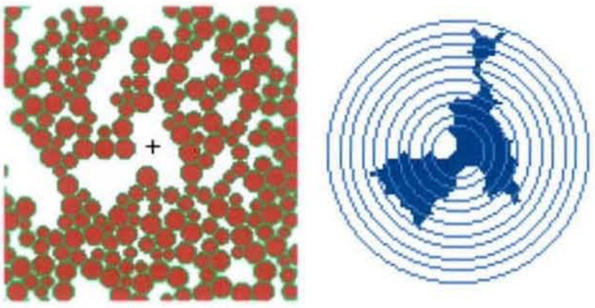


Figure 2.21 – Individual pore space, left (indicated by the small cross in the centre of the object) characterised using enclosing circle technique, right

2.4.3 Radius of gyration technique

The radius of gyration technique was used solely to classify the seed agglomeration models (both in two and three dimensions). This was due to the fact that for the fractal dimension to be calculated via the radius of gyration, it was necessary to know the order in which the particles had attached to the agglomerate and for the structure to have grown outwards from a central seed particle.

The radius of gyration technique (defined in Chapter 1) required the measurement of the moments of inertia and areas of the particles involved. In order to ascertain the number of particles included for each calculation step, the user was again prompted for the number of data points required. The total number of particles in the agglomerate under scrutiny was then divided by the number of data points to give the number of particles in each measurement. Once again, the data were passed to the *Regression* and *GraphIt* procedures for analysis and display. The algorithm for the measurement of the fractal dimension using the radius of gyration can be found in the procedures *RadGyr* (GYRATION.PAS) and *RofG* (FUNCS.PAS). The first of the two procedures (*RadGyr*) is the main calling routine, which relies upon *RofG* to calculate the radius of gyration of each group of particles.

2.4.4 Porosity measurement

The physical rather than fractal measurement used in this work (on all but the images of individual particles) was the measure of porosity. The measure of porosity was defined as the fraction of void space within a structure or part of a structure.

Porosities for the two and three dimensional seed agglomerates required the calculation of the centre of gravity and identification of the outermost particle in order to determine the size of the agglomerate in question. The porosity of the structure was then measured at increasing radial distances between the centre of the agglomerate and the outermost particle. The number of points of measurement was again set by the user. As with the enclosing circle technique, the porosity calculations for the seed agglomeration models only included particles wholly within the circle or sphere being used for the measurement. The routine for measuring the porosity of seed agglomerates is the *Pores* procedure (POROSITY.PAS) found in Appendix A.

The porosity measurement of structures built using the filtration simulation models (in both two and three dimensions) used a similar technique to that for ascertaining the porosity of agglomerates built using the seed agglomeration models in that the porosity was measured using the area or mass of particles within a given boundary. It varied, however, in that more types of measurement were possible and that fractions of particles were taken into account when measurement was being performed.

The inclusion of the fractions of particles into the porosity measurement for filtration simulations was a later addition to the program structure. The inclusion of similar code into the original porosity measurement algorithms (for seed agglomeration) would have required disproportionately more time than the extra results it would have achieved.

Figure 2.22 shows the method used for calculating the partial area of a particle. The algorithms for measuring the porosity of the filtration simulations can be found in the *FiltPores* procedure (POROSITY.PAS) in Appendix A.

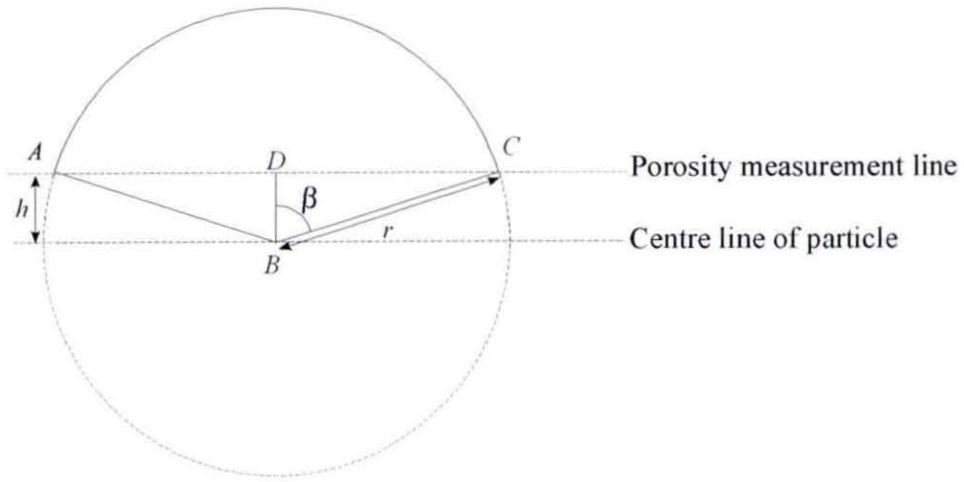


Figure 2.22 - Calculation of partial area for a two-dimensional particle

The area of the *sector ABC* is given by:

$$\frac{2\beta}{2\pi}\pi r^2 = \beta r^2 \tag{2.6}$$

where β is the angle subtended by the measurement line in radians and r the radius of the particle.

The area of the triangle *ABC* is given by:

$$\tfrac{1}{2} AC\, h = DC\, h \tag{2.7}$$

where the length *DC* is calculated as:

$$DC = \sqrt{r^2 - h^2} \tag{2.8}$$

and the area of triangle *ABC* is given by:

$$h\sqrt{r^2 - h^2} \tag{2.9}$$

Hence, the area of the *segment* subtended by the chord *AC* is calculated as:

$$\beta r^2 - h\sqrt{r^2 - h^2} \tag{2.10}$$

Eq. (2.10) can be applied whether the top, bottom or both top and bottom of the particle are cut off by the porosity measurement line.

Three types of porosity were measured for the structures built using the filtration simulation model - two vertical measures and one horizontal measure. The first vertical profile to be employed was the cumulative profile, in which the porosity for each level of the agglomerate was given as the average overall porosity up to the height at which the measurement was taken. Figure 2.23 illustrates the porosity obtained from the first four measurements up the height of a two dimensional filtration simulation.

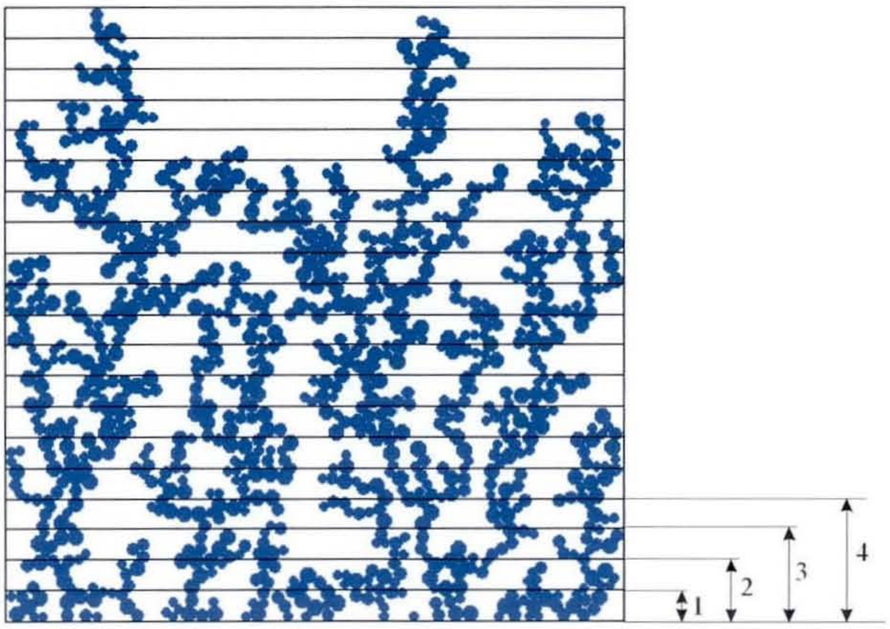


Figure 2.23 - Cumulative vertical porosity measurement

The second of the two vertical porosity measures was the porosity profile. The value of porosity obtained for each level of the agglomerate from this measure was the porosity only of the section between the current height and the height of the last measurement. The vertical profile for the first four sections of the same structure as Figure 2.23 is illustrated in Figure 2.24.

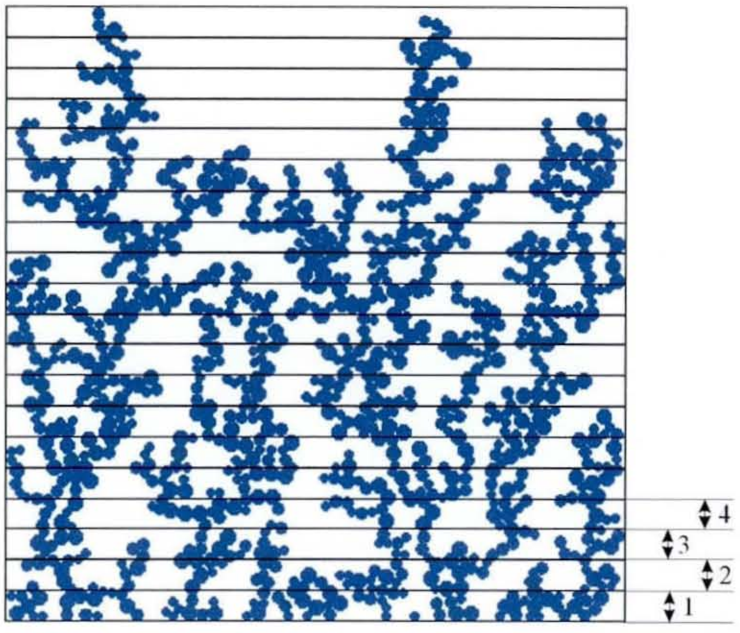


Figure 2.24 - Vertical profile porosity measurement

The third type of porosity measurement carried out was a horizontal profile. The horizontal profile allowed characteristics such as wall effect to be examined in the structure under scrutiny. In the case of the horizontal measurement of porosity, the cake was sliced from right to left, rather than from bottom to top, and the porosity of each section measured. Figure 2.25 demonstrates the horizontal porosity profile as applied to the same structure as Figure 2.23 and Figure 2.24. It should be noted that the sections for the horizontal profile are taken to the highest particle in the particular section being measured, so that a true value for the porosity of the slice can be obtained. This was to prevent including any space above at the top of the slice, which would have been the case had the highest particle in the agglomerate been used as the height for all the slices. As with the vertical porosity measurements, fractions of particles were taken into account, using the same equations to calculate the partial areas. The only difference between the two calculations being that the left and/or right hand sides of the particle were removed by the porosity measurement line, rather than the top and/or bottom.

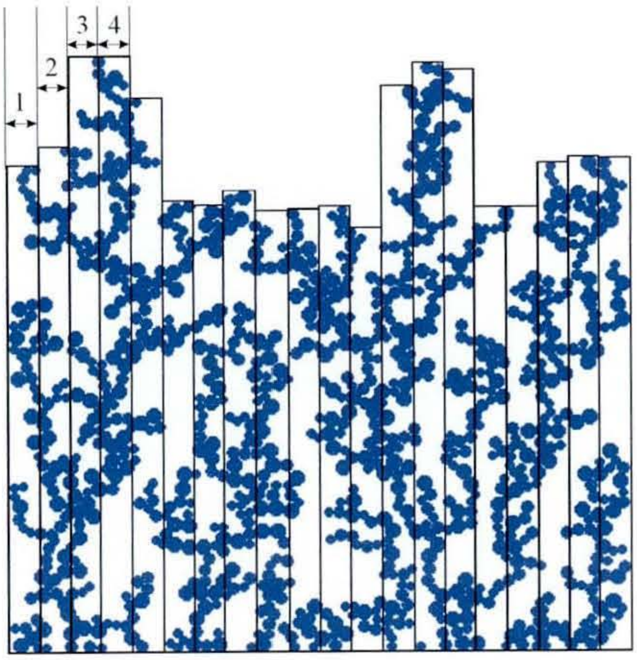


Figure 2.25 - Horizontal porosity profile measurement

The porosity measurements for three dimensional agglomerates were performed in much the same way as for those created in two dimensions. The exception (apart from using volume instead of area to measure the porosity) was that the horizontal profile was obtained using 'cylinders' of increasing size, from the centre of the agglomerate under scrutiny to the boundary of the cell. Figure 2.26 shows a plan view of a typical three dimensional agglomerate with the 'cylinders' used to measure the horizontal porosity profile overlaid. The entire height of the structure (as measured) is shown here, rather than measuring the two dimensional porosity of any particular vertical slice.

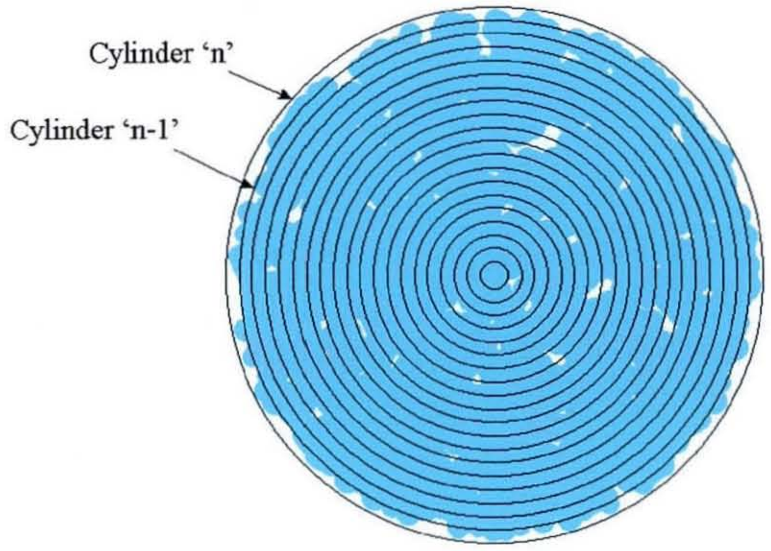


Figure 2.26 - Top view of 3-D structure showing measurement of horizontal porosity profile

The equations for calculating the fraction of particles cut off by the porosity measurement line for the vertical profiles were similar to those for the two dimensional analysis, applied in three dimensions (volume rather than area). Figure 2.27 shows an elevation view of a three dimensional particle with an intersecting porosity measurement plane passing through it (*AC*). This representation is in fact identical to Figure 2.22, repeated in order to provide a reference for the symbols used in the following equations.

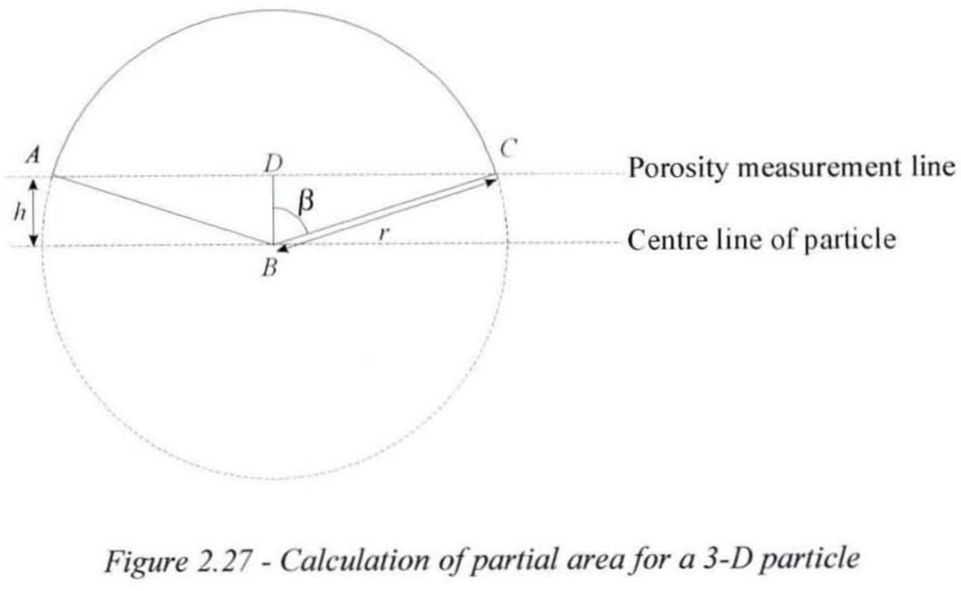


Figure 2.27 - Calculation of partial area for a 3-D particle

The volume of the *section ABC* is given by:

$$\frac{2\beta r^3}{3} \tag{2.11}$$

and the volume of the *cone ABC* is equal to:

$$\frac{\pi r^2}{3}h \tag{2.12}$$

Thus, the volume of the *segment* subtended by the plane *AC* is given by:

$$\frac{2\beta r^3}{3} - \frac{\pi r^2}{3}h = \frac{2\beta r^3 - \pi r^2}{3} \tag{2.13}$$

The calculation of the fraction of particles cut off during the measurement of the horizontal porosity profiles for three dimensional agglomerates was more complicated than that for the vertical profiles. This is due to the fact that the particles are intersected by a curved surface, rather than a straight line. Figure 2.28 shows a typical particle intersected by a curved line (the 'cylinder' used to measure porosity).

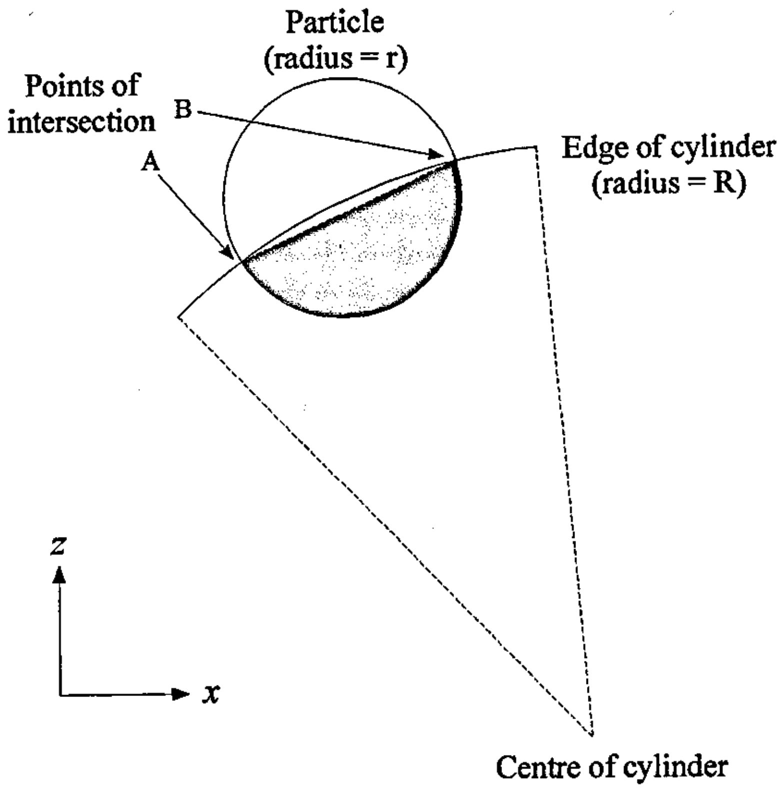


Figure 2.28 - Calculation of fraction of particle using a curved measurement line

To calculate the area of the fraction either inside or outside the measuring cylinder, it was necessary to first find the points of intersection between the particle and cylinder. This was achieved by solving a pair of simultaneous equations (i.e. the equations of the two circles).

Equation for cylinder, centre x_1, z_1, radius R :

$$(x-x_1)^2+(z-z_1)^2=R^2 \tag{2.14}$$

Equation for particle, centre x_2, z_2, radius r :

$$(x-x_2)^2+(z-z_2)^2=r^2 \tag{2.15}$$

where x and z are the co-ordinates of any points on the circumference of a circle using the co-ordinate system explained earlier in this chapter.

Expanding Eqs. (2.14) and (2.15) gives:

$$x^2-2xx_1+x_1^2+z^2-2zz_1+z_1^2=R^2 \tag{2.16}$$

and

$$x^2-2xx_2+x_2^2+z^2-2zz_2+z_2^2=r^2 \tag{2.17}$$

or

$$x^2+x_1^2+z^2+z_1^2=R^2+2xx_1+2zz_1 \tag{2.18}$$

and

$$x^2+x_2^2+z^2+z_2^2=r^2+2xx_2+2zz_2 \tag{2.19}$$

Subtracting Eq. (2.19) from Eq. (2.18)gives:

$$(x_1^2-x_2^2)+(z_1^2-z_2^2)=R^2-r^2+2x(x_1-x_2)+2z(z_1-z_2) \tag{2.20}$$

rearranging gives:

$$2x(x_1-x_2)=(x_1^2-x_2^2)+(z_1^2-z_2^2)+(r^2-R^2)-2z(z_1-z_2) \tag{2.21}$$

now, making x the subject,

$$x = \frac{(x_1^2 - x_2^2) + (z_1^2 - z_2^2) + (r^2 - R^2) - 2z(z_1 - z_2)}{2(x_1 - x_2)} \tag{2.22}$$

Now, let $A = (x_1^2 - x_2^2) + (z_1^2 - z_2^2) + (r^2 - R^2)$, then:

$$x = \frac{A - 2z(z_1 - z_2)}{2(x_1 - x_2)} \tag{2.23}$$

Substituting Eq.(2.23) into Eq. (2.16) gives:

$$\left[\frac{A - 2z(z_1 - z_2)}{2(x_1 - x_2)}\right]^2 - 2\left[\frac{A - 2z(z_1 - z_2)}{2(x_1 - x_2)}\right]x_1 + x_1^2 + z^2 - 2zz_1 + z_1^2 - R^2 = 0 \tag{2.24}$$

Now, expanding Eq.(2.24) gives:

$$\begin{gathered}\frac{A^2 - 4Az(z_1 - z_2) + 4z^2(z_1 - z_2)^2}{4(x_1 - x_2)^2} - \frac{2Ax_1}{2(x_1 - x_2)} + \frac{4z(z_1 - z_2)}{2(x_1 - x_2)}x_1 + \\ x_1^2 + z^2 - 2zz_1 + z_1^2 - R^2 = 0\end{gathered} \tag{2.25}$$

Multiplying Eq. (2.25) by $4(x_1 - x_2)^2$ gives:

$$\begin{gathered}A^2 - 4Az(z_1 - z_2) + 4z^2(z_1 - z_2)^2 - 4Ax_1(x_1 - x_2) + 8(z_1 - z_2)(x_1 - x_2)x_1 + \\ 4(x_1 - x_2)^2\left[x_1^2 + z^2 - 2zz_1 + z_1^2 - R^2\right] = 0\end{gathered} \tag{2.26}$$

Factorising Eq. (2.26) into multiples of z^2, z and units gives:

z^2:

$$4(z_1 - z_2)^2 + 4(x_1 - x_2)^2$$

or

$$4\left[(z_1 - z_2)^2 + (x_1 - x_2)^2\right]$$

z:

$$-4A(z_1 - z_2) + 8(z_1 - z_2)(x_1 - x_2)x_1 - 8(x_1 - x_2)^2 z_1$$

or

$$4\left[2(z_1 - z_2)(x_1 - x_2)x_1 - 2(x_1 - x_2)^2 z_1 - A(z_1 - z_2)\right]$$

units: $A^2 - 4Ax_1(x_1 - x_2) + 4(x_1 - x_2)^2(x_1^2 + z_1^2 - R^2)$

This procedure allows Eq. (2.26) to be written in the form of a quadratic equation:

$$az^2 + bz + c = 0 \tag{2.27}$$

where:

$$a = 4\left[(z_1 - z_2)^2 + (x_1 - x_2)^2\right]$$

$$b = 4\left[2(z_1 - z_2)(x_1 - x_2)x_1 - 2(x_1 - x_2)^2 z_1 - A(z_1 - z_2)\right]$$

and

$$c = A^2 - 4Ax_1(x_1 - x_2) + 4(x_1 - x_2)^2(x_1^2 + z_1^2 - R^2)$$

Eq. (2.26) can then be solved using the standard quadratic solution:

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.28}$$

The results of Eq. (2.28) gave the *z* co-ordinates of the points of interception. The *x* co-ordinates were found by substituting the *z* co-ordinate back into Eq. (2.14) or Eq. (2.15). Once the co-ordinates of the points of interception were found the fraction of the particle cut off by the measurement line was calculated in much the same way as for the previous examples (Eqs. (2.6) to (2.10)). The difference between the two was that it was necessary to consider both the dark *and* light shaded region of Figure 2.28 when calculating the fraction.

Due to the mathematical complexity of determining the cut off fraction in three dimensions, it was decided that the best approach was to use numerical integration, and calculate the fraction cut off in pixel-size 'slices'. Figure 2.29 shows the intersection of a three-dimensional particle intersected by a cylinder used to measure the porosity.

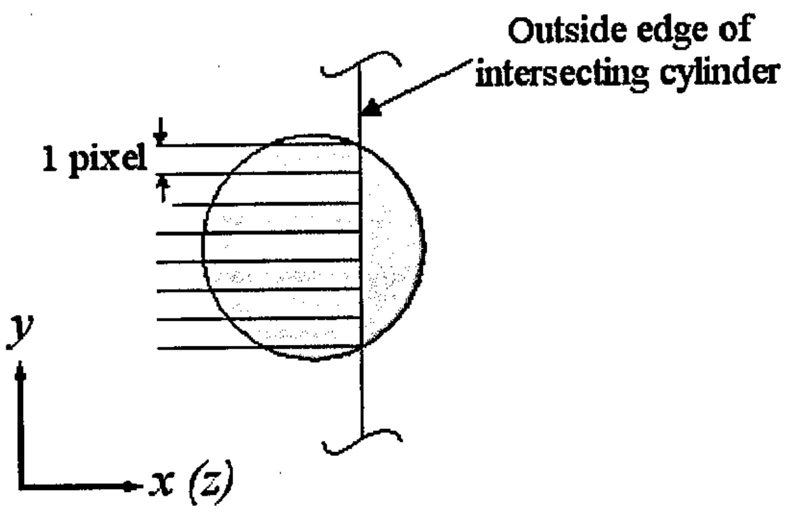


Figure 2.29- Calculation of the fraction of a sphere intersected by a cylinder

The number of slices was therefore equal to the diameter of the particle under scrutiny and the fractions of each of the vertical slices (two dimensions) intersected by the cylinder are combined to give a three-dimensional porosity.

The algorithm for the determination of the points of intersection and subsequent calculation of the fraction of the particle cut off can be found in the *CylPore* procedure (FUNCS.PAS) in Appendix A.

All characterising techniques for digitised images was carried out on a pixel-by-pixel basis. Measuring the porosity of a digitised image was a relatively simple procedure. The porosity was calculated as the number of pixels containing positive information (pixel value of '1') as a fraction of the total number of pixels in the image (pixel values of '1' + '0'). The algorithm for calculating the porosity of digitised image is *SEMPore* (POROSITY.PAS) in Appendix A.

2.4.5 Box counting fractal dimension

The box counting technique was used solely for the calculation of the fractal dimensions of digitised images of filter cakes, as it gives a mass fractal dimension that does not rely on the knowledge of the 'age' of the particles within the image. The first step in calculating the box counting fractal dimension was to input the image and determine the optimum size of grid to be overlaid. The size of the individual boxes within the grid had sides equal to a fraction of the largest dimension of the image, in much the same way as the structured walk method used a fraction of the Feret diameter. Only whole boxes were used to create the grid, the number of boxes calculated as the integer of the image size divided by the box size. Figure 2.30 shows a typical digitised image of a filter cake, and with a box counting grid overlaid.

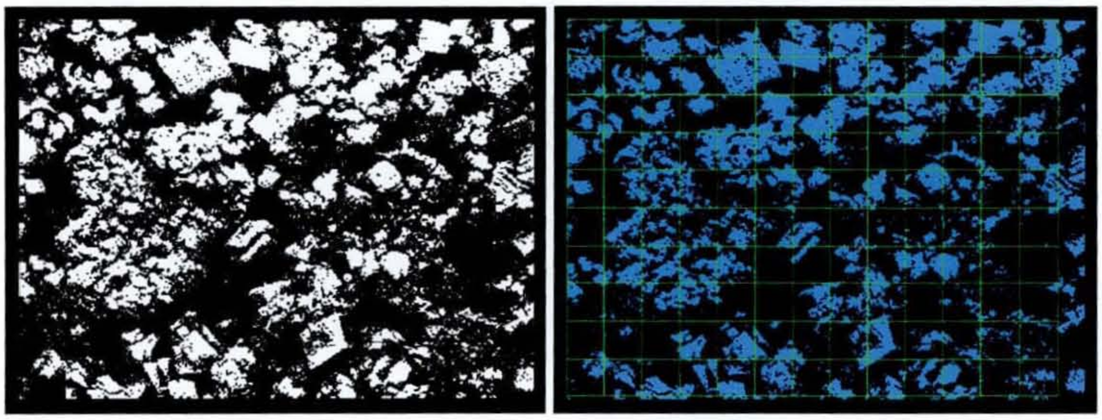


Figure 2.30 - Typical digitised image (originating from SEM photograph) of a filter cake (left), and with grid overlaid (right)

After the grid had been overlaid, a check was made inside each box to ascertain whether the box contained any positive information (pixel value of '1'). If the box did contain positive information, the mass counted was increased by an increment of 1 and the next box checked. Once each box in the grid had been checked, the total mass for the grid was stored, and the box size reduced. The algorithm for the box counting technique can be found in the *Boxes* procedure (BOXCOUNT.PAS) in Appendix A.

2.5 Conclusions

The growth of seed agglomerates in both two and three dimensions showed that changing the simulation parameters used to control their growth changed the structure, as measured in both physical and fractal terms. This lead to the development of a model to simulate the build up of particles on a virtual filter medium, which again showed that the structure could be altered by changing the way in which the agglomerates were built.

A robust method for determining both the physical and fractal nature of a range of simulated agglomerates and naturally occurring systems has now been developed and is used in Chapters 3 and 6 to characterise the fractal and physical properties of a number of different systems.

3 PROPERTIES OF SIMULATED AGGLOMERATES

This chapter details the results obtained for agglomerates simulated using the methods described in Chapter 2 (i.e. both seed agglomeration and filtration simulations). Data show how both the fractal and physical properties of both types of structure change as the system parameters controlling the growth are altered.

Also shown in Section 3.5 are examples of the variance in recorded data obtained from the analyses.

3.1 Methods of analysis

Each of the points on the graphs showing the fractal and physical characteristics of the agglomerates represents the average property of a minimum of 20 simulations. For the seed agglomerates, each data point represents the average property of 40 simulations. The total number of simulations performed is shown in Table 3.1.

Table 3.1 - Number of simulations performed

Type of Simulation	Number of Simulations
2-D Seed Agglomerate	840
3-D Seed Agglomerate	840
2-D Filtration Simulation	3360
3-D Filtration Simulation	2520
TOTAL	**7560**

The agglomerates have been analysed using the methods outlined in Chapters 1 and 2. As mentioned previously, not all the structures could be analysed using all the methods. The program described in Chapter 2 was able to automatically analyse a large number of structures and store the data generated in a text file, which could be imported into a spreadsheet. Using the program to automatically measure structures ensured a standardised measurement technique, which in turn reduced the potentially variable interpretation of the operator. When analysing the data, the program displayed not only the final result, but also where appropriate, a regression coefficient and confidence interval for the data. Whilst the confidence interval was only displayed on screen, the regression coefficient was saved along with the data, to allow for viewing in other applications.

It was possible for the analysis to fail at certain times, in particular when measuring the perimeter fractal dimension using the structured walk technique (for both the seed agglomerates and the filtration simulations). In this case (usually when the structured walk 'dividers' became stuck inside a feature of the agglomerate and thus the calculated perimeter rose above a certain sensible level), a fail-safe mechanism was activated and the data in question removed from the set. The fail-safe mechanism ensured that the program did not 'crash' during an analysis, and could continue to analyse the remainder of the data in the set. As well as the individual data and regression information from the analyses, the program calculated and stored a 'quick average' - simply the average value of the analysis results. This quick value was given as a guide only, and a spreadsheet was used to calculate an average value for the property in question. The number of each type of analysis carried out on the structures (both seed agglomerates and filtration simulations) is shown in Table 3.2. The total number of analyses carried out was **36240**. It is obvious from this value that the analysis of the simulated structures needed to be an automated, yet controlled process. Where '*3 x*' or '*2 x*' are shown in Table 3.2, this indicates the number of views or directions for each structured walk performed.

Table 3.2 - Number of analyses carried out on simulated structures

Type of Simulation	**Structured Walk**	**Enclosing Circle / Sphere**	**Radius of Gyration**	**Porosity**
2-D Seed Agglomerate	880	840	840	840
3-D Seed Agglomerate	*(3 x 840)* 2600	840	840	840
2-D Filtration Simulation	*(2 x 3360)* 6720	-	-	3360
3-D Filtration Simulation	*(2 x 3 x 2520)* 15120	-	-	2520
TOTAL	**25320**	**1680**	**1680**	**7560**

The program written for the project stored the analysis data for a given set of structures in a single file, enabling the user to calculate such parameters as variance, standard deviation etc., and also to check that no analysis values outside of given ranges were being generated (thus potentially indicating a bug in the program).

3.2 Analysis of seed agglomeration simulations

The structures built using the seed agglomeration model were analysed in two and three dimensions using three fractal and one physical measurement. The fractal measurements employed were the structured walk, enclosing circle / sphere and radius of gyration, whilst the physical measurement was the calculation of the structures' porosity.

The variation of the structure of agglomerates with the simulation parameters can be seen in Figure 3.1 to Figure 3.21 (contained in Table 3.3) and Figure 3.22 to Figure 3.24 (Table 3.4).

Table 3.3 shows two-dimensional seed agglomerates, built using diffusion influences between 0% (Figure 3.1) and 100% (Figure 3.21) in 5% increments. Whilst the change in structure between each agglomerate and its neighbour are not immediately noticeable, the differences are clearly visible over the whole range of diffusion influences. As shall be seen, these differences can also be quantified by the various measurement techniques used to classify the structures.

Table 3.4 shows three examples of three-dimensional agglomerates, again built using varying diffusion influences. These three examples again serve to demonstrate the visible change in structure with increasing diffusion influence. In the case of the three-dimensional examples, as only three are shown here, the changes are perhaps more marked.

Table 3.3 – Examples of 2-D agglomerates showing changes in structure with increasing diffusion influence (all agglomerates contain 800 particles)

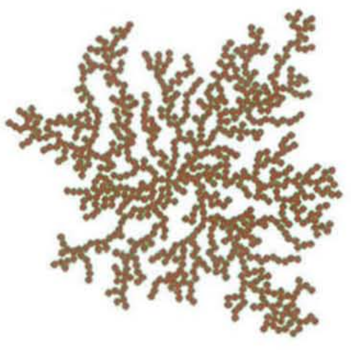
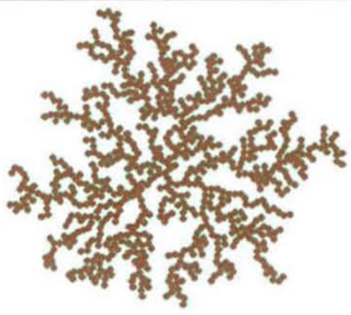
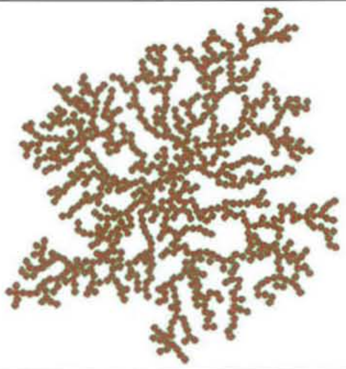
		

Figure 3.1 - 0%	*Figure 3.2 - 5%*	*Figure 3.3 - 10%*

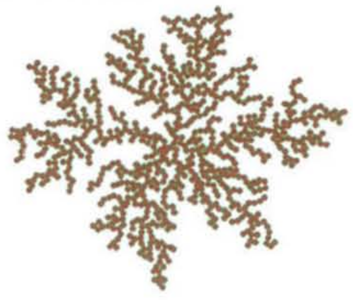
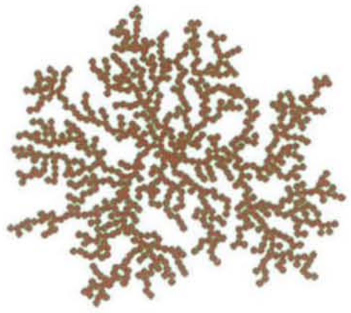
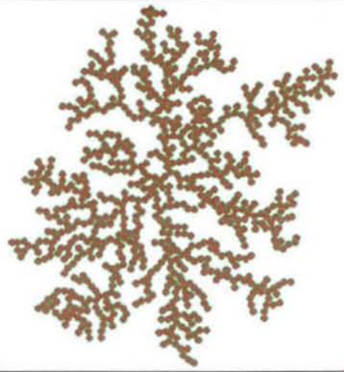

		

Figure 3.4 - 15%	*Figure 3.5 - 20%*	*Figure 3.6 - 25%*

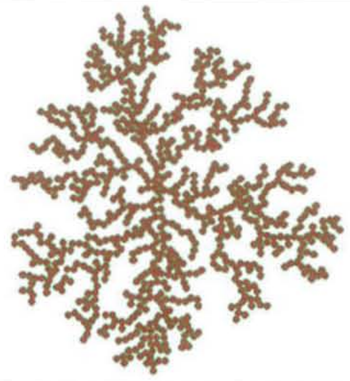
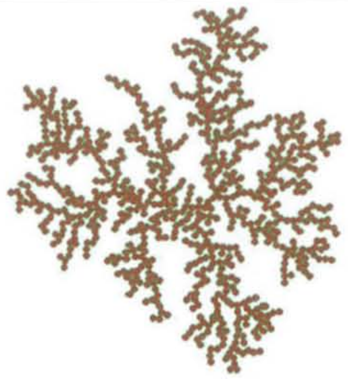
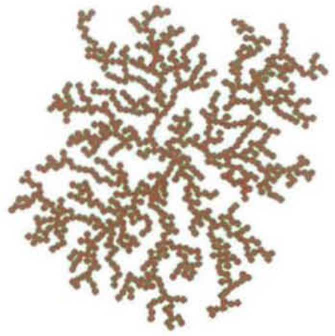

		

Figure 3.7 - 30%	*Figure 3.8 - 35%*	*Figure 3.9 - 40%*

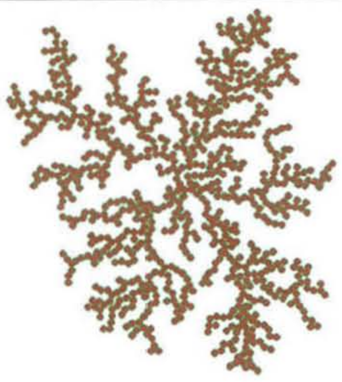
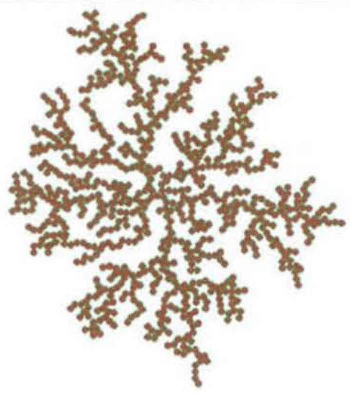
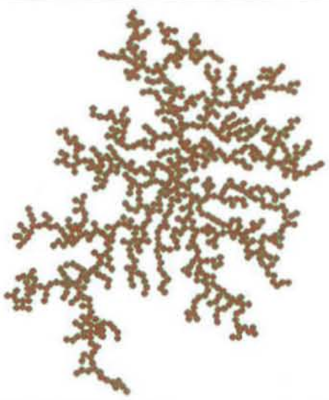
		

Figure 3.10 - 45%	*Figure 3.11 - 50%*	*Figure 3.12 - 55%*

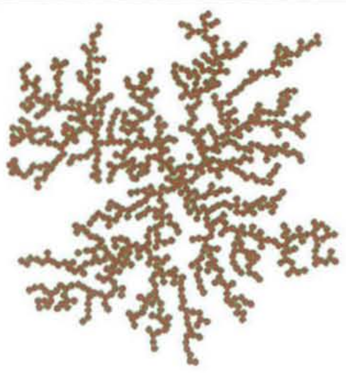
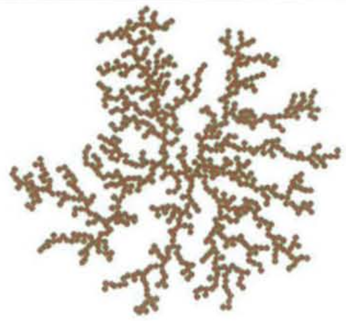
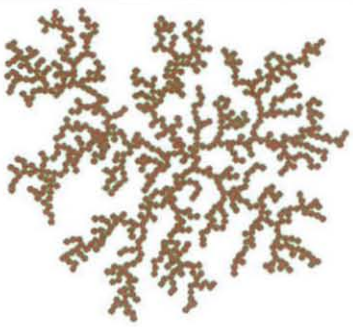

		

Figure 3.13 - 60%	*Figure 3.14 - 65%*	*Figure 3.15 - 70%*

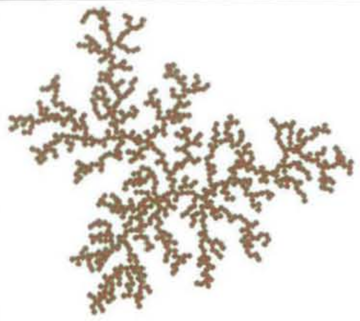
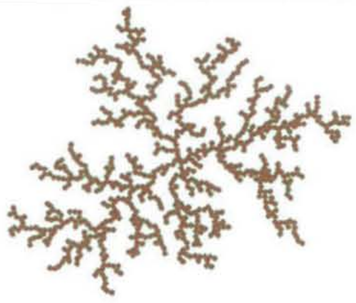
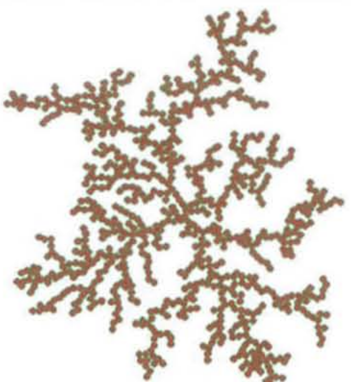

		

Figure 3.16 - 75%	*Figure 3.17 - 80%*	*Figure 3.18 - 85%*

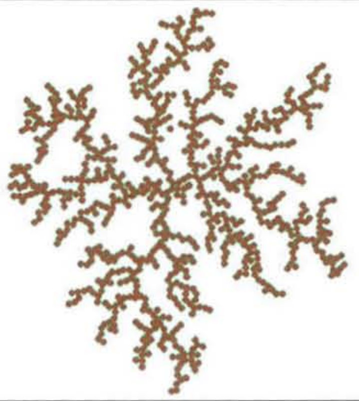
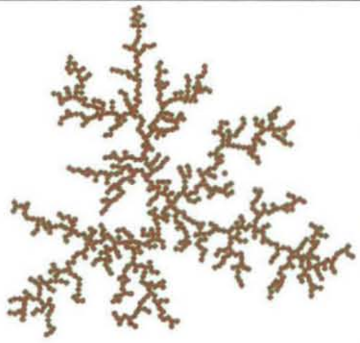
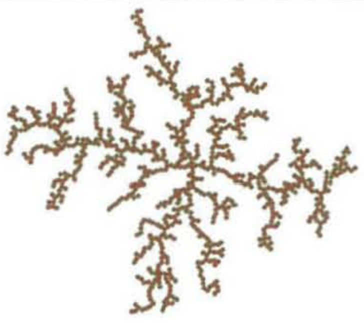
		

Figure 3.19 - 90%	*Figure 3.20 - 95%*	*Figure 3.21 - 100%*

Table 3.4 - Examples of 3-D agglomerates showing changes in structure with increasing diffusion influence (all agglomerates contain 800 particles)

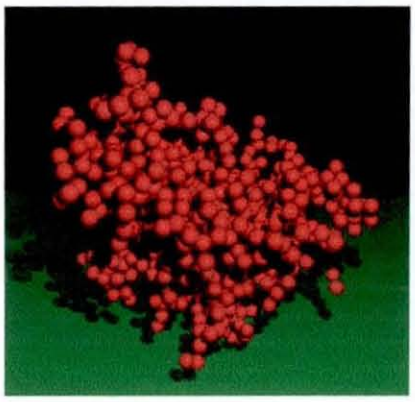
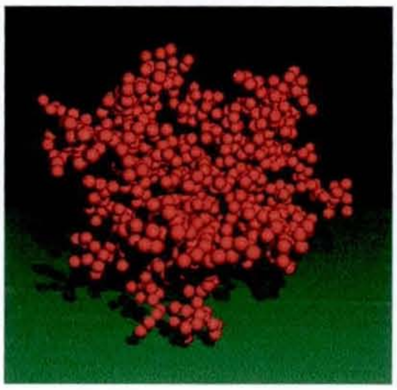
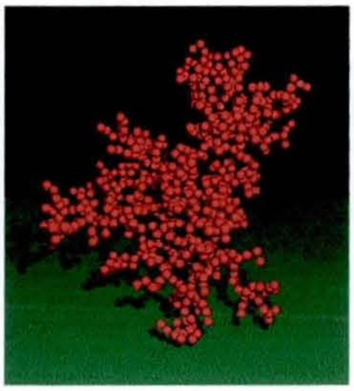
		

Figure 3.22 - 0%	*Figure 3.23 - 50%*	*Figure 3.24 - 100%*

3.2.1 Structured walk analysis

Figure 3.25 shows a typical structured walk measurement around a two-dimensional seed agglomerate. The start point was determined as the particle furthest away from the first particle in the agglomerate (as described in Chapter 2). The structured walk was always performed in a clockwise direction around the agglomerate, until the end of the current step was within one steplength of the start point of the walk. This technique was repeated for each of the steplengths in the chosen range (generally 0.08 to 0.32 times the maximum Feret diameter), obtaining a perimeter in each case.

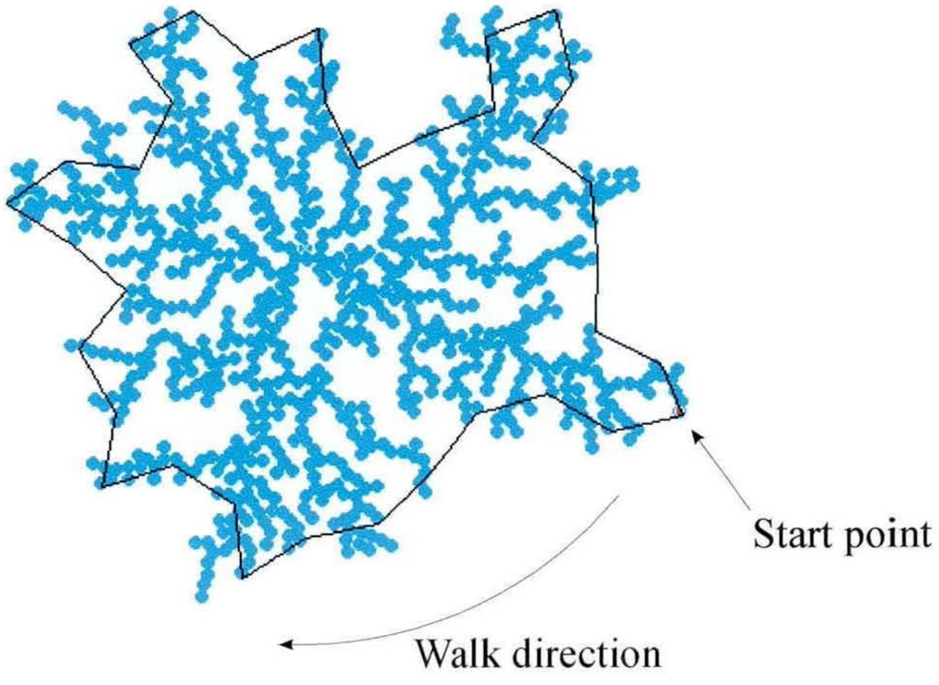


Figure 3.25 - Example of a structured walk measurement around a two-dimensional seed agglomerate ($\lambda = 0.1\ F$)

Figure 3.26 shows the perimeter fractal dimensions, measured using the structured walk technique, for both two and three-dimensional agglomerates of the type shown in Figure 3.1 to Figure 3.21. Each of the two-dimensional agglomerates contained 800 particles of radius 3 pixels, whilst the three-dimensional agglomerates contained 1000 particles of the same size. The measurement used for analysing the two-dimensional agglomerates was identical to that shown in Figure 3.25, whilst the perimeter roughness of the three-dimensional agglomerates was measured using three different (front, side and top) projections of the agglomerates in two dimensions. The abscissa shows the percentage diffusion influence used to create the agglomerates, from ballistic interception (i.e. 0% diffusion) to Brownian motion (i.e. 100% diffusion). The differences between the ruggedness of the two-dimensional agglomerates compared with the projected three-dimensional agglomerates can be seen by the fact that the structured walk fractal dimensions vary between approximately 1.25 and 1.45 for the two-dimensional agglomerates (16% variation) and only between 1.15 and 1.25 (5% variation) for the three-dimensional agglomerates. This difference in variation is most likely due to the inability to measure the full shape properties of a three-dimensional object from a two-dimensional projection.

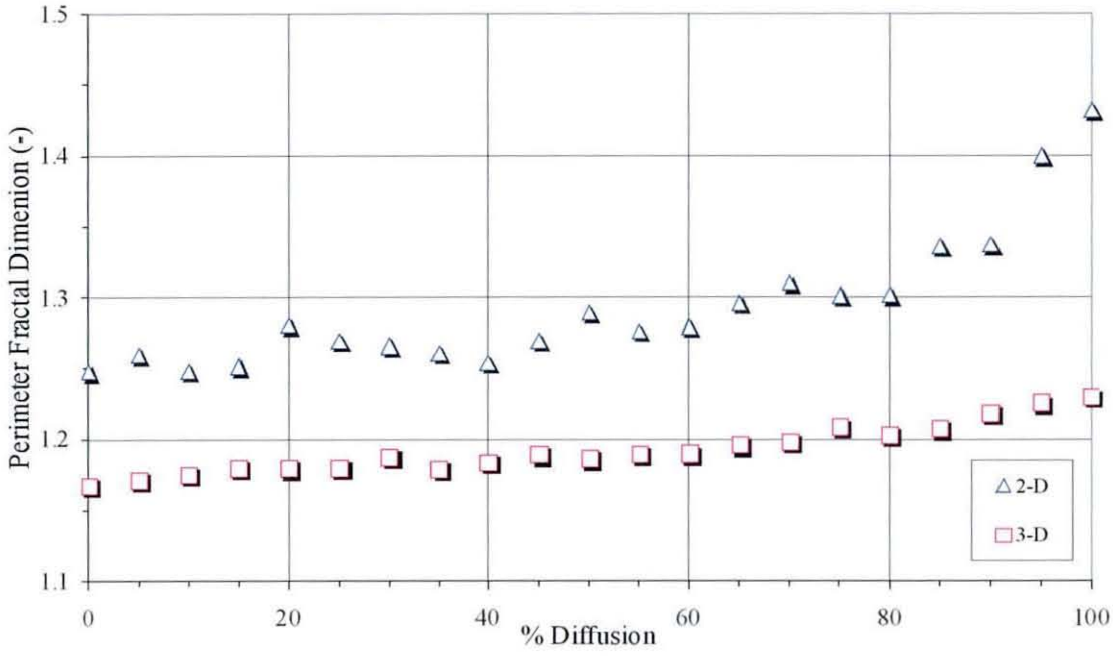


Figure 3.26 - Structured walk fractal dimensions for 2-D and 3-D (using two-dimensional projections) seed agglomerates

The results for the perimeter fractal dimension of two-dimensional agglomerates show a gradual rise in ruggedness until around 70% diffusion, whence a sharper rise can be seen. Although the results for two dimensions appear to have a much larger rise than those for three dimensions, if the latter is plotted on a larger scale, this shape would still be apparent, as shown in Figure 3.27.

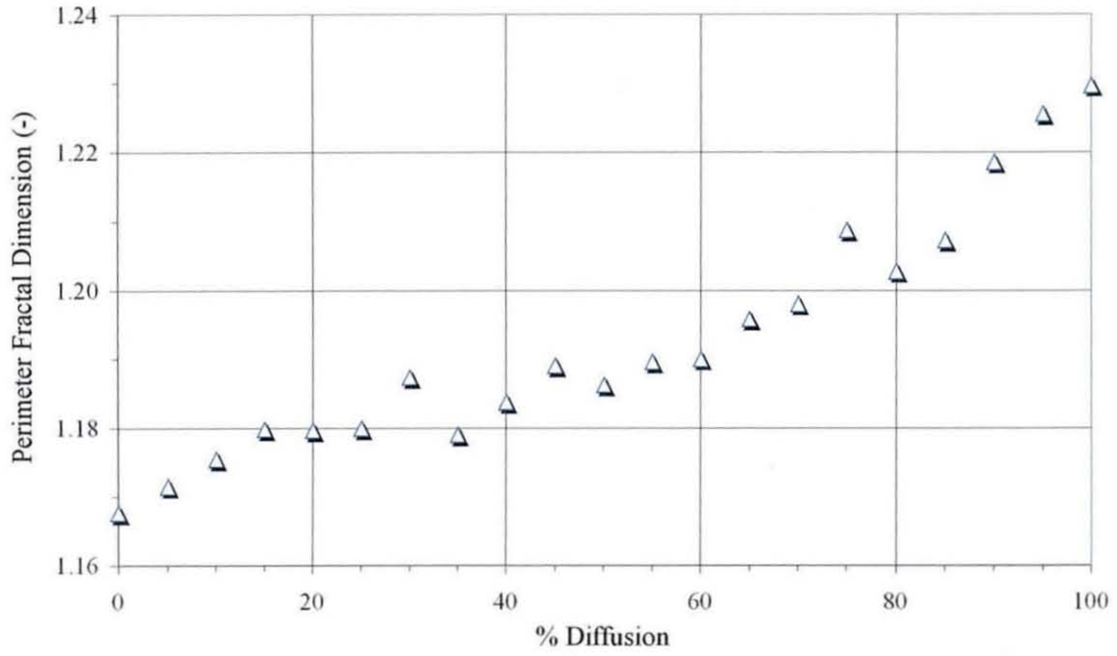


Figure 3.27 - Structured walk fractal dimensions for 2-D projections of 3-D seed agglomerates (average of three projections)

The fractal dimensions for the three-dimensional structures displayed in Figure 3.26 and Figure 3.27 represent a total of 3360 analyses, as they were measured on three differing projections (front, side and plan views) for the three-dimensional agglomerates. Figure 3.28 shows that the view from which the surface fractal dimensions were measured made little difference to the results.

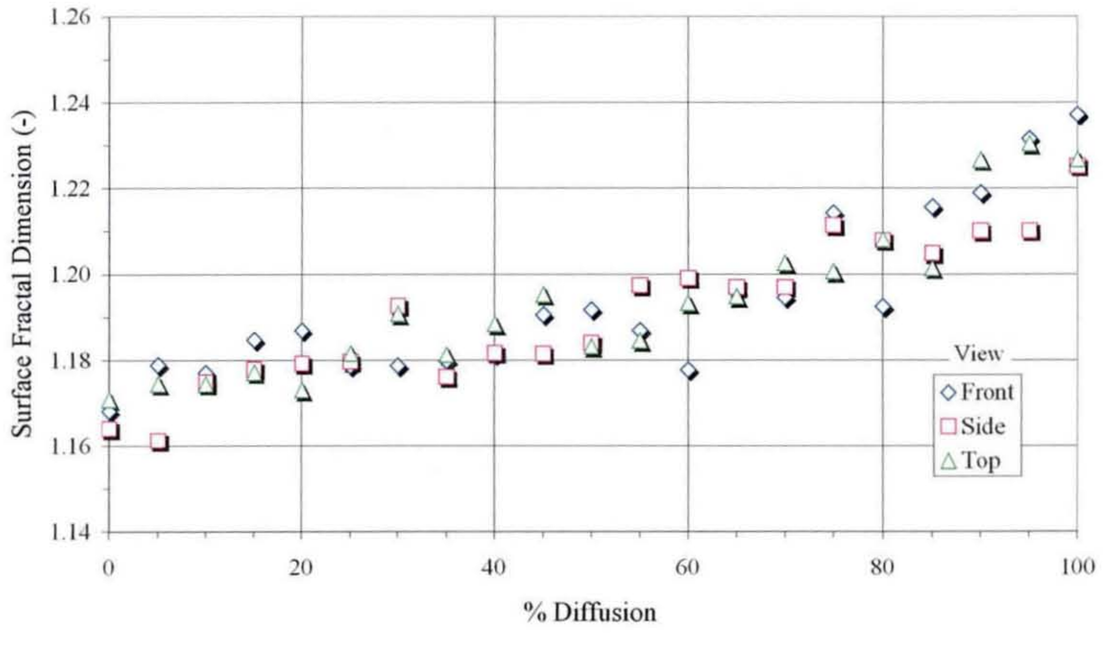


Figure 3.28 - Variation in fractal dimension with view for 3-D seed agglomerates

3.2.1.1 Relationship between number of particles and fractal dimension

As well as the structured walk analysis for the entire structure of each agglomerate, work was also performed to investigate the relationship between the number of particles in the agglomerate and its perimeter fractal dimension. The perimeter fractal dimension was measured for each 100 particles in both two and three-dimensional agglomerates (each containing a total of 800 particles. One agglomerate for each of 0, 25, 50, 75 and 100% diffusion influence was chosen at random, and the fractal dimensions measured. The 'growing' two-dimensional agglomerate (with 50% diffusional influence), as measured by the program can be seen in Figure 3.29 to Figure 3.32. The perimeter fractal dimensions of these structures are shown in both Table 3.5 and Figure 3.33.

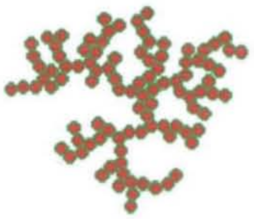


Figure 3.29 - First 100 particles of a 2-D seed agglomerate

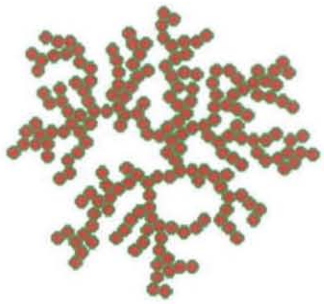


Figure 3.30 - First 200 particles of a 2-D seed agglomerate

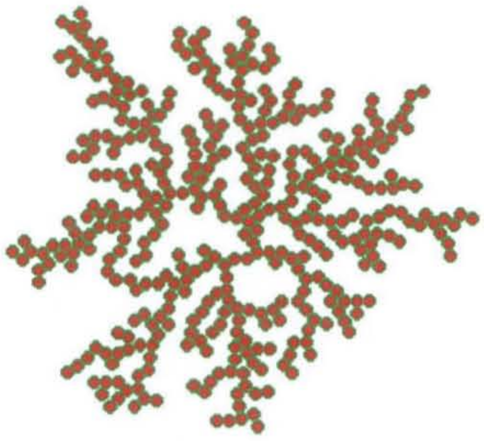


Figure 3.31 - First 400 particles of a 2-D seed agglomerate

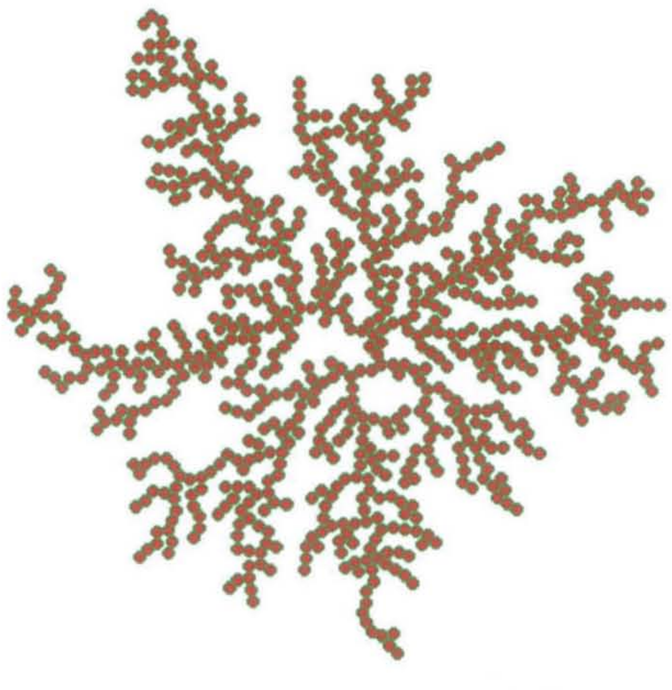


Figure 3.32 - Complete 2-D agglomerate (800 particles)

A typical set of data (two-dimensional seed agglomerate built using 50% diffusion influence) is shown in Table 3.5. The data in Table 3.5 are obtained from the structures shown in Figure 3.29 to Figure 3.32 and is also represented in the results for 50% diffusion shown in Figure 3.33.

Figure 3.33 shows the results for two-dimensional agglomerates. Generally, the fractal dimensions of agglomerates with fewer particles appeared to be higher, although no definite pattern can be seen, and the results for 100% diffusion influence do not follow the same pattern as the rest of the data. As far as the regression coefficient of the data is concerned (the measure used to determine the accuracy of the data), again no trend can be seen with the structures studied.

Table 3.5 - Typical fractal dimensions for two-dimensional seed agglomerates

Number of Particles	Perimeter Fractal Dimension
100	1.412
200	1.317
300	1.256
400	1.229
500	1.248
600	1.303
700	1.277
800	1.240

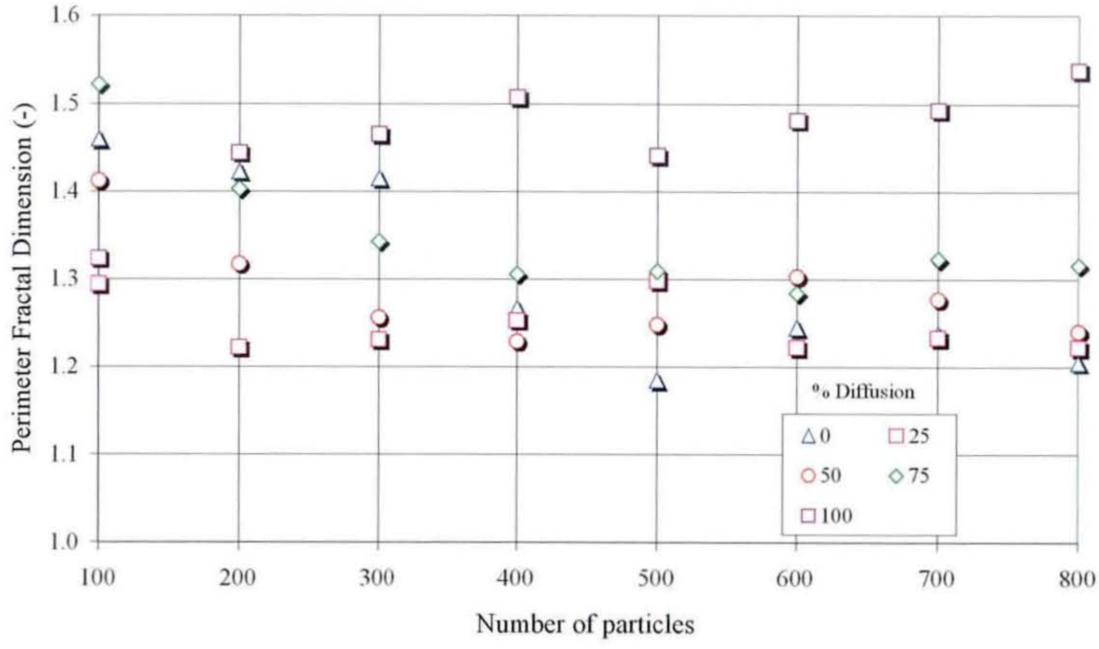


Figure 3.33 - Relationship between the number of particles in an agglomerate and the perimeter fractal dimension (shown for two-dimensional agglomerates)

The same analysis was carried out for projections of three-dimensional agglomerates, again consisting of 800 particles and built using 0, 25, 50, 75 and 100% diffusion influence. In the case of three dimensions, the perimeter fractal dimensions appeared to be less dependent of the number of particles in the agglomerate (see Figure 3.34).

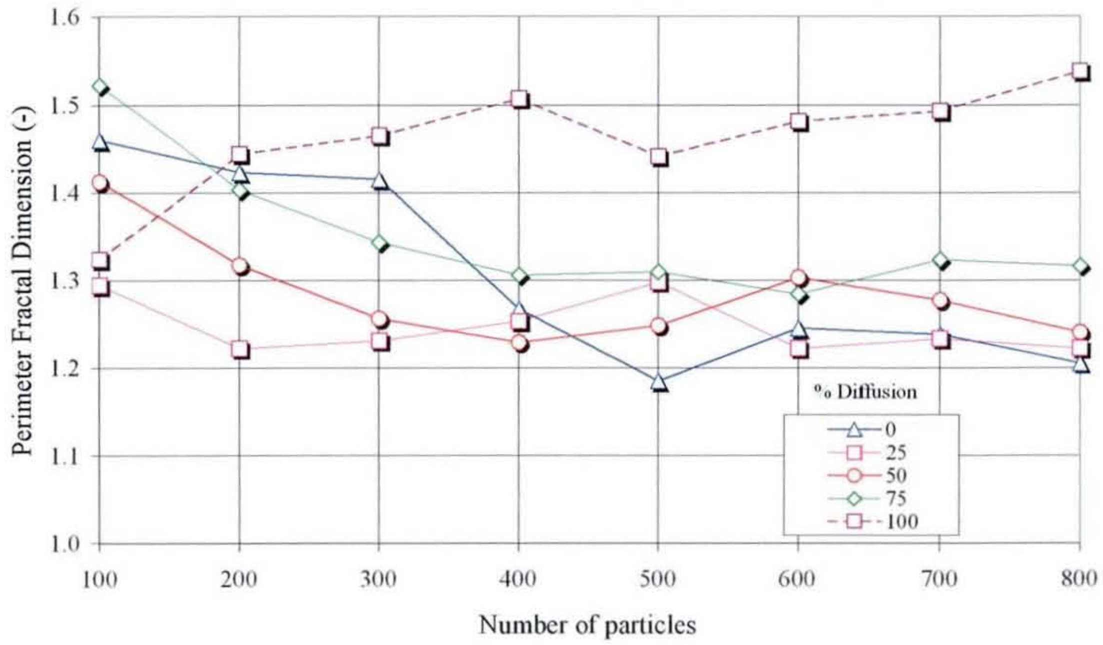


Figure 3.34 - Relationship between the number of particles in an agglomerate and the perimeter fractal dimension for projections of 3-D agglomerates

3.2.2 Enclosing circle / sphere analysis

It is perhaps more difficult to compare the results for the enclosing circle results because the theoretical maximum density fractal dimension for two-dimensional structures is 2.00 and the maximum for three-dimensional structures is 3.00 (the respective Euclidean dimensions). The results for both the two and three-dimensional agglomerates, however, do show the same general trends as each other, with a decline in the density fractal dimension with increasing diffusion influence. The enclosing circle fractal dimension results for two-dimensional agglomerates can be seen in Figure 3.35, whilst those for three-dimensional agglomerates are shown in Figure 3.36.

As with the perimeter fractal dimension, a steeper change was noted at around 70% diffusion for the two-dimensional analysis. The two-dimensional enclosing circle results shown in Figure 3.35 vary from 1.96 to 1.76 (a reduction of 10.2%), again with an apparent step change in gradient at approximately 70% diffusion ($D \approx 1.92$).

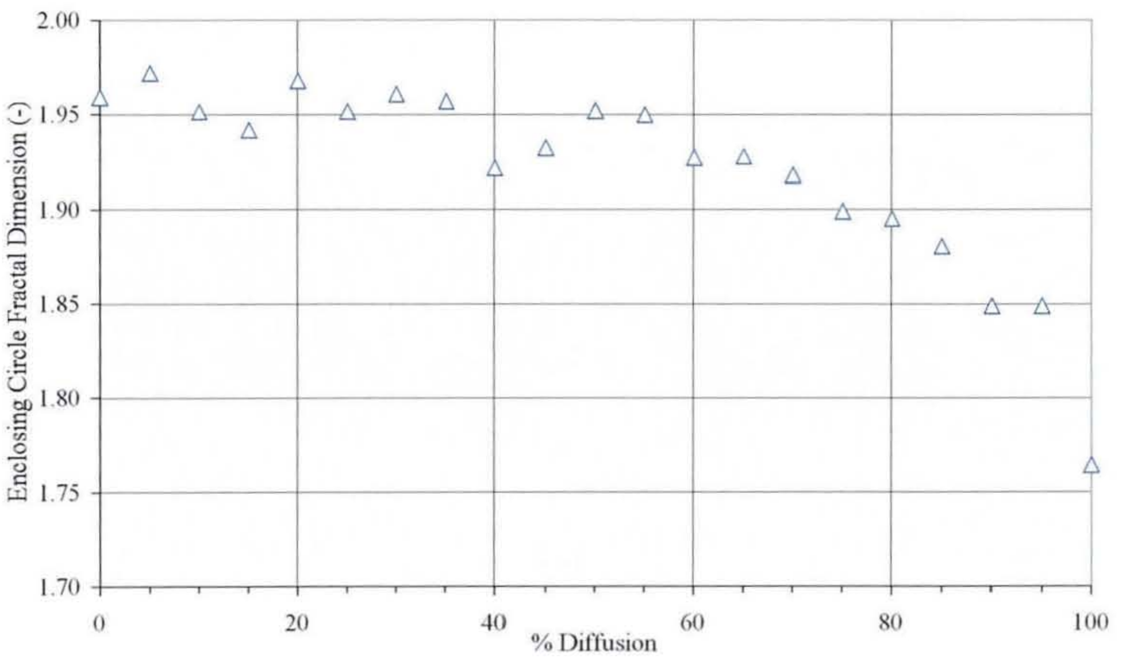


Figure 3.35 - Enclosing circle fractal dimensions for 2-D seed agglomerates

The three-dimensional simulations, on the other hand, seemingly showed a more progressive decrease in enclosing sphere density with increasing diffusion influence (Figure 3.36). The three-dimensional enclosing sphere fractal dimension decreased from 2.72 to 2.44, this time giving a decrease of 10.3% (similar to the decrease in the enclosing circle fractal dimension for two-dimensional agglomerates).

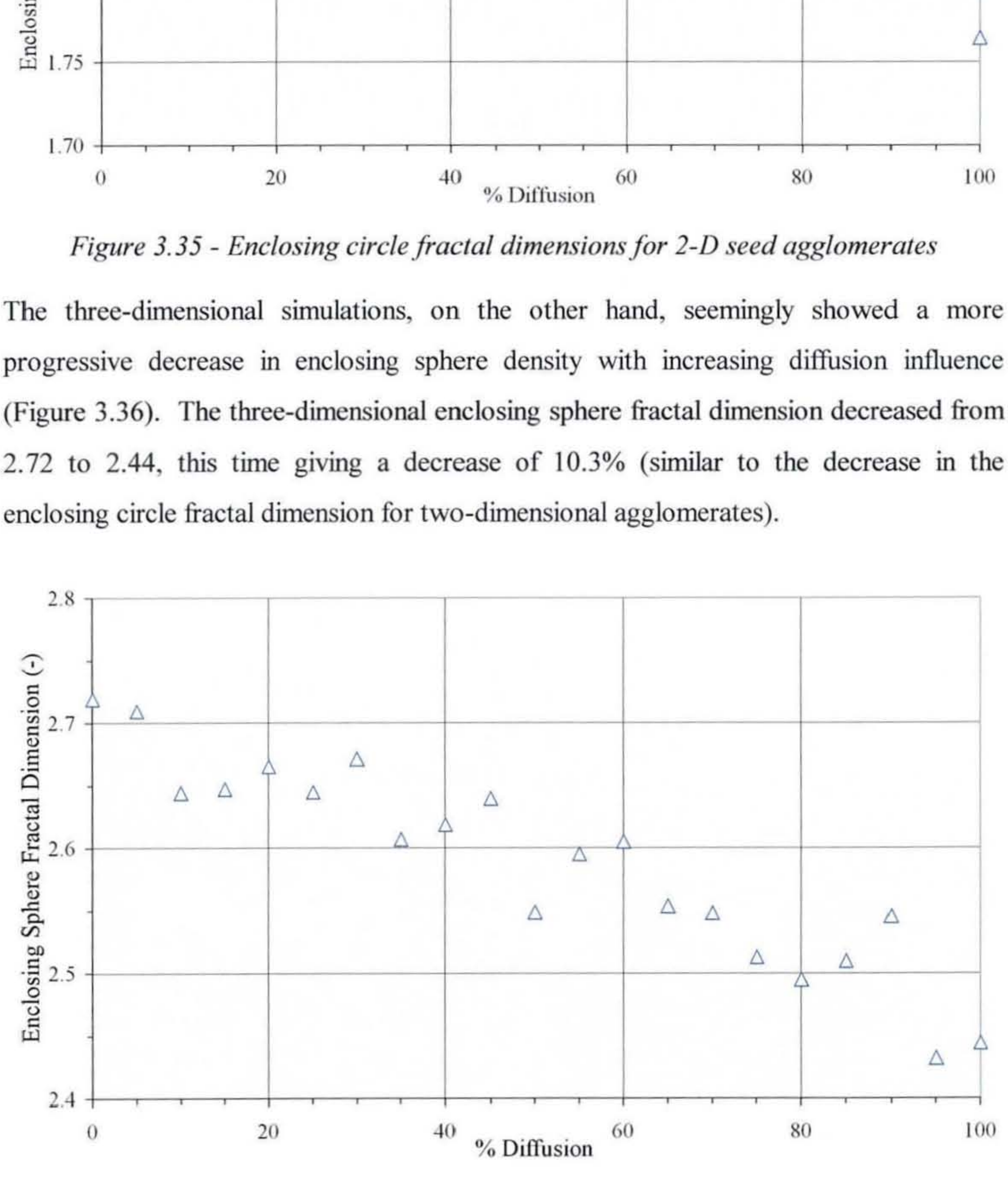


Figure 3.36 - Enclosing sphere fractal dimensions for 3-D seed agglomerates

3.2.3 Radius of gyration analysis

Figure 3.37 shows that the density fractal dimension of the two-dimensional seed agglomerates decreased as the diffusion influence of the simulation increased. Again, a steeper rate of change was seen around 70% diffusion influence. The radius of gyration measured for the two-dimensional structures dropped from 1.95 to 1.74 as the diffusion influence was increased from 0% to 100%, with the radius of gyration fractal dimension at 70% being ≈ 1.91.

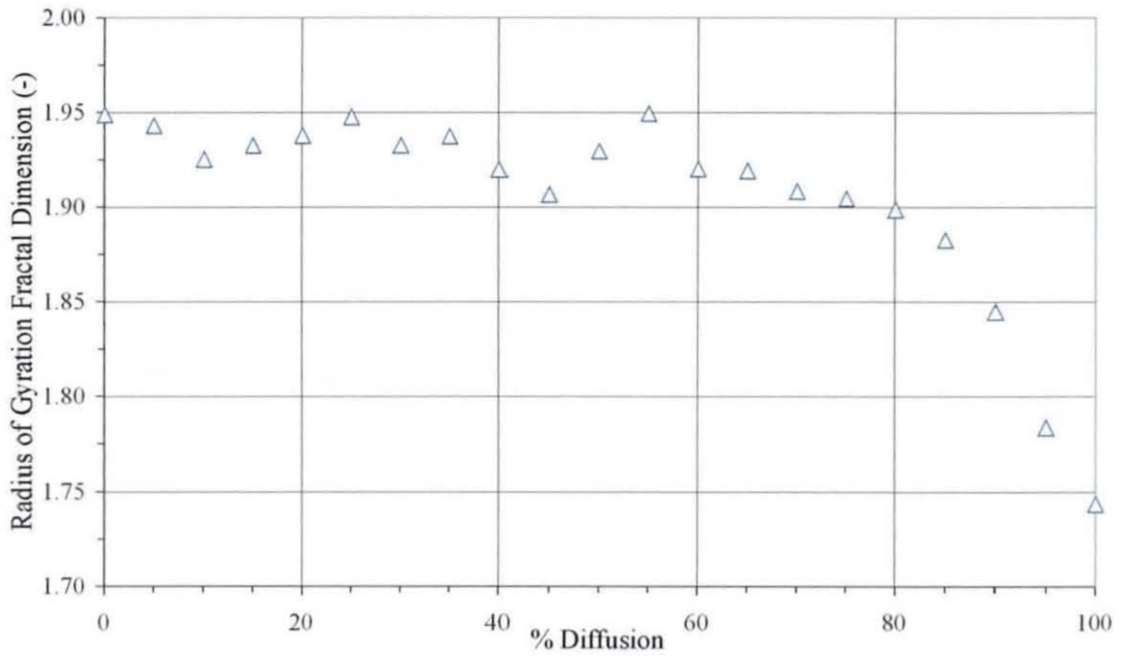


Figure 3.37 - Radius of gyration fractal dimensions for 2-D seed agglomerates

The radius of gyration fractal dimension for three-dimensional agglomerates again decreased with increasing diffusion influence. As with the enclosing sphere fractal dimension, the change was more gradual than for the two-dimensional fractal dimension. The radius of gyration decreased from 2.99 to 2.54 as the diffusion influence increased from 0 to 100%.

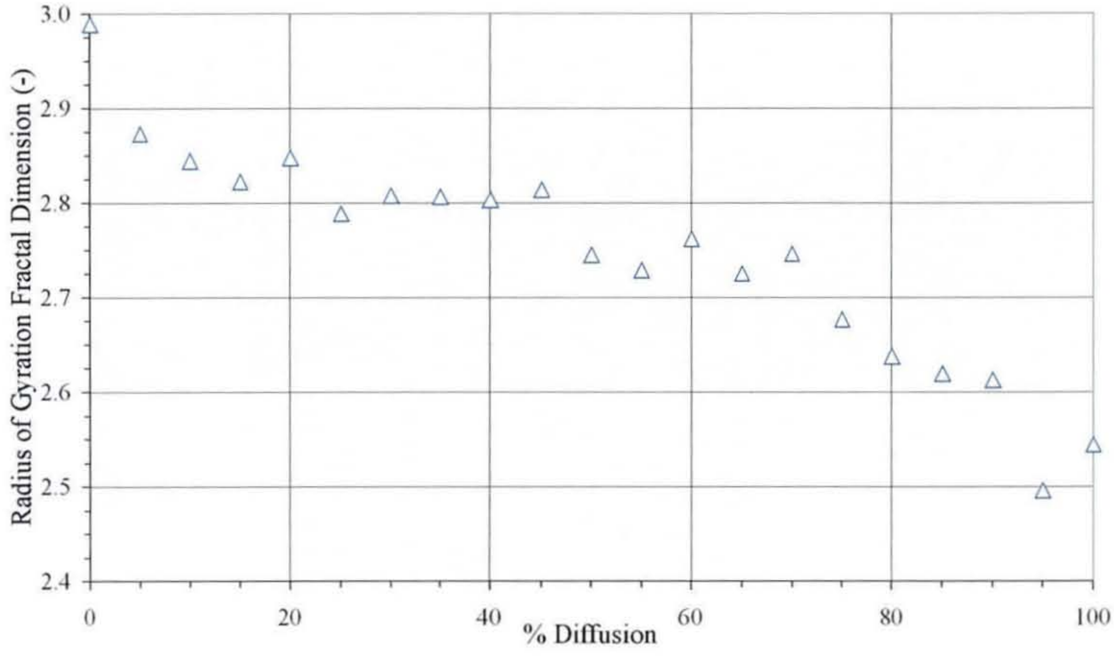


Figure 3.38 - Radius of gyration fractal dimensions for 3-D seed agglomerates

The data for two-dimensional radius of gyration analysis showed reduced scatter compared to that for the three-dimensional analysis. This can be seen in the smoothness of line in Figure 3.37 compared to that in Figure 3.38.

3.2.4 Porosity analysis

The fourth method of analysis applied to the seed agglomerates was the measurement of their porosity. As described previously, this was calculated as the fraction of the structure containing matter (area or volume) compared to the whole. In the case of the seed agglomerates, the 'whole' structure was taken as the size of a circle or sphere that would encompass the agglomerate completely.

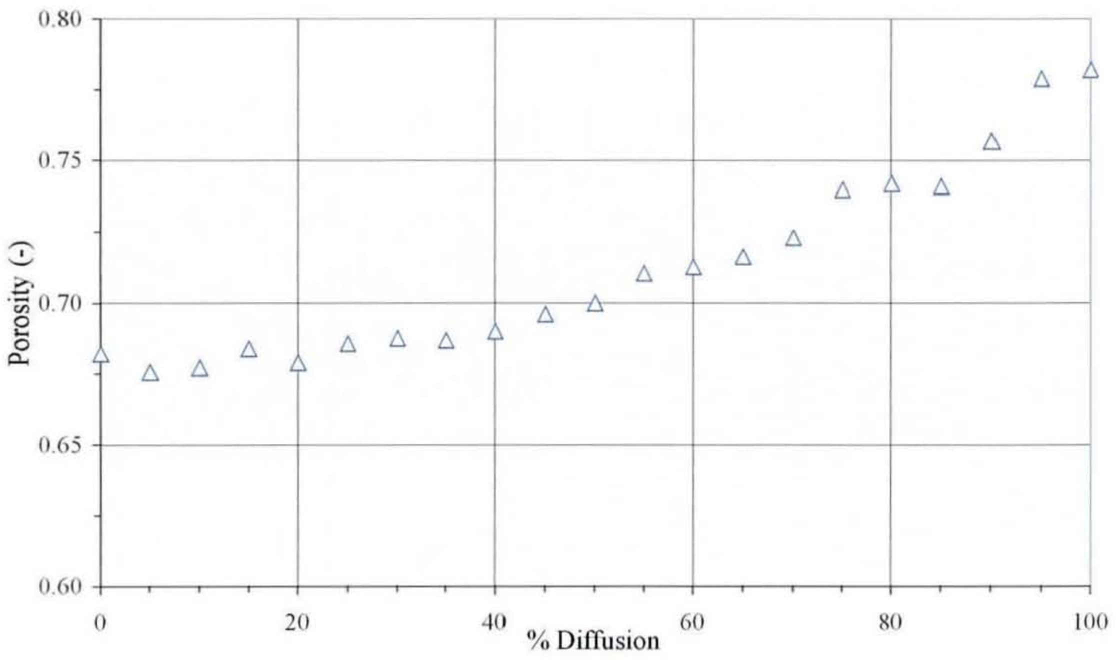


Figure 3.39 - Porosities of 2-D seed agglomerates

It can be seen from Figure 3.39 that the porosity of the two-dimensional seed agglomerates progressively increased with the extent of diffusion influence. A steeper change may also be apparent at around 70% diffusion (porosity ≈ 0.72), which perhaps confirms the legitimacy of the other (fractal) measures. The porosity of the two-dimensional agglomerates increased from 0.68 to 0.78 as the diffusion influence increased from 0 to 100%.

Figure 3.40 shows the porosities measured for three-dimensional seed agglomerates. It can be seen that only a small change in porosity occurs as the diffusion influence is varied. In fact, the porosity only increases from 0.89 to 0.94 over a 100% change in diffusion influence.

The porosity of the three-dimensional agglomerates appears to be very high. After careful analysis of the program, these values were found to be correct, and can be attributed to the dendritic nature of the agglomerates and the method used to analyse them (i.e. the use of a total enclosing circle). The method takes a sphere of increasing size, with a maximum radius equal to the distance between the centre of gravity of the agglomerate and the outermost particle. The volume of the particles contained within the sphere is then ascertained and the porosity of the agglomerate calculated as the fraction of the sphere not occupied by the particles.

For example, the distance between the centre of gravity and outermost particle for a typical three-dimensional ballistic agglomerate (i.e. one built with 0% diffusion influence) containing 1000 particles was calculated as 85.91 pixels which equates to a volume of 2,655,949 pixels (where pixels can be used as a length, area or volume measurement). The particles themselves, all with a radius of 3 pixels, will have a combined volume of just 113,097 pixels, giving a porosity of 95.7 %. The distance between the centre of gravity and outermost particle in an agglomerate built using 100% diffusion influence, on the other hand was calculated as 105.11 pixels, giving a sphere volume of 4,864,519 pixels. As the agglomerate contained the same number of particles as the ballistic agglomerate, the porosity was calculated as 97.7%.

The minimum value for the average porosity (for the ballistic interception model) is considerably out of step with the rest of the data, with the porosity of the 5% diffusion influence agglomerates having an average porosity of 0.91. This means that the change in porosity across 95% of the range of diffusion influence (5% to 100%) is only 0.035 (3.9%). While this may be a small change, when plotted as in Figure 3.40, a change in porosity with diffusion influence can still clearly be seen, including the variation in the rate of change at around 70% as seen in the other results.

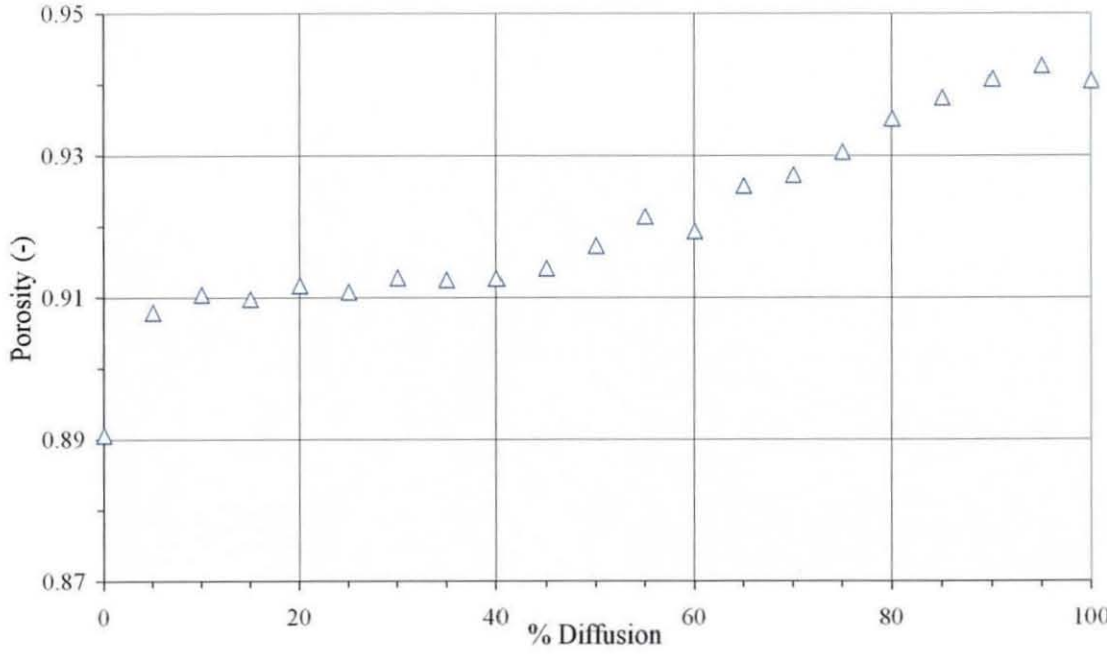


Figure 3.40 - Porosities of three-dimensional seed agglomerates

3.2.5 Conclusions

Table 3.6 - Summary of fractal and physical measurements for two and three-dimensional seed agglomerates[1]

% Diffusion	2-D				3-D			
	S. W.	E. C.	R. G.	P	S. W.	E. S.	R. G.	P
0	1.248	1.959	1.949	0.682	1.169	2.719	2.988	0.891
5	1.259	1.972	1.943	0.676	1.167	2.709	2.873	0.908
10	1.248	1.951	1.925	0.677	1.177	2.644	2.844	0.910
15	1.251	1.942	1.932	0.684	1.178	2.647	2.822	0.910
20	1.280	1.968	1.938	0.679	1.179	2.665	2.847	0.912
25	1.269	1.951	1.947	0.686	1.182	2.645	2.788	0.911
30	1.266	1.961	1.932	0.687	1.186	2.671	2.807	0.913
35	1.260	1.957	1.937	0.687	1.182	2.607	2.806	0.912
40	1.254	1.922	1.920	0.690	1.187	2.619	2.803	0.913
45	1.269	1.932	1.906	0.696	1.190	2.640	2.814	0.914
50	1.289	1.952	1.929	0.700	1.184	2.549	2.745	0.917
55	1.276	1.950	1.949	0.710	1.188	2.595	2.729	0.921
60	1.279	1.927	1.920	0.713	1.191	2.605	2.762	0.919
65	1.296	1.928	1.919	0.716	1.197	2.554	2.725	0.926
70	1.310	1.918	1.908	0.723	1.199	2.548	2.746	0.927
75	1.301	1.899	1.904	0.740	1.212	2.513	2.676	0.930
80	1.301	1.895	1.899	0.742	1.204	2.495	2.637	0.935
85	1.336	1.880	1.882	0.741	1.209	2.509	2.619	0.938
90	1.337	1.849	1.844	0.757	1.217	2.545	2.612	0.941
95	1.400	1.849	1.783	0.779	1.230	2.432	2.495	0.943
100	1.432	1.764	1.743	0.782	1.232	2.444	2.544	0.941

The results of the various analyses show that the method by which the agglomerates have been constructed influences the final physical and fractal characteristics of the structure.

[1] S.W. - Structured Walk; E.C./E.S. - Enclosing Circle / Sphere; R.G. - Radius of Gyration; P - Porosity

As the diffusion influence was increased, and particle motion moved from pure ballistic interception towards pure diffusion, agglomerate structure became more open and dendritic, as indicated by the accompanying decrease in the two density fractal dimensions used and the increase in the structures' porosity and perimeter fractal dimensions.

The change in structure of two-dimensional agglomerates is generally more evident than that for three-dimensions. In other words, the change in the openness of the structures with varying diffusion was quite visible for two dimensions, whereas the changes in structure for three dimensions required a quantitative software analysis. The full set of results for the analyses (both physical and fractal) is shown in Table 3.6.

3.3 Analysis of filtration simulations

The properties measured for the filtration simulations were the structured walk fractal dimension for the surface of the simulated cakes and their overall porosity. The results are evaluated for a range of sticking and downward probabilities. The porosity measure was chosen as a basic physical measure of the cake structure whilst the surface fractal was chosen to give an indication of the dendritic nature of the cakes.

As explained previously, the enclosing circle and radius of gyration techniques are not suitable for the measurement of the agglomerates built using the filtration simulation model. However, some work has been done to enable the measurement of the enclosing circle and structured walk fractal dimensions of the interstitial spaces within the two-dimensional filter cakes, as described in Section 3.4.

As with the seed agglomerates, the variation in structure was not particularly noticeable with each step increase in system parameters, but can readily be seen when viewing the data as a whole. Figure 3.41 to Figure 3.61 (Table 3.7) show typical filtration simulations performed in two dimensions, with 50% downward probability. All the structures contain the same number of particles (1000). The percentages in the title of each figure represent the sticking probability used in each simulation.

Table 3.7 – Examples of two-dimensional filtration simulations showing visible change in structure with increasing sticking probability

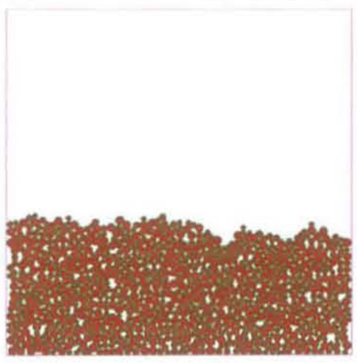
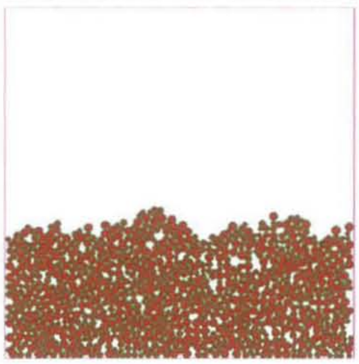
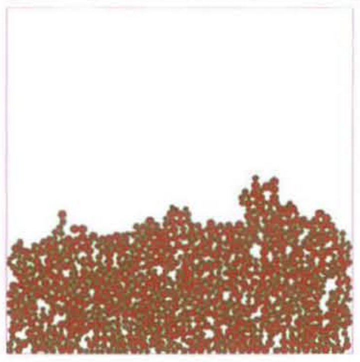
		

Figure 3.41 - 0%	*Figure 3.42 - 5%*	*Figure 3.43 - 10%*

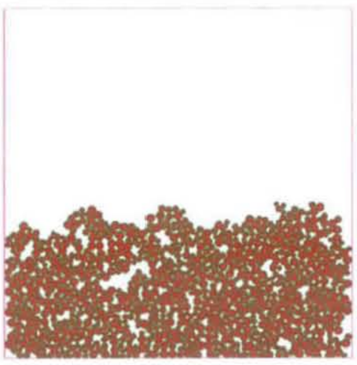
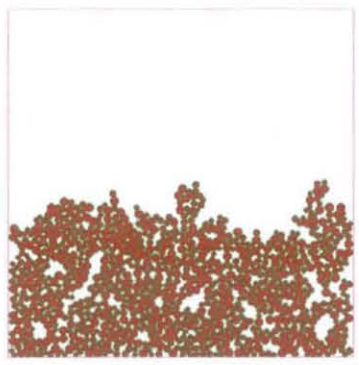
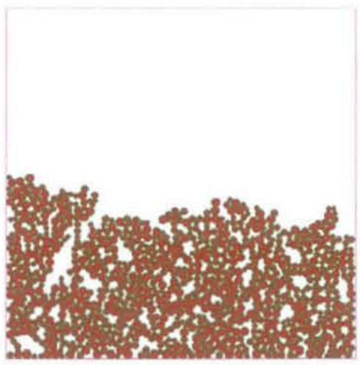

		

Figure 3.44 - 15%	*Figure 3.45 - 20%*	*Figure 3.46 - 25%*

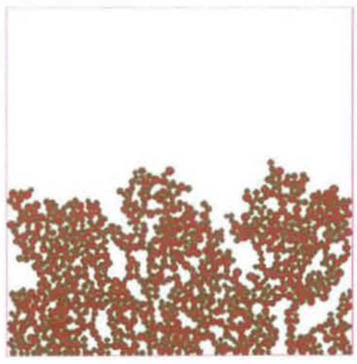
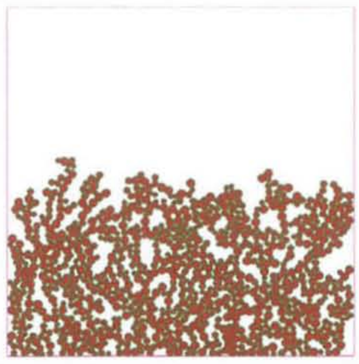
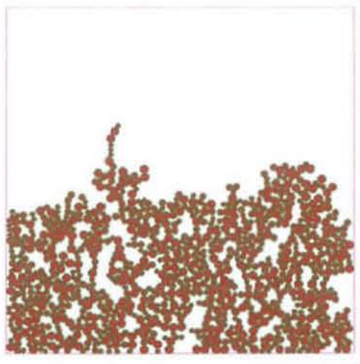

		

Figure 3.47 - 30%	*Figure 3.48 - 35%*	*Figure 3.49 - 40%*

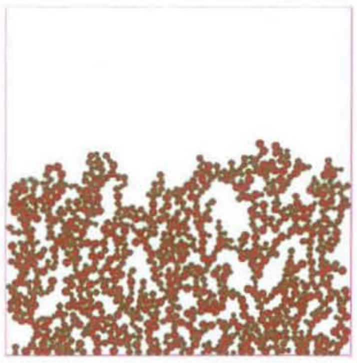


Figure 3.50 - 45%

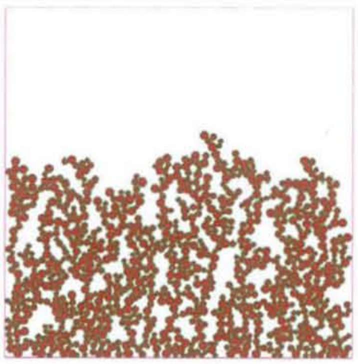


Figure 3.51 - 50%

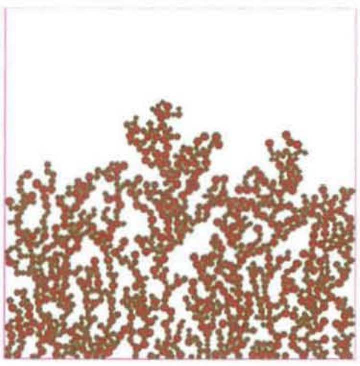


Figure 3.52 - 55%

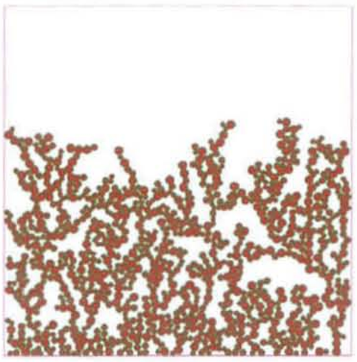


Figure 3.53 - 60%

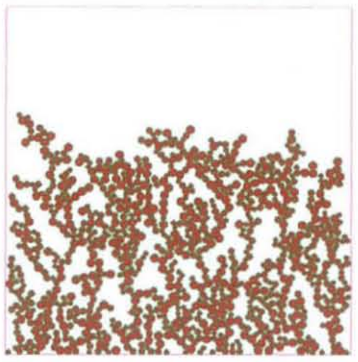


Figure 3.54 - 65%

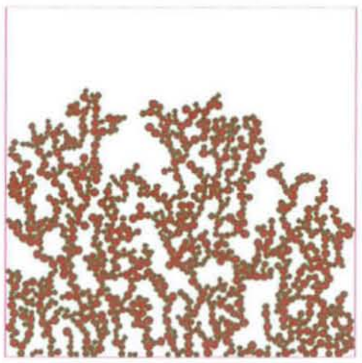


Figure 3.55 - 70%

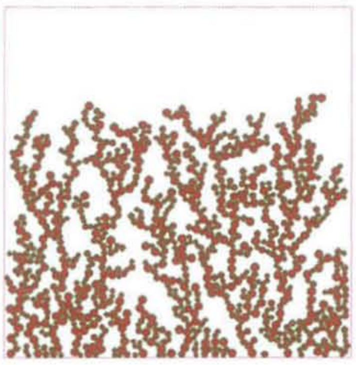


Figure 3.56 - 75%

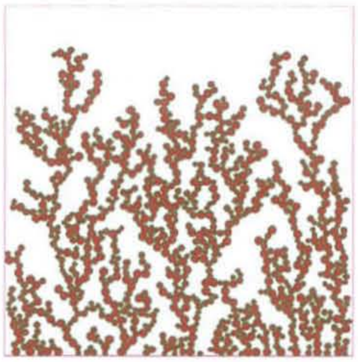


Figure 3.57 - 80%

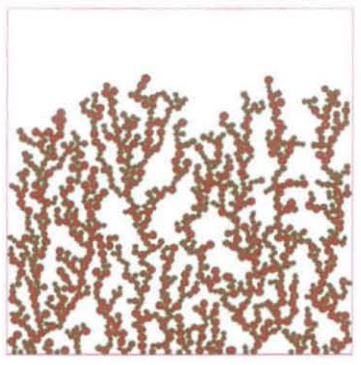


Figure 3.58 - 85%

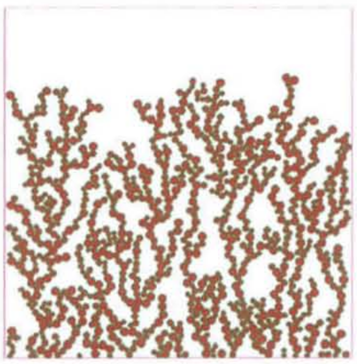


Figure 3.59 - 90%

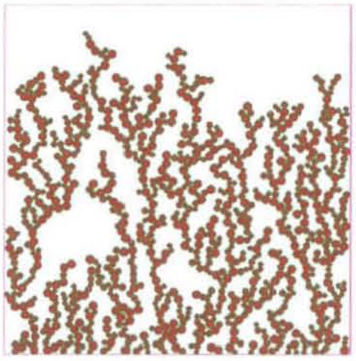


Figure 3.60 - 95%

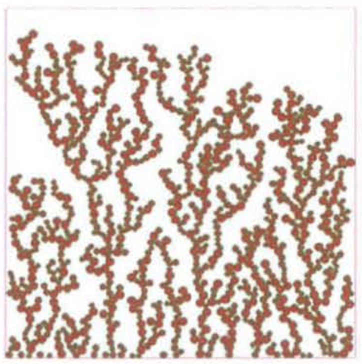


Figure 3.61 - 100%

3.3.1 Surface fractal dimension (structured walk) analysis

The surface fractal dimension was measured in the same way as for the seed agglomeration, by using a virtual pair of dividers at varying steplengths, and measuring the perimeter obtained. The structured walk analysis was performed in both directions (left to right and right to left) across the surface of the simulated filter cake. Figure 3.62 shows the structured walk carried out in each direction on a two-dimensional agglomerate (50% downward probability, 50% sticking probability). The validity of measuring the fractal dimension in both directions can be seen by the fact that in the example shown in Figure 3.62, each of the walk directions has missed a feature of the structure, but the walk in the opposite direction has picked it out.

The structured walk routine was optimised to minimise these errors, however, the computer algorithms could not be written to be as intuitive as a human operator performing the same function would have been. The cause of the dividers missing features is the way in which it was necessary to reset the 'dividers' for the next step of the structured walk routine. Any variations between the two walks, including features of the structures being missed by the 'dividers' were cancelled out by the number of simulations performed. Figure 3.63 shows that the overall error between the two directions, when averaged across the number of simulations was low.

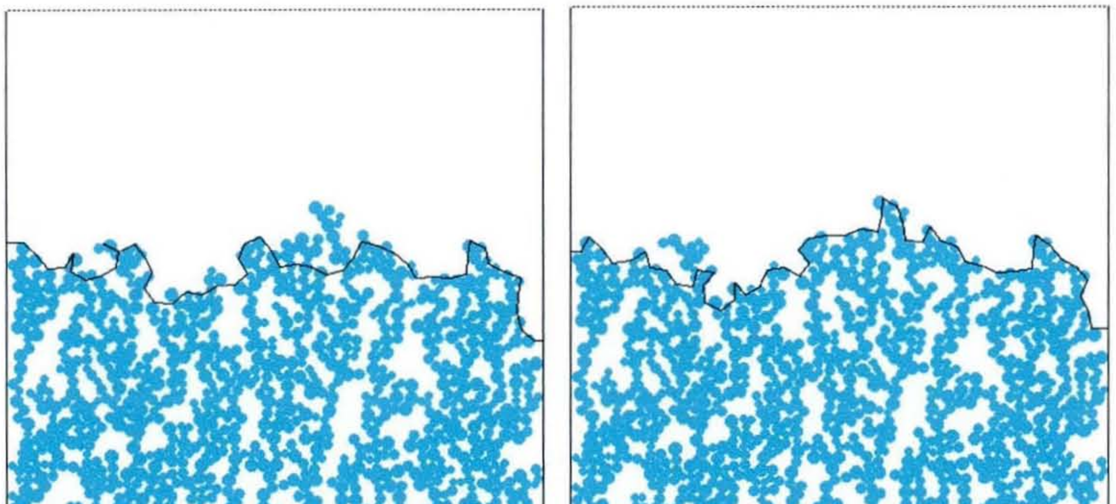


Figure 3.62 - Structured walk analysis performed on a two-dimensional filtration simulation in each direction - left to right (LHS) and right to left (RHS)

Figure 3.63 shows how the surface fractal dimension varied with the walk direction used to measure the property. It can be seen that the two directions are similar at low sticking probabilities, whilst slightly larger differences become visible for the cakes built using higher sticking probabilities. This is due to the fact that dendritic structures were more likely to form with cakes built using higher sticking probabilities, and thus different walk directions were more likely to miss parts of the structure.

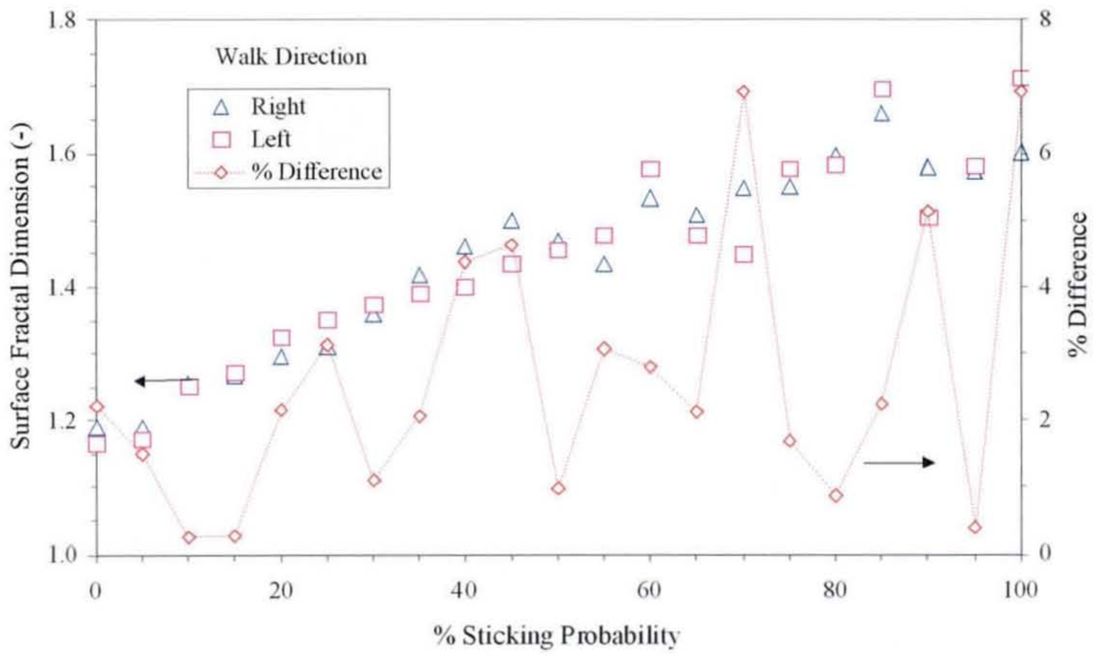


Figure 3.63 - Variation of surface fractal dimension with walk direction

Table 3.8 shows the values for the surface fractal dimension measured using the structured walk technique in both directions and the difference between the two. The cakes analysed to give the data for Figure 3.63 and Table 3.8 were all built using a downward probability of 50%. The percent difference shown in Table 3.8 is the *maximum* difference, i.e. the absolute value of the difference as a percentage of the smaller value for surface fractal dimension. Only sticking probability was chosen to illustrate the variation with simulation parameters, as the downward probability resulted in lower overall changes in the properties of structures formed.

Table 3.8 – Example of the typical variation in structured walk fractal dimension with walk direction

Sticking Prob.	Direction		% Difference
	Right	Left	
0	1.190	1.164	2.22
5	1.189	1.172	1.50
10	1.254	1.250	0.25
15	1.267	1.271	0.27
20	1.296	1.324	2.15
25	1.310	1.351	3.12
30	1.358	1.373	1.09
35	1.416	1.388	2.06
40	1.460	1.399	4.36
45	1.498	1.432	4.62
50	1.468	1.454	0.97
55	1.433	1.477	3.06
60	1.532	1.575	2.79
65	1.506	1.475	2.13
70	1.547	1.447	6.92
75	1.549	1.575	1.69
80	1.596	1.582	0.87
85	1.659	1.696	2.24
90	1.579	1.502	5.12
95	1.573	1.579	0.39
100	1.601	1.712	6.92

For three-dimensional filter cakes, the structured walk measurement was used on three views of the filter cake, each one rotated 120° from the previous position. Using the structured walk in this way means that each of the points on the graph for the surface fractal dimension represents not only twenty agglomerates, but six measurements of each of the twenty - a total of 120 analyses per point.

Figure 3.64 to Figure 3.66 show a typical three-dimensional agglomerate (50% downward probability, 75% sticking probability) rotated through 0° (Figure 3.64), 120° (Figure 3.65) and 240° (Figure 3.66) with the respective structured walk measurement (left to right) being carried out on their projection in two dimensions. A sticking probability of 75% has been chosen in order to demonstrate the effect of the rotation on the shape, due to the dendritic nature displayed by filtration simulations built using higher sticking probabilities.

It should be borne in mind that when visually comparing the three-dimensional rendering of the structure and the two-dimensional projection, the two may not look completely alike. This is because the structure is rendered from a viewpoint somewhere above the mid-point, allowing at least some the surface of the structure to be visible.

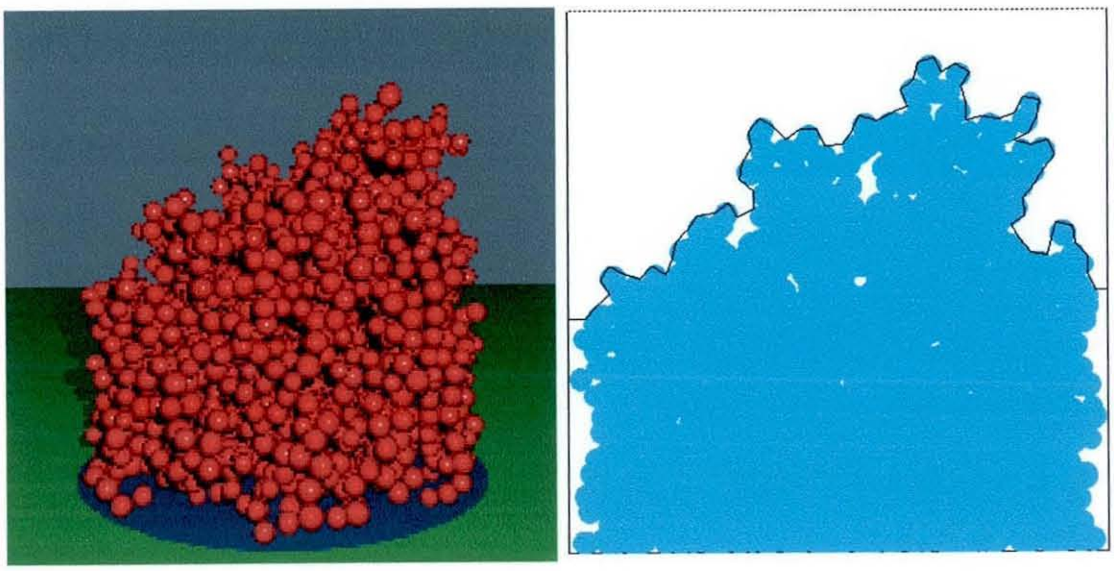


Figure 3.64 – 3-D agglomerate rotated through 0° with respective structured walk measurement (λ = 0.1 F)

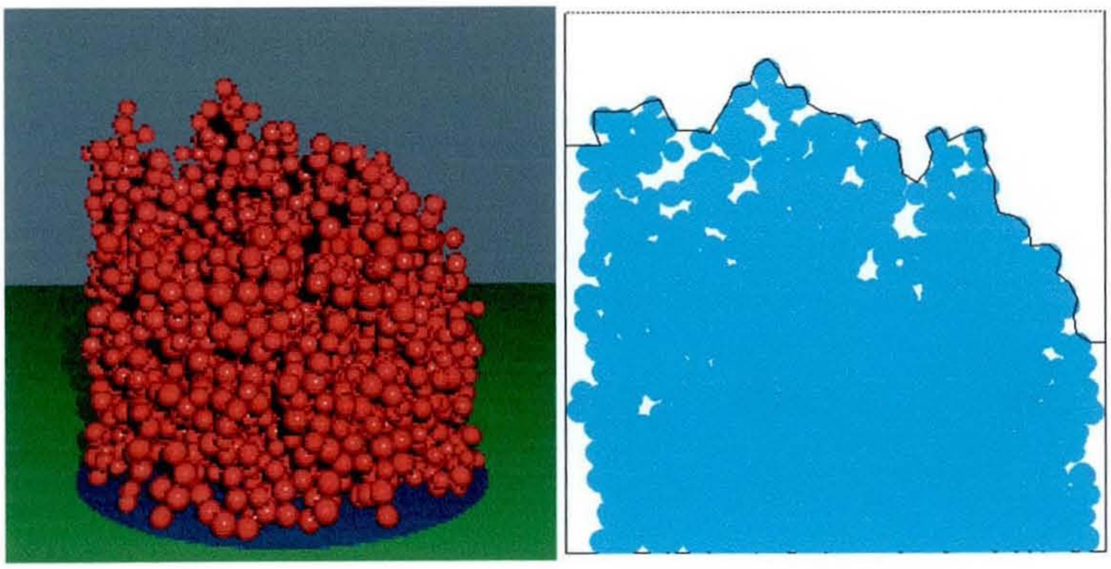


Figure 3.65 – 3-D agglomerate rotated through 120° with respective structured walk measurement (λ = 0.1 F)

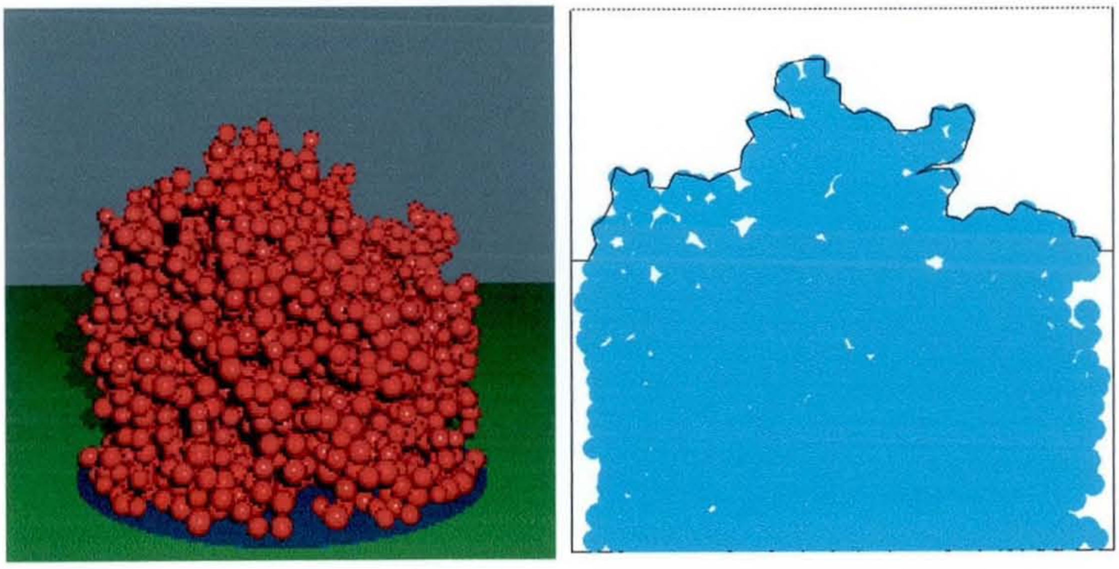


Figure 3.66 – 3-D agglomerate rotated through 240° with respective structured walk measurement (λ = 0.1 F)

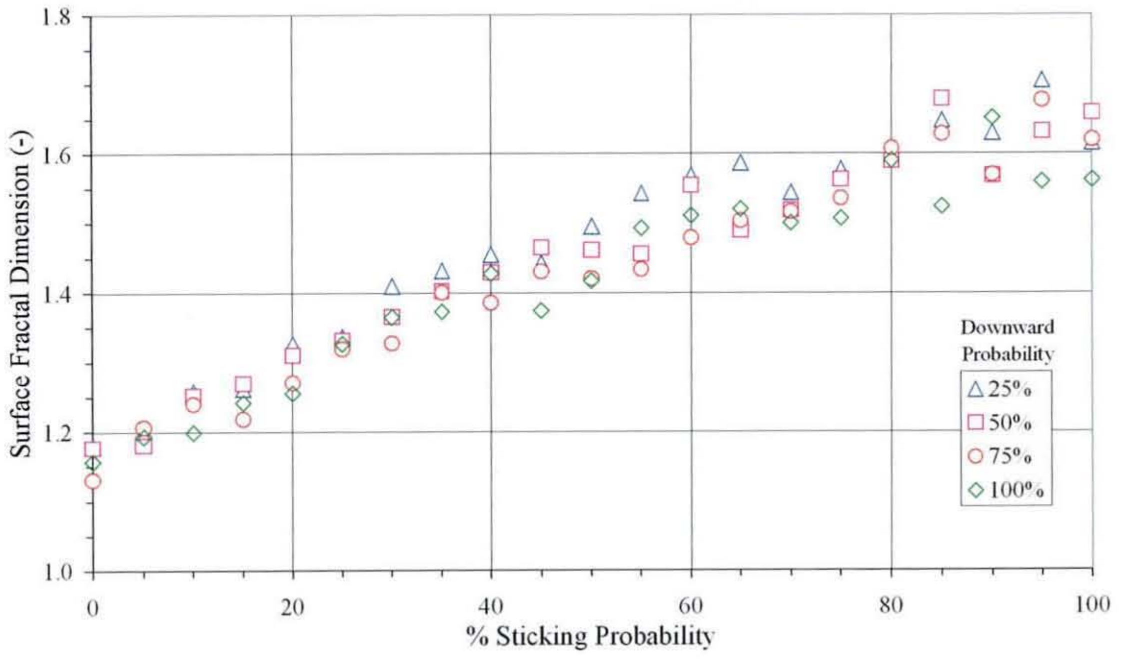


Figure 3.67 - Structured walk analysis of 2-D filtration simulations

A summary of the results obtained from the structured walk analyses of the two-dimensional cakes are shown in Figure 3.67. The data show that the surface fractal dimension of the cakes increased from a minimum of 1.13 (0% sticking probability, 75% downward probability) to 1.66 (100% sticking probability, 50% downward probability).

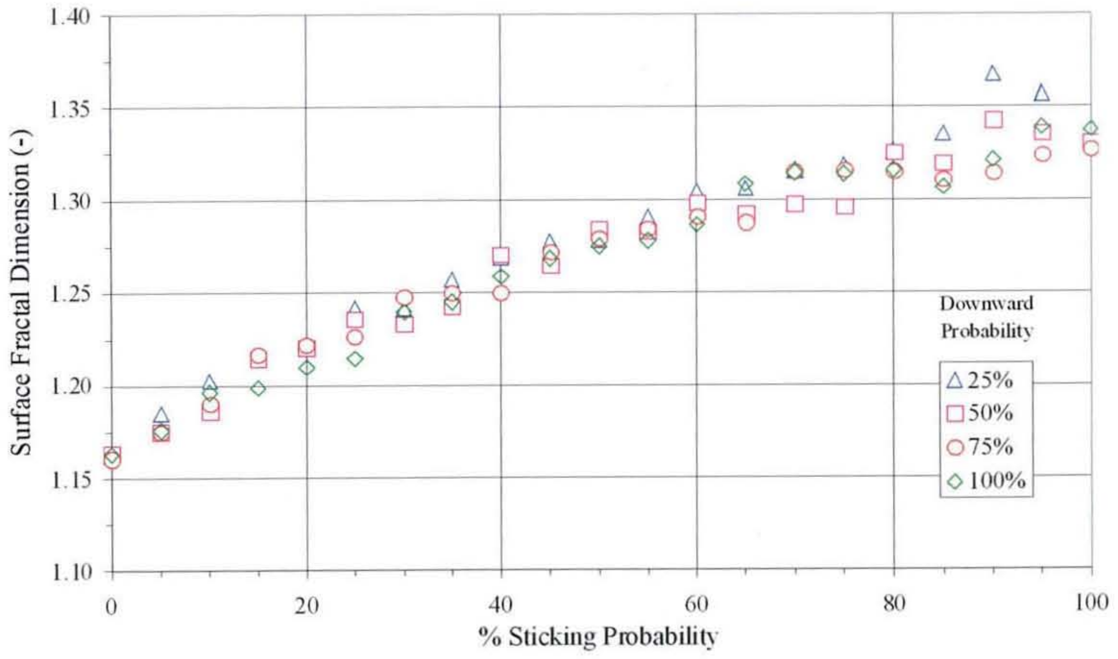


Figure 3.68 - Structured walk analysis of three-dimensional filtration simulations

The results for the surface roughness of the cakes built using the same model in *three* dimensions show the same trends as for two dimensions. Figure 3.68 shows that the surface fractal dimension again increased with the sticking probability, in this case, from 1.16 (0% sticking probability, all downward probabilities) to 1.34 (100% sticking probability, 100% downward probability). In a similar way to the surface roughness of the seed agglomeration models, the change in surface roughness with system parameters was not as noticeable with the three-dimensional cakes as it was with the two-dimensional structures. Once again, the change in the surface fractal dimension is relatively smooth. In fact the change in surface roughness for the three-dimensional cakes, whilst not as marked as for the two-dimensional structures, had less scatter and was more linear.

As opposed to the seed agglomeration models discussed earlier, the change in surface roughness with system parameters was smoother and more linear, in particular the results obtained from the analysis of three-dimensional structures.

3.3.1.1 Relationship between number of particles and fractal dimension

An investigation was also made into the relationship between the number of particles in the filter cake and the surface fractal dimension, in a similar way to the seed agglomerates. In the case of two-dimensional structures, the analysis was carried out on the first 300 particles onwards, to ensure that the base of the filter cell had been sufficiently covered. The interval of growth of the structure was 100 particles, again giving eight values for the perimeter fractal dimensions of the structures. The structures analysed were built with a downward probability of 100% and sticking probabilities of 0, 25, 50, 75 and 100%. The greatest variation in the fractal dimension can be seen for the structures built with 100% sticking probability, although no definite trend is visible. The smallest variation in fractal dimension is seen with the cakes built using a sticking probability of 0%. This is to be expected, as the lower sticking probabilities do not give rise to dendritic structures, which would affect the fractal dimension. Again, there is no obvious trend representing a change in the fractal dimension with variation of the number of particles in the structure.

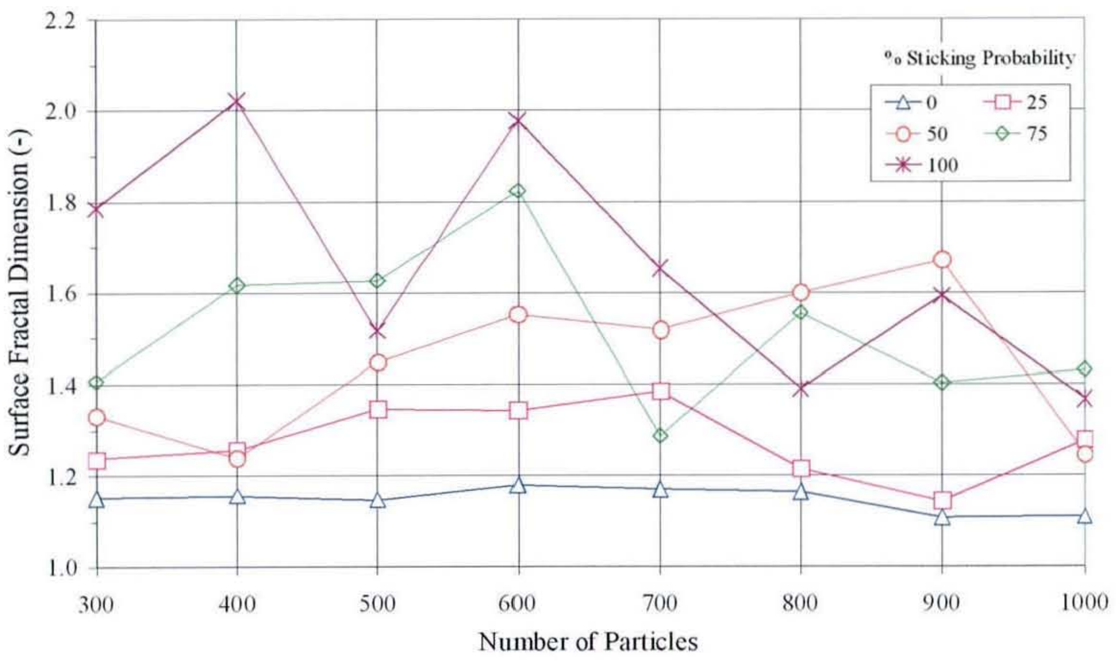


Figure 3.69 - Relationship between number of particles and surface fractal dimension for 2-D filtration simulations

The same analyses were also carried out on the three-dimensional structures, this time taking the first 650 particles as a minimum (again to ensure that the surface of the filter cell was covered) and then steps of 150 particles, up to the 2000 particles that each cake contained. Figure 3.70 shows the results for the surface fractal dimension of these structures. The results for the three-dimensional structures show less variation in the fractal dimension than those for two dimensions, and again no pattern is visible.

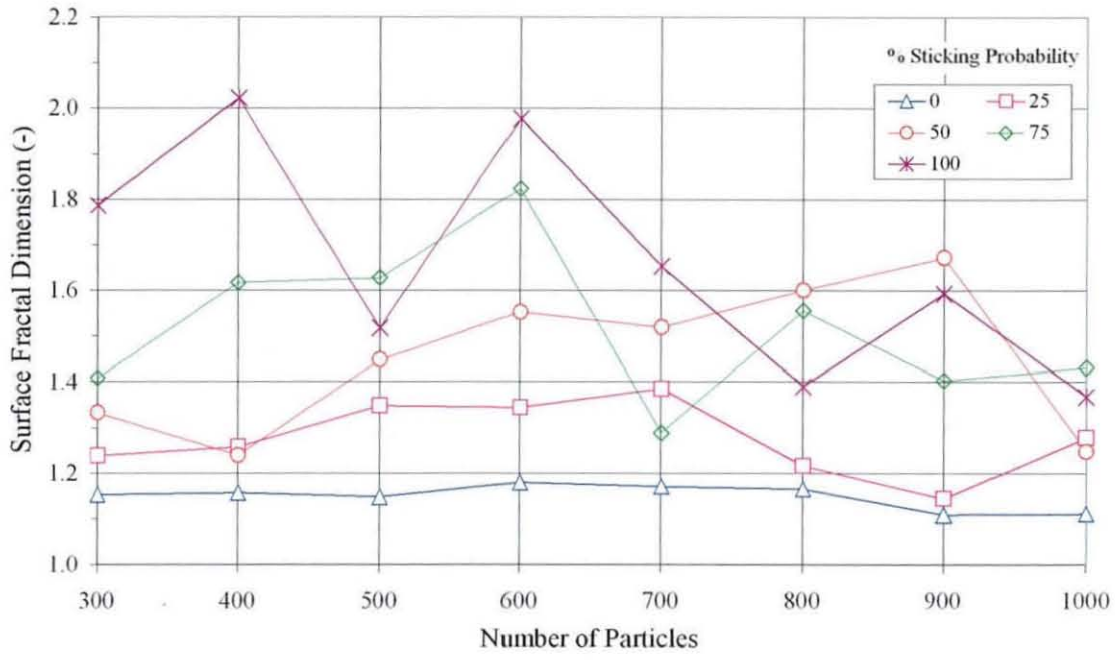


Figure 3.70 - Relationship between number of particles and surface fractal dimension for 3-D filtration simulations

3.3.2 Porosity Measurement

The porosity was measured for all the simulated cakes with specific simulation parameters. The technique used was the vertical porosity profile, whereby the porosity was measured in vertical slices up from the base of the simulated cake to its surface. A weighted average of the results was taken, where the porosity measured closer to the centre of the cake (vertically) was weighted more than the porosity measured at the top and bottom slices of the cake. This weighting allowed a value for the bulk porosity of the cake to be calculated. The results from the porosity analysis are plotted against the downward and sticking probabilities, in the same way as the results for surface fractal dimension.

Figure 3.71 shows this variation of porosity with downward and sticking probabilities for two-dimensional filtration simulation agglomerates. As with the surface fractal dimension, the sticking probability has more of an effect than the downward probability on the cake structure. In the case of porosity, however, the variation in structure with downward probability was much clearer, with porosity decreasing with increasing downward probability.

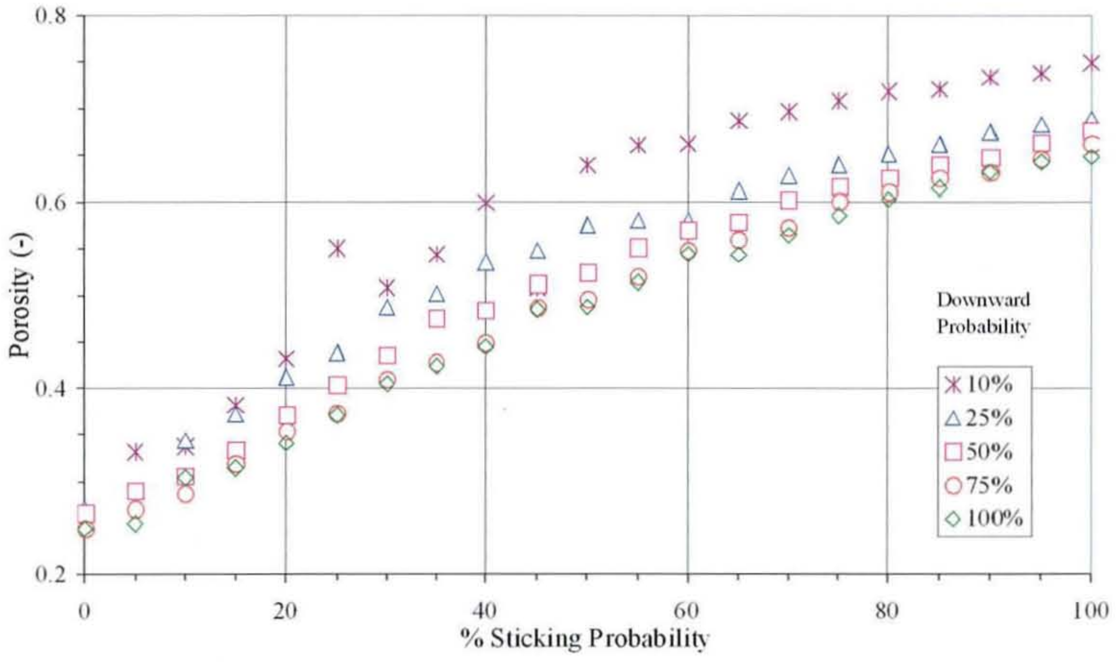


Figure 3.71 - Porosity analysis of 2-D filtration simulations

Other methods for measuring the porosity of the cakes were included in the program, but do not offer any advantage over the method described here. The two other methods (as described in Chapter 2, Figure 2.24 and Figure 2.25) were a second vertical profile and a horizontal, or wall effect profile. Figure 3.72 shows a typical two-dimensional filtration simulation, with the wall effect porosity for the structure shown in Figure 3.73. Only a minor amount of work was performed with regards to this type of porosity measurement, but early indications showed that it could be a viable method of measuring filtration characteristics.

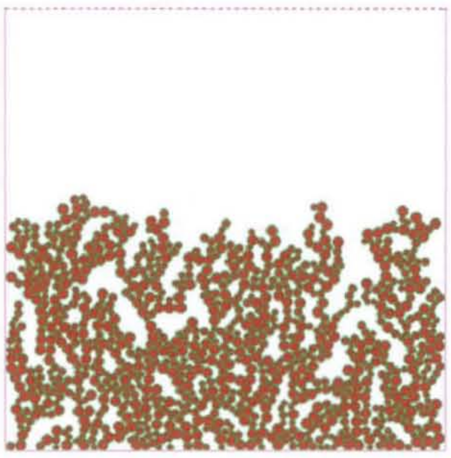


Figure 3.72 – Typical 2-D filtration simulation, 100% Downward probability, 50% Sticking probability

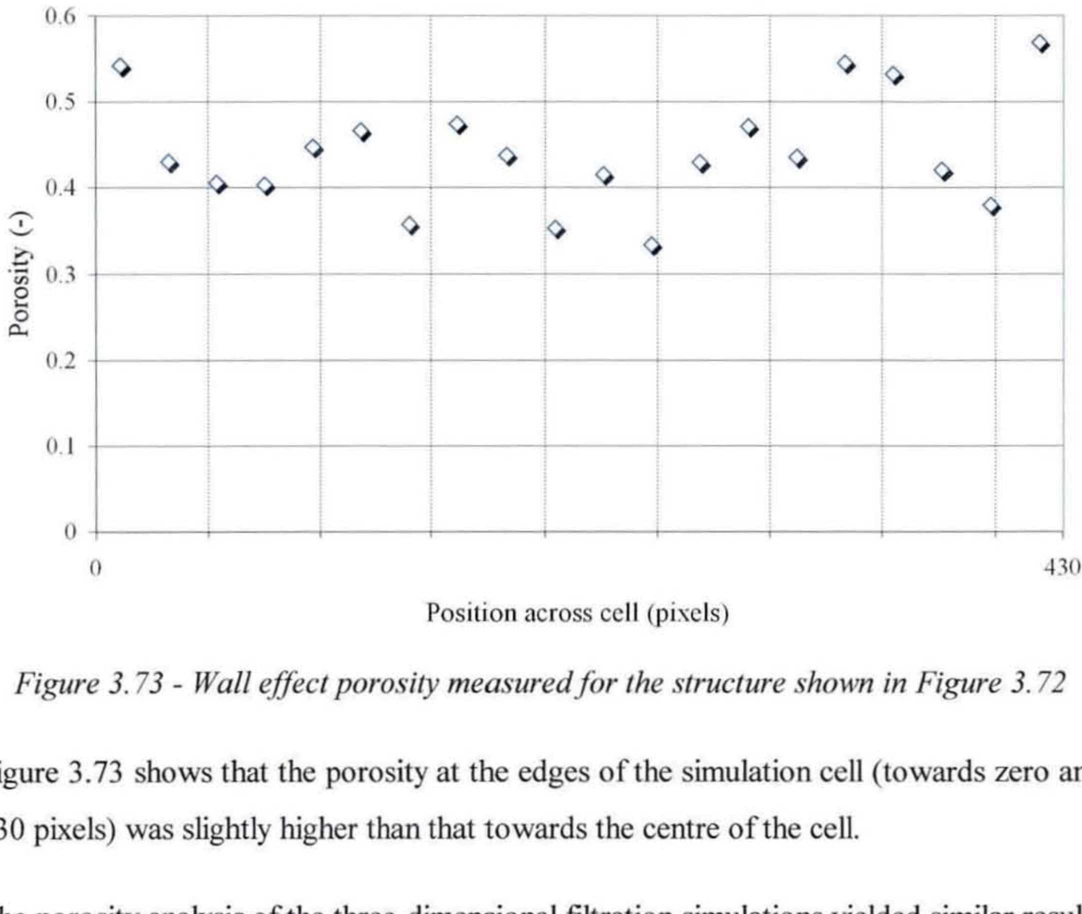


Figure 3.73 - Wall effect porosity measured for the structure shown in Figure 3.72

Figure 3.73 shows that the porosity at the edges of the simulation cell (towards zero and 430 pixels) was slightly higher than that towards the centre of the cell.

The porosity analysis of the three-dimensional filtration simulations yielded similar results to the two-dimensional analysis. Figure 3.74 shows the variation in porosity of the three-dimensional cakes with the sticking and downward probabilities. The difference between the results for two and three dimensions appears to be that the porosity of the three-dimensional cakes increases more rapidly than the porosity for the two-dimensional cakes and then almost reaches a plateau near 100% sticking probability.

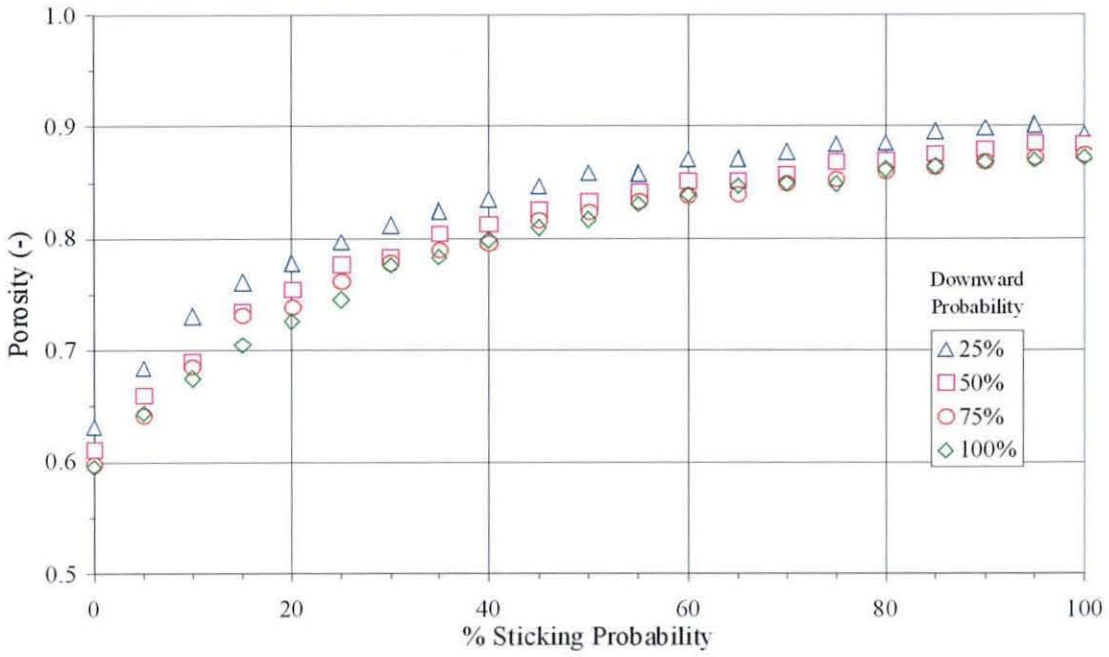


Figure 3.74 - Porosity analysis of 3-D filtration simulations

3.3.3 Relationship between size distribution and structural properties

Further investigations were made using the filtration simulation models to ascertain any affect that the size distribution of particles might have on the properties of the cakes.

For the two-dimensional structures, the size distributions (radius of individual particles) used were: 2, 3 and 5 (monodisperse) and 3 to 5 pixels. Each of the size distributions were chosen arbitrarily to give structures with a significant number of particles, whilst still fitting within the available space and maintaining measurability. The width of the virtual filter cell was 440 pixels, which meant that up to 110 particles of diameter 4 pixels could be dropped across the cell. The parameters chosen for these simulations were a downward probability of 50% with the sticking probability varied from 0 to 100% in 5% increments. These parameters were chosen as previous analysis showed that the structure was more affected by the variation in sticking probability than downward probability. Figure 3.75 shows how the surface fractal dimension varies with the size distribution of particles. Generally, as the size of the particles used in the simulations was increased, the surface fractal dimension also increased.

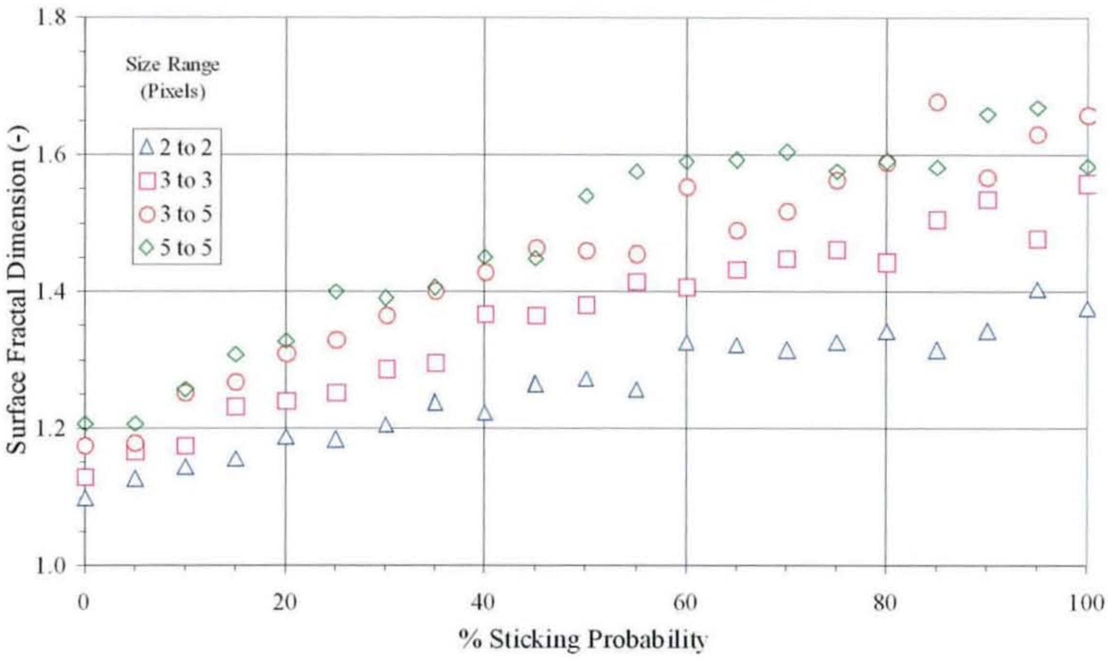


Figure 3.75 - Effect of size distribution on surface fractal dimension for 2-D filtration simulations

Analyses were also performed to determine the effect of size distribution on the porosity of the filtration simulations. Figure 3.76 shows the variation in porosity with particle size distribution for the same simulations as Figure 3.75. Again, it can be seen that the size distribution of the particles has an effect on the porosity of the cakes under scrutiny. In this case, the porosity tends to increase with decreasing particle size, although the boundaries are not clear between the 3, 5 and 3 to 5 pixel radius size distributions, especially at the higher sticking probabilities (> 60%). This is possibly because at the high sticking probabilities, when the cakes took on an open structure, the porosity is more governed by particles simply occupying the space rather than showing marked differences due to particle packing at low sticking probabilities (giving more closed structure to the cakes).

These results are in contrast to the results shown for the surface fractal dimension (Figure 3.75) where the variation in cake properties was less noticeable at low sticking probabilities than at high sticking probabilities.

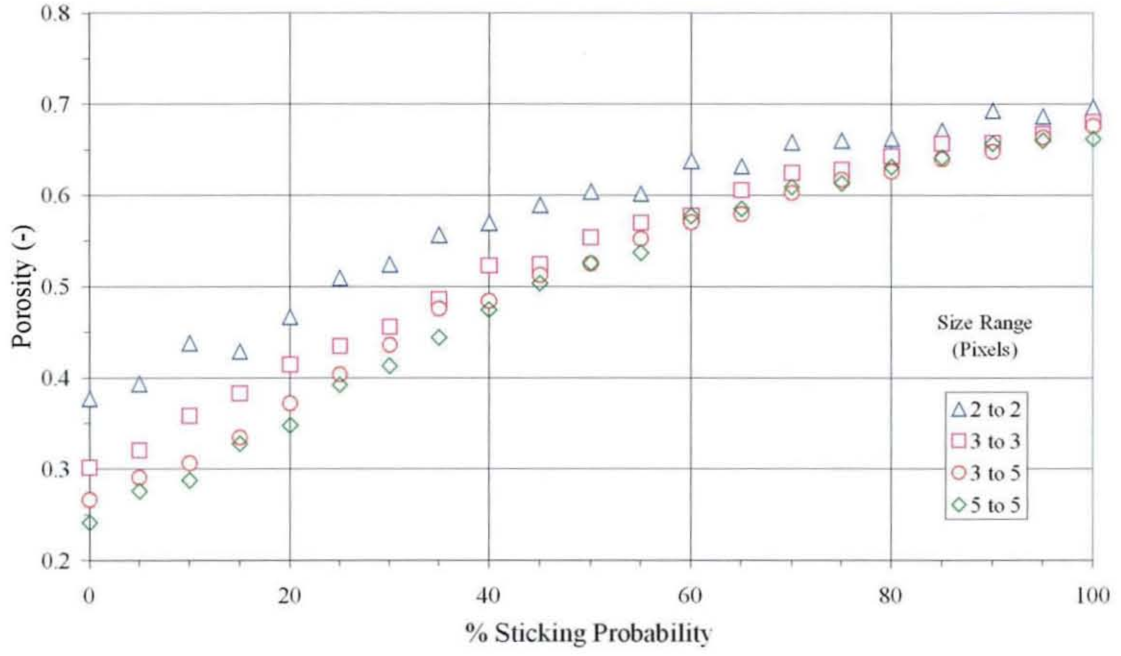


Figure 3.76 - Effect of size distribution on porosity for 2-D filtration simulations

Each of the points in both Figure 3.75 and Figure 3.76 represent the average property (either porosity or surface fractal dimension) for twenty simulations. In the case of the surface fractal dimension, each point also represents the fractal dimension averaged in two directions (left to right and right to left) as in the previous analyses.

There appears to be more scatter at the higher sticking probabilities for the surface fractal dimensions compared to the porosity measurements. This is most likely due to the dendritic nature of the structures involved. In the case of the porosity analysis, the porosity values for the structures built using lower sized particles look to be more porous than those built with the larger particles. The scatter at the lowest values of sticking probabilities are most likely due to the measurement techniques. As the porosity measurement has a resolution of one pixel, and the particles are only four pixels in diameter, when the particles are closely packed (low sticking probabilities), errors are more likely to occur. This is, unfortunately, a limitation in the program.

The same type of investigation was carried out for the three-dimensional agglomerates. A downward probability of 50% was again chosen for the simulations, but with different size distributions from the two-dimensional analyses. The sizes chosen were an 8 pixel monodisperse system and a range of 3 to 12 pixels diameter. For the particles with a radius of 8 pixels, the base of the filter cell could be covered with a *theoretical* maximum of 722 particles (cross-sectional area of filter cell / cross-sectional area of particle). This value is not the actual maximum number of particles that could pack the cell, as the voidage is not included. The second size range was perhaps the most useful, as it offered a wider range of particle sizes, as would occur in a physical environment. This range offered a range of between 321 and 5136 particles over the area of the cell, again discounting the voidage of the system. Larger particles were chosen for the three-dimensional analyses, as more particles can fit into the filter cell, and higher accuracy can be achieved when analysing systems with larger particles, due to the mathematics involved.

As with the previous simulation, both the surface fractal dimensions and porosities of the simulations were measured at a range of sticking probabilities (0 to 100% in 5% increments).

Figure 3.77 shows the variation in surface fractal dimension with size distribution. Unlike the surface fractal dimensions of the two-dimensional simulations shown in Figure 3.75, the fractal dimensions of three-dimensional simulations appear not to follow similar trends.

Whilst there is some variation in the fractal dimension with size range (notably the eight to ten pixel size range), the data cross each other and offer no conclusive evidence of the influence of size distribution on surface roughness. As with previous results from the three-dimensional analyses, this is possibly due to the fact that the measure is only of two-dimensional projections of three-dimensional objects.

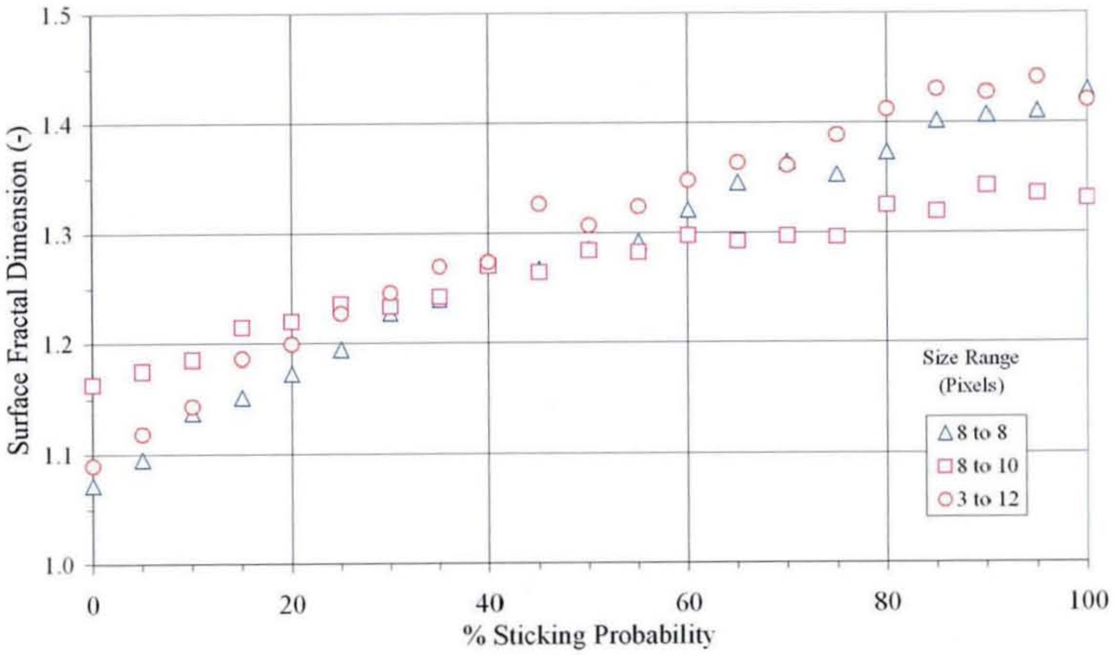


Figure 3.77 - Effect of size distribution on surface fractal dimension for 3-D filtration simulations

Figure 3.78 shows how the porosity varies for the same simulations. As with the surface roughness, it would appear that the size distribution of the constituent particles has little effect on the internal structure if the simulated filter cakes. Unlike the surface roughness results, however, the data is much closer grouped, with no one distinct group of data.

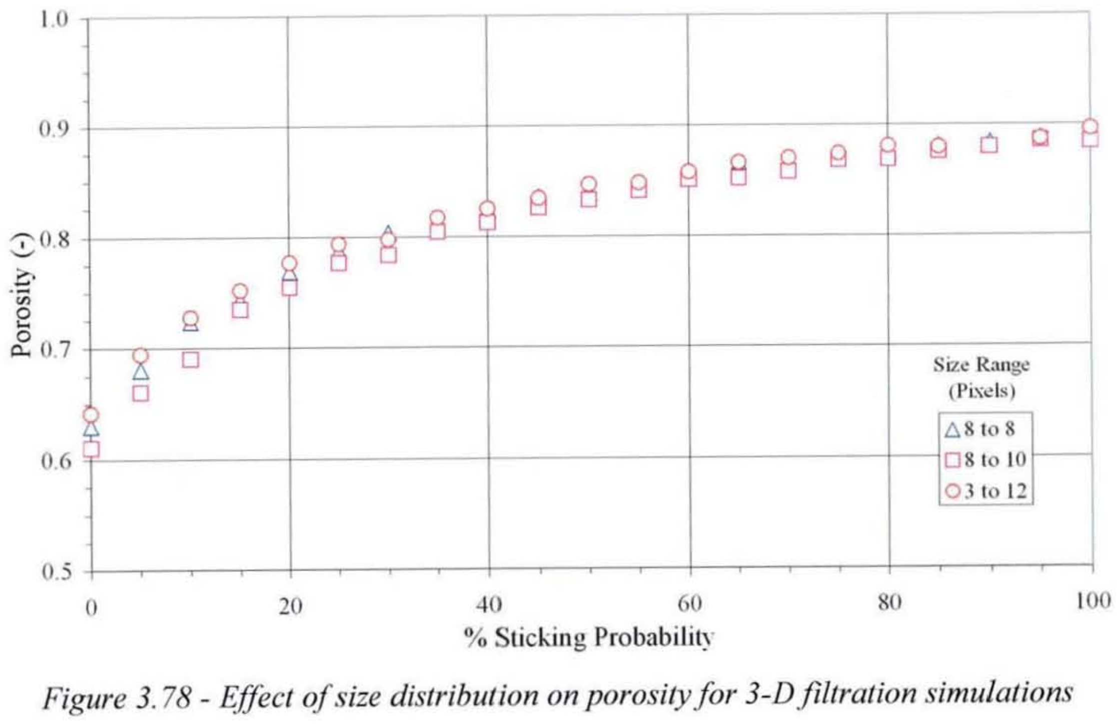


Figure 3.78 - Effect of size distribution on porosity for 3-D filtration simulations

3.3.4 Conclusions

It can be seen from Figure 3.67 and Figure 3.68 that the downward probability has some effect on the surface structure of the cake, although less than the effect of sticking probability. As a general trend, the surface ruggedness of the cake decreased as the downward probability was increased.

The downward probability had more of an influence on the measured porosity of a cake, as can be seen from Figure 3.71 and Figure 3.74, although still not as much of an influence as the sticking probability.

The analysis of both the two and three-dimensional cakes showed that the downward and sticking probabilities both had a visible effect on the physical properties of the simulated cakes. As a general guide, as the sticking probability was increased, the cake became more open (less dense) and had a more rugged surface. The opposite is true of the way in which varying the downward probability altered the structure of the cakes. As the downward probability was increased, the structure of the cake became less open and the surface less rugged, although the downward probability had less of an influence on the cakes' structure than the sticking probability.

The porosity values obtained match well with both literature values for maximum packing density [Gray, 1968] and experimental results for the porosity of cakes formed from the pressure filtration of mineral suspensions (see Chapter 1).

3.4 Analysis of interstitial spaces

This project has concentrated on the analysis of the overall structure of the simulations created (both seed agglomeration and filtration simulations), however, the program also included the facility for analysing the interstitial spaces within the two-dimensional filtration simulations. These spaces could be isolated and analysed using both structured walk (perimeter) and enclosing circle fractal measurements.

3.4.1 Isolation of the interstitial spaces

The technique used for isolating the individual pore spaces is described in Chapter 2. Once the pore spaces are isolated, their pertinent information (maximum and minimum Feret diameters – length and position, centre of gravity and the position of point furthest from the centre) was saved in a separate file for easy retrieval. Procedures were also in place to ensure the pore spaces were valid (i.e. actual spaces, not solid material, completely enclosed by boundaries other than the cell walls).

In general, groups of interstitial spaces were analysed, in the same way as was possible for the simulations themselves. On average, 25 pore spaces were created for each pair of simulation parameters (downward and sticking probabilities) analysed. These spaces were always obtained from more than one simulation.

3.4.2 Fractal dimensions of interstitial spaces

3.4.2.1 Surface fractal dimension

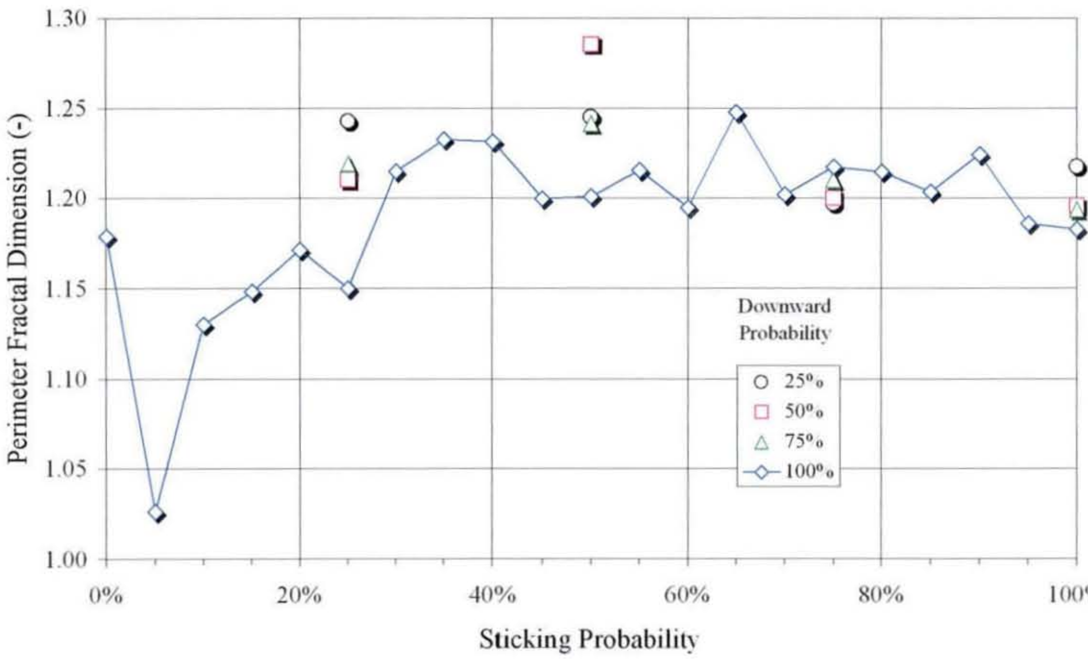


Figure 3.79 – Perimeter fractal dimensions for interstitial spaces taken from 2-D filtration simulations

Figure 3.79 shows the structured walk fractal dimensions for interstitial spaces taken from two-dimensional filtration simulations. A full set of data is shown for 100% downward probability, whilst only limited data are shown for 25, 50 and 75% downward probability. Full data are not shown for all of the simulations, as no trends with variation in downward probability were visible for any of the data sets using the results shown in Figure 3.79.

The only noticeable trend in the data for pore spaces taken from simulations constructed using 100% downward probability is that the surface fractal dimension tended to decrease as the sticking probability neared zero. The possible reasons for this trend are explained later in this section.

3.4.2.2 Density fractal dimension

Figure 3.80 shows the density fractal dimension results (measured using the enclosing circle technique) for the same pore spaces analysed in Figure 3.79. In the case of the density fractal dimension, the change of fractal dimension with sticking probability was more visible, with less variation than that for surface fractal dimension. Again, no particular trends were visible for the change in density fractal dimension with downward probability.

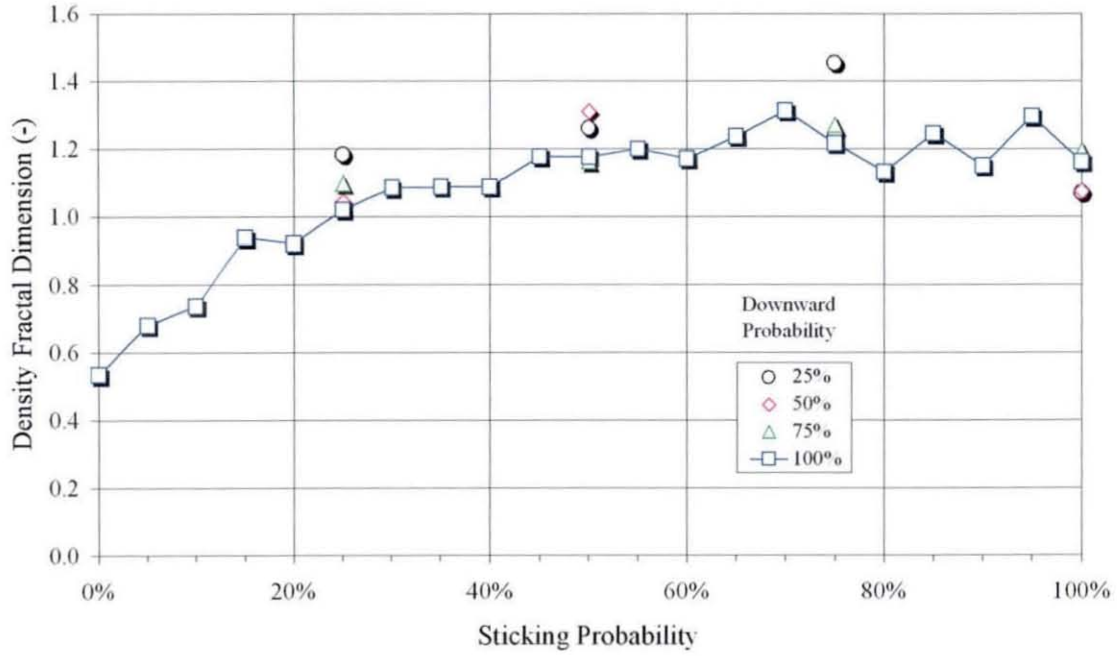


Figure 3.80 - Density fractal dimensions (enclosing circle technique) for interstitial spaces taken from 2-D filtration simulations

It can be seen in Figure 3.80 that the density fractal dimension is calculated as less than 1.0 for simulations constructed with low sticking probabilities. These low values for the fractal dimension are caused by the fact that the average size of the interstitial spaces for these simulations was very low. The fractal dimensions were calculated by determining the mass within using twenty expanding circles. As the areas of the pores was generally low, the routine for calculating the enclosed mass used fractions of pixels for the radius of the circles (which could not effectively be measured by a computer), resulting in the mass not increasing with increasing radius, as would normally have been the case. Further explanation of this situation is given in Section 3.4.3.

3.4.3 Other trends within filtration simulation structures

During the analysis of the interstitial structures, it was noticed that the average pore size within the structures decreased with decreasing sticking probability. This was an expected occurrence, as it had previously been noted that the openness of the structures decreased in this manner. The variation in average pore size with sticking probability (for simulations constructed with 100% downward probability) is shown in Figure 3.81.

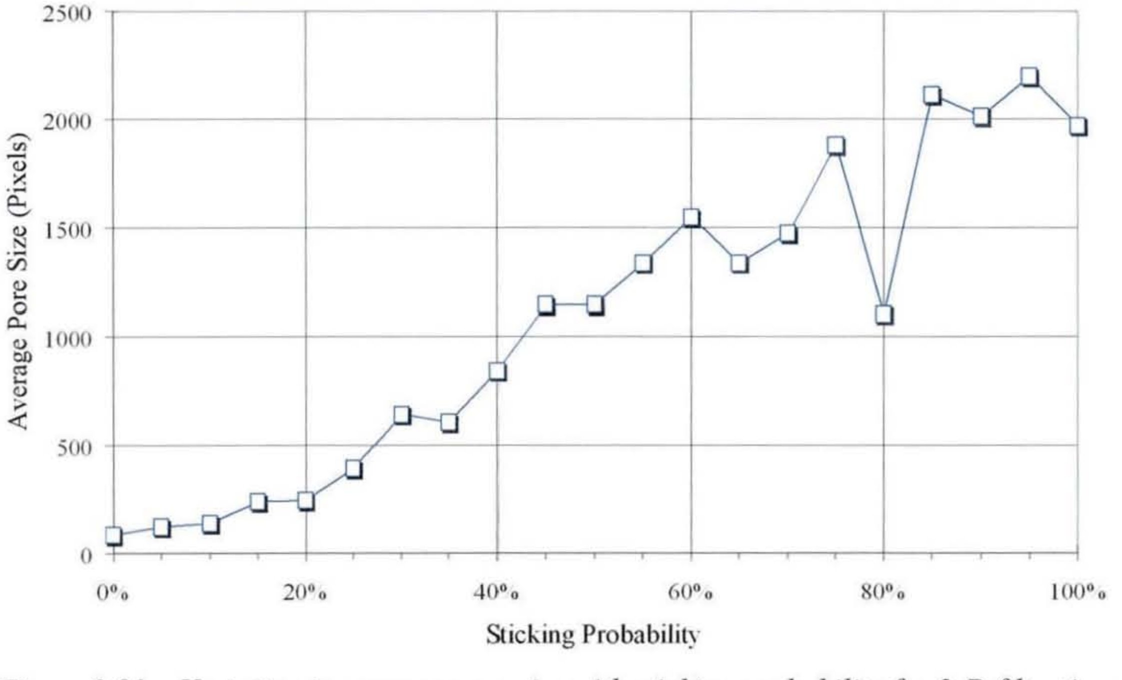


Figure 3.81 – Variation in average pore size with sticking probability for 2-D filtration simulations

It is obvious from Figure 3.81 that the pore size changes dramatically with the simulation parameter previously shown to have the greatest effect on simulation structure.

If the two fractal measures are plotted against pore size, it can be seen that this was probably the main influence on fractal dimension rather than actual fractal properties. This dependency was more noted as the sticking probability neared zero and the average size of the pore spaces decreased accordingly. As the pore sizes neared their minimum, the scale of scrutiny became too large compared to the size of the object and accuracy in fractal measure would have been lost. This is borne out by the fact that the density fractal dimensions for those simulations constructed using sticking probabilities below 25% were less than one (impossible under the earlier definition of the dimension). Figure 3.82 shows the variation with average pore size of the structured walk fractal dimension, whilst Figure 3.83 shows the same variation for the enclosing circle (density) fractal dimension. The size ranges shown in these two graphs are independent of the sticking probability.

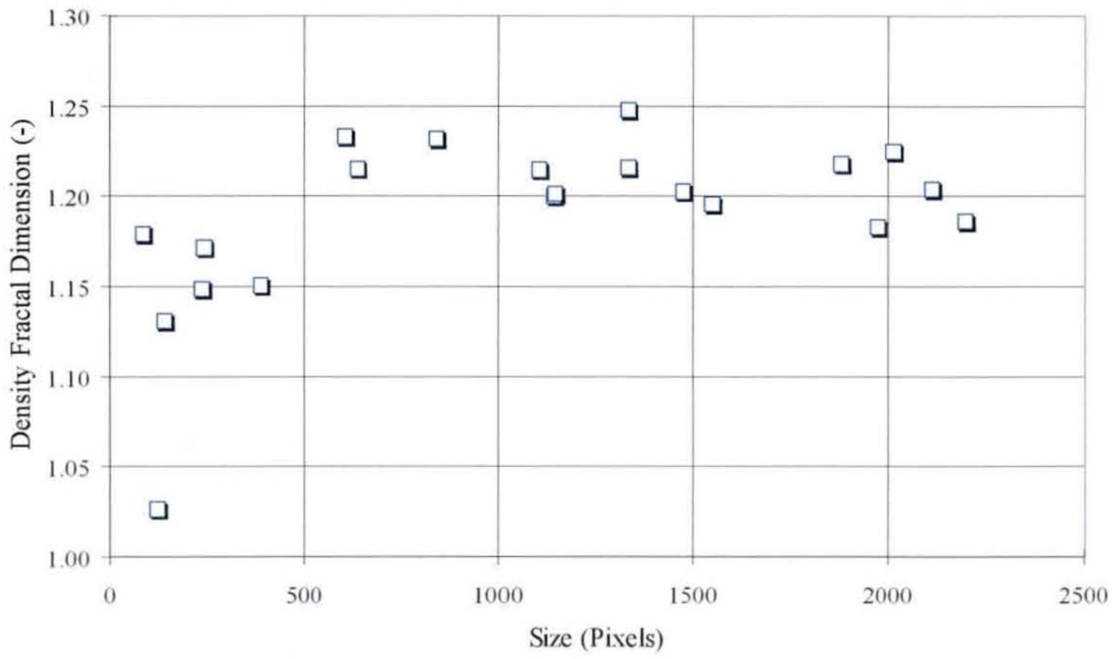


Figure 3.82 - Variation in surface fractal dimension with average pore size

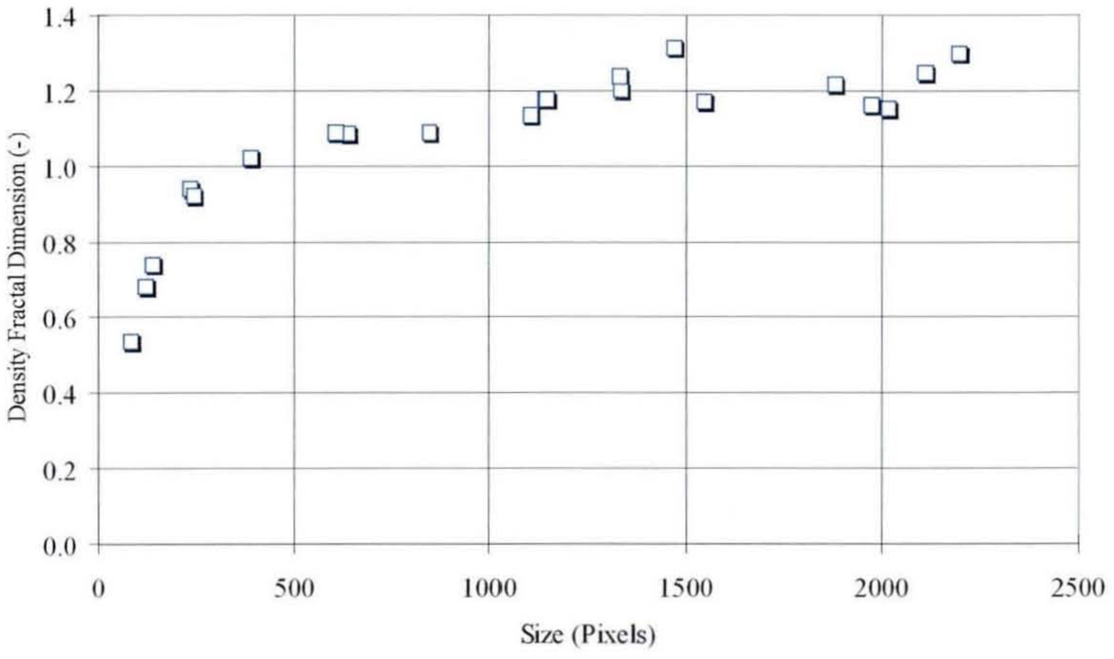


Figure 3.83 - Variation in density fractal dimension with average pore size

3.4.4 Conclusions

It would appear from the work conducted as part of this research that no direct link has been proven between simulation parameters and interstitial pore space fractal dimensions. However, there now exists a rigorous and repeatable method to isolate and measure the characteristics of these spaces in the future.

3.5 Variation in measured results

Some degree of variation did occur within the measured parameters for the structures under scrutiny in this chapter.

Figure 3.84 shows the scatter within a typical set of data. This particular figure shows the variation in surface fractal dimension for two-dimensional seed agglomerates. The data shows an average fractal dimension of 1.289 with a variance of 0.004 and standard deviation of 0.067. It can be seen from Figure 3.84 that the majority of data (approximately 83%) lies within one standard deviation of the mean.

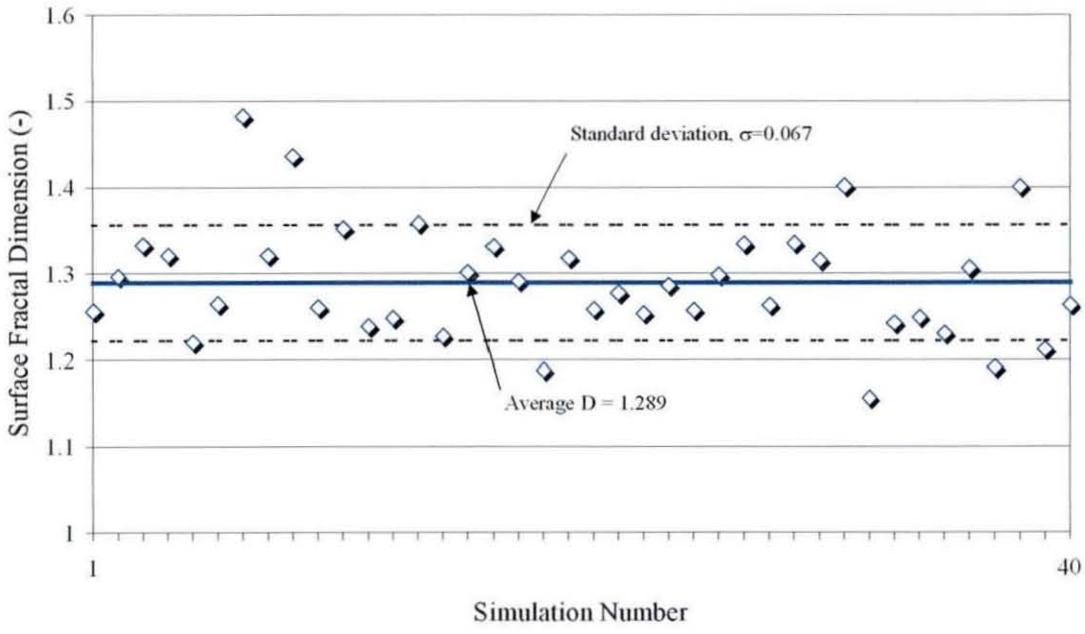


Figure 3.84 - Scatter within a single data set (50% diffusion influence)

Figure 3.85 shows the variance of all the data in the set to which the data in Figure 3.84 belongs. It can be seen that the majority of data sets (76% or 16/21) have variance below 0.01 with all variances for the measurement in question remaining below 0.03.

From the variances recorded, it may be assumed that the measurements performed on the structures were statistically significant, and could therefore be taken as a true indication of the variation of fractal dimension with construction parameters.

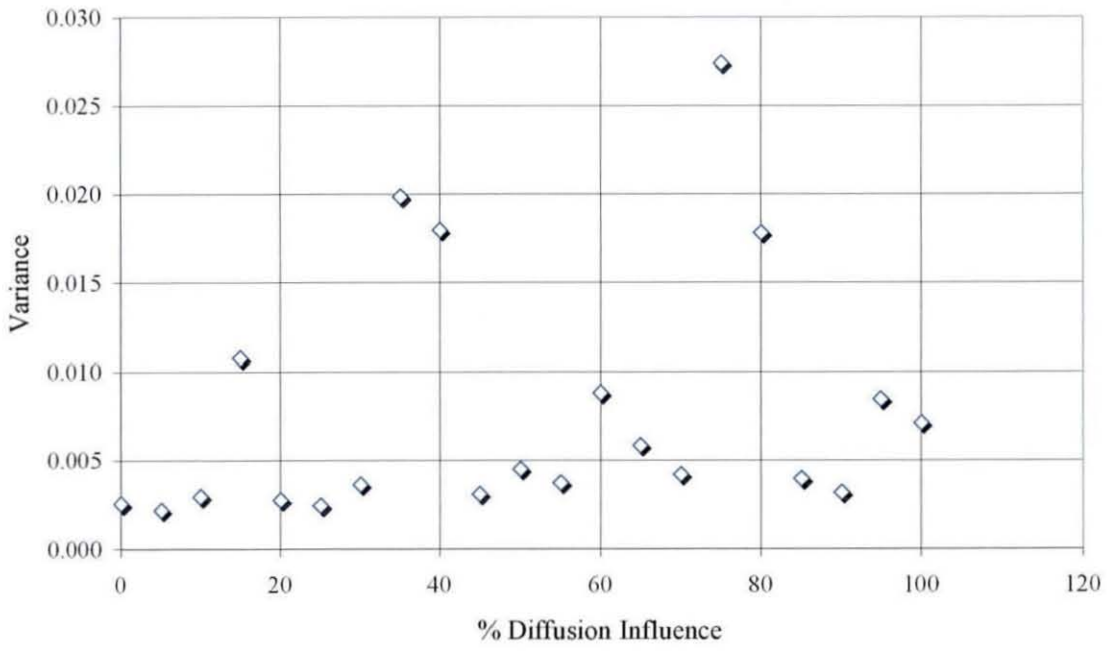


Figure 3.85 - Variance in measured fractal dimension with diffusional influence

3.6 Overall conclusion of results from simulated structures

It has been shown that the method by which simulated structures are built, be it via seed agglomeration or filtration simulation, in two or three dimensions, had an influence on both the physical and fractal properties of the structure.

Generally, as the motion of the particle moved away from a ballistic model towards a diffusive model, the structure became more open (less dense) and had a rougher surface or perimeter. In the case of the filtration simulations, a high downward probability can be thought of as ballistic motion of the descending particle, whereas a low downward probability can be thought of as a more diffusive movement of the particle.

It has also been proved that the mechanisms used for particle capture, once the moving particle intercepted the growing structure have an effect, albeit lesser on the structure of the simulation. If the moving particle was allowed to reach a state of lower entropy, the structure was generally shown to have had a higher density than if the moving particle stuck on contact.

4 EXPERIMENTAL WORK

The aim of the experimental work was to obtain filtration data and construct filter cakes that could be subsequently examined in order to characterise their internal structure. A dead-end filtration system was chosen, thus offering the most convenient method of filter cake sampling.

Meeten [1993] suggested a method for the analysis of filter cakes. Here, a number of different spatial properties of filter cakes were measured, but it was not possible to examine the internal structure of the cake, as the technique was destructive and involved slicing the cake whilst still 'wet'. The method involved pushing a formed cake up a cylinder using a piston, and slicing the cake into sections of uniform thickness. These were subsequently analysed for properties such as voids ratio and fluid mass. For the work described here it was necessary to examine the internal structure of a cake, thus it was necessary to make every effort not to disturb the constituent particles.

Schmidt and Löffler [1990, 1991] and Schmidt [1995] used an alternative technique to examine the internal structure of dust cakes formed in a bag filter. In order to obtain a sample of the dust cake for analysis, it was first necessary to stabilise the entire cake before embedding it in a permanent, hard-setting resin. Vaughan and Brown [1996] have also previously used a similar method to examine the structure of fibrous filter materials.

The method used for preserving the structure of filter cakes in the current work followed the basic procedures of Schmidt and Löffler [1990,1991], Schmidt [1995] and Vaughan and Brown [1996]. The only major difference was the manner in which the epoxy resin was introduced. In the previous work, the material being analysed was either placed on a porous medium and the epoxy resin allowed to absorb upwards through the sample via capillary action, or a slow 'drip' method used to embed the sample. Neither of these was suitable for the embedding of filter cakes used in the current work, due to the depth of the filter cakes and relatively low porosity.

4.1 Experimental apparatus

The apparatus comprised of two distinct sections. The first part was the pressure filtration rig used to filter the suspensions, thus forming filter cakes. The second part was used to sample and fix a cake structure prior to sectioning and subsequent examination.

4.1.1 Pressure filtration apparatus

A schematic of the pressure filtration rig is shown in Figure 4.1, whilst pictures of the rig itself can be seen in Figure 4.2 and Figure 4.3. Figure 4.2 gives an overview of the apparatus as it was set up to perform the filtration experiments.

The equipment comprised a sealed stainless steel vessel, with various inlet and outlet lines attached. Inside the vessel was a sampling probe, which consisted of a tube connected to a stainless steel rod (see Figure 4.4). Figure 4.3 shows the inlet funnel on the left-hand side of the photograph, with the sampling rod exiting the vessel in the centre of the photograph. Also visible in this photograph are the air lines used to pressurise and de-pressurise the vessel during the experimental procedure.

The pressure of the compressed air in the system was controlled by the filter-regulator. Once set, the pressure of the system was left constant throughout the experiment (i.e. to give constant pressure filtration). The filter was connected to a data-logging PC, which received data from an electronic balance as the experiment proceeded. The balance, coupled with the PC, measured filtrate volume with time, which, along with other physical data, allowed the characteristic properties of the cake, such as specific cake resistance to be calculated.

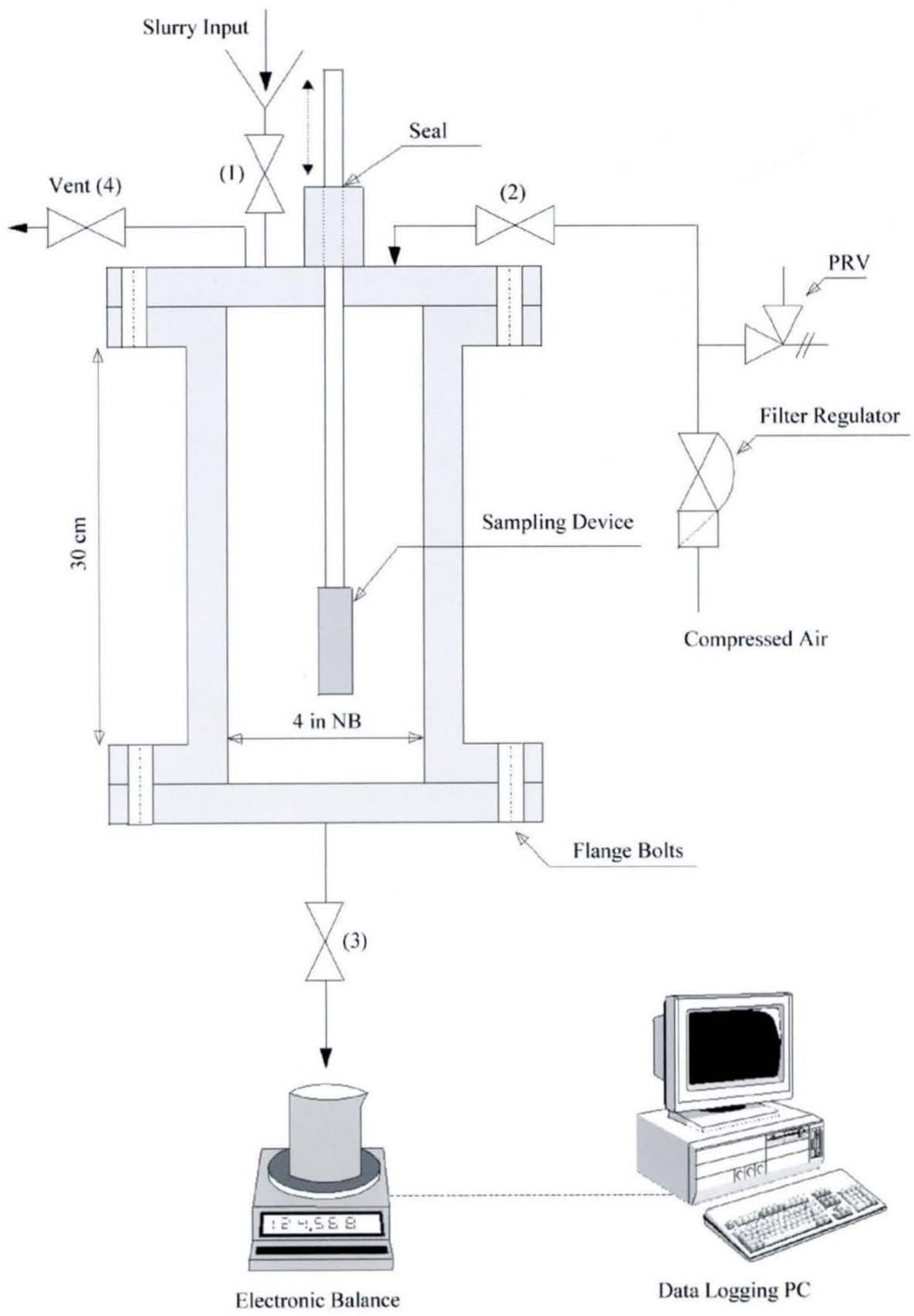


Figure 4.1 - Pressure filtration apparatus

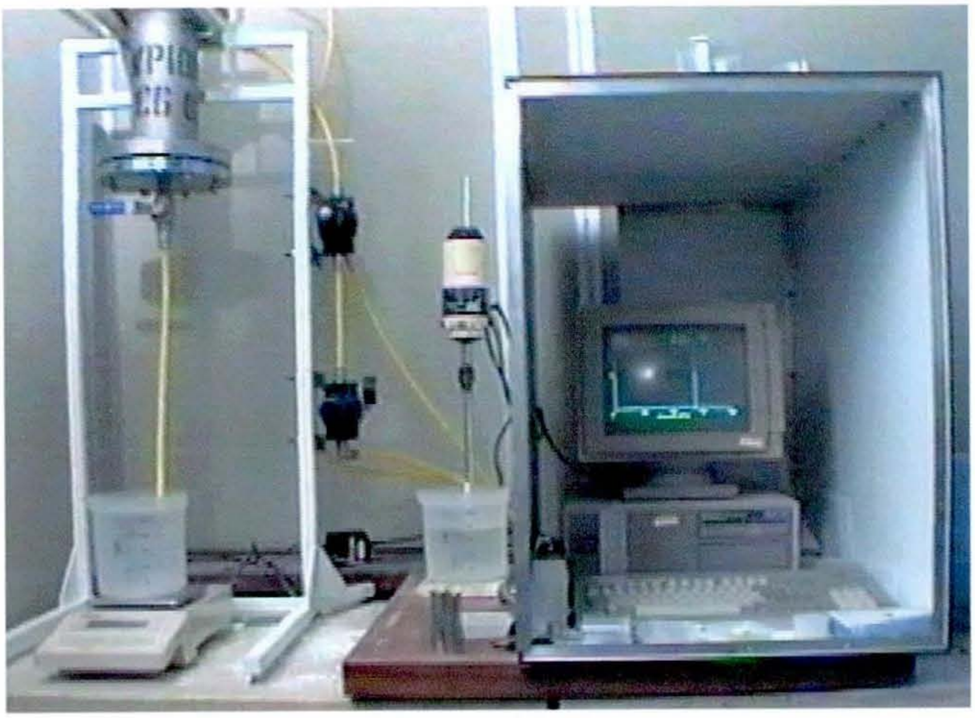


Figure 4.2 – Filtration apparatus with the balance shown connected to data-logging PC

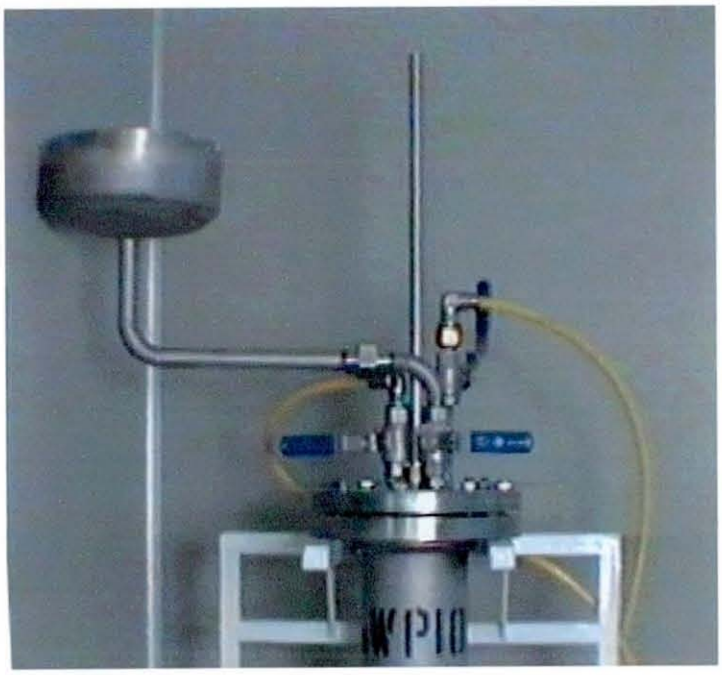


Figure 4.3 – Close-up of the top section of the filtration rig

The probe used to sample a filter cake was at the end of the rod that can be seen in the centre of the vessel in Figure 4.3. Figure 4.4 shows a close up of this, which highlights the sharp leading edge of the sample probe and screw thread used to discharge the cake sample. It should be noted that the sample probe was in place inside the vessel throughout the duration of each experiment and was pushed into the filter cake at the bottom of the filtration cell without opening the cell, therefore integrity of the cake thus ensured.

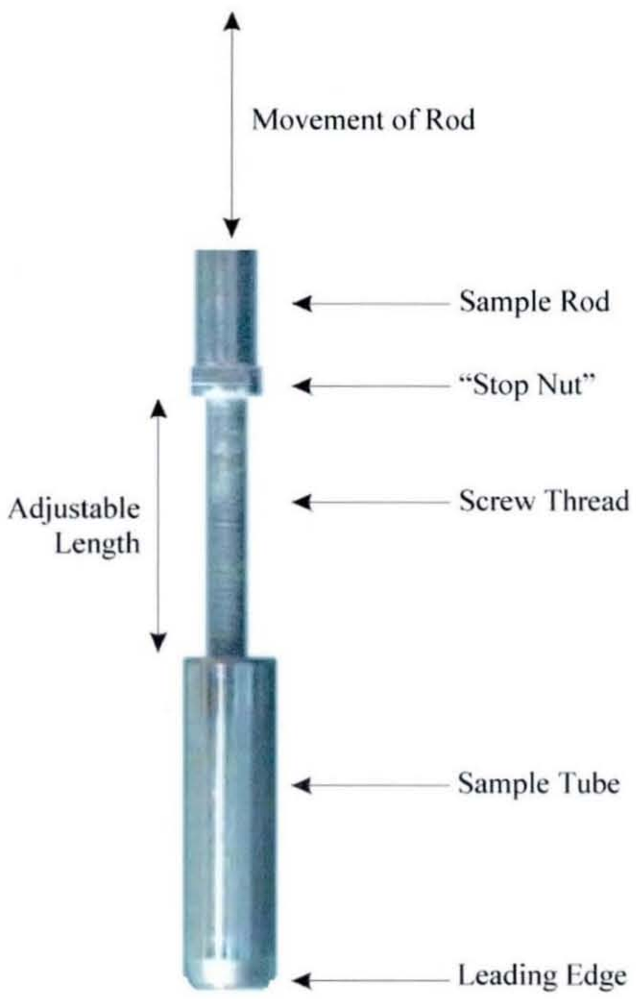


Figure 4.4 – Detail of the sample probe shown in the fully extended position, ready to take a sample of filter cake.

The probe was designed with sharp leading edges in order to cause the minimum amount of disruption to the particles in a cake. Attached to the base of the rod was a disc, which was used to push the sample out of the container with a minimum of disturbance to its structure. The disc was designed to rotate freely at the base of the rod to further ensure that the cake was not disturbed by any rotational motion (caused by the unscrewing of the probe) whilst being ejected. This disc is shown in Figure 4.5, which would have been the normal state of the probe after a sample had been ejected. It can also be seen in

Figure 4.5 that the disc and sample probe were detachable via a set of grub screws, enabling both to be replaced or the size of probe changed. The size of the sample probe used in the experiments was 1 cm diameter. Whilst a larger size of tube was constructed (2.5 cm diameter), it was found that this size gave the best, sample, both in terms of being most representative and of a size that could be managed and prepared by the method described later in this chapter.

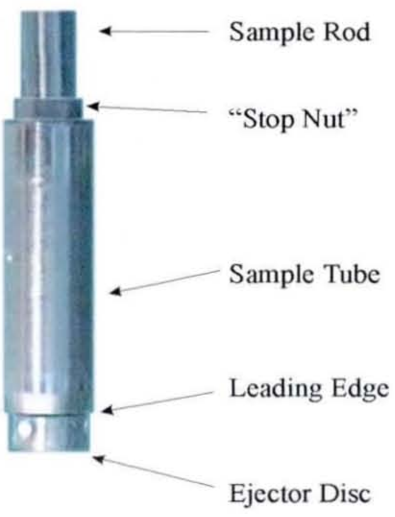


Figure 4.5 – Detail of sample probe showing the ejector disc and grub screws used to keep the disc in place on the rod whilst allowing it to turn freely.

The sample rod was sealed at the top of the vessel by a rotating seal, which allowed the rod to move up and down, without losing pressurisation. The rod was long enough to be able to push all the way through to the bottom of the cake (at the base of the filter cell). This also meant raising the filtration cell up from the level of the laboratory surface in order to remove the rod with the sample intact.

After the cake had been sampled using the sample tube shown in Figure 4.4 and Figure 4.5, a second tube (shown in Figure 4.6) was used to mount the cake in the second apparatus for temporarily setting the cake before embedding in epoxy resin. The tube shown in Figure 4.6 was the same size as that in Figure 4.4, but the leading edge was inward rather than outward sloping, thus enabling the two ends of the tubes to come together perfectly, so that once again the particles in the cake sample were disturbed to a minimum degree. The arrangement of the two tubes is show in Figure 4.7.

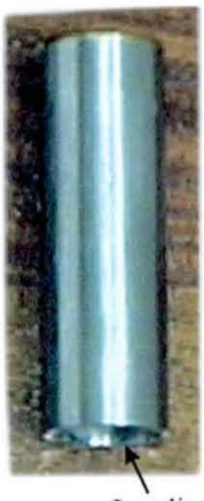


Figure 4.6 – Detail of the second sample tube, designed to be held in the apparatus used to temporarily set the cake prior to embedding in epoxy resin

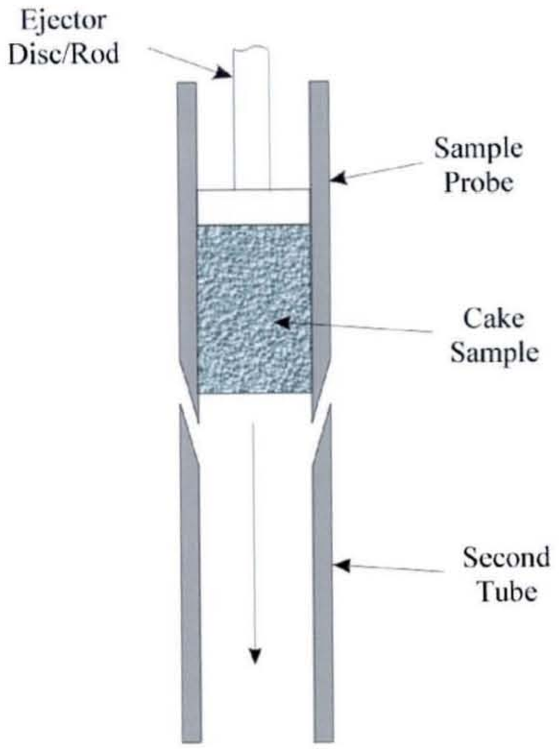


Figure 4.7 – Schematic showing the cake sample being pushed from original sample probe into the second sample tube

4.1.2 Cake fixing apparatus

The second of the two pieces of apparatus was the equipment used to temporarily set the filter cake before final embedding and sectioning. A schematic of this apparatus is shown in Figure 4.8.

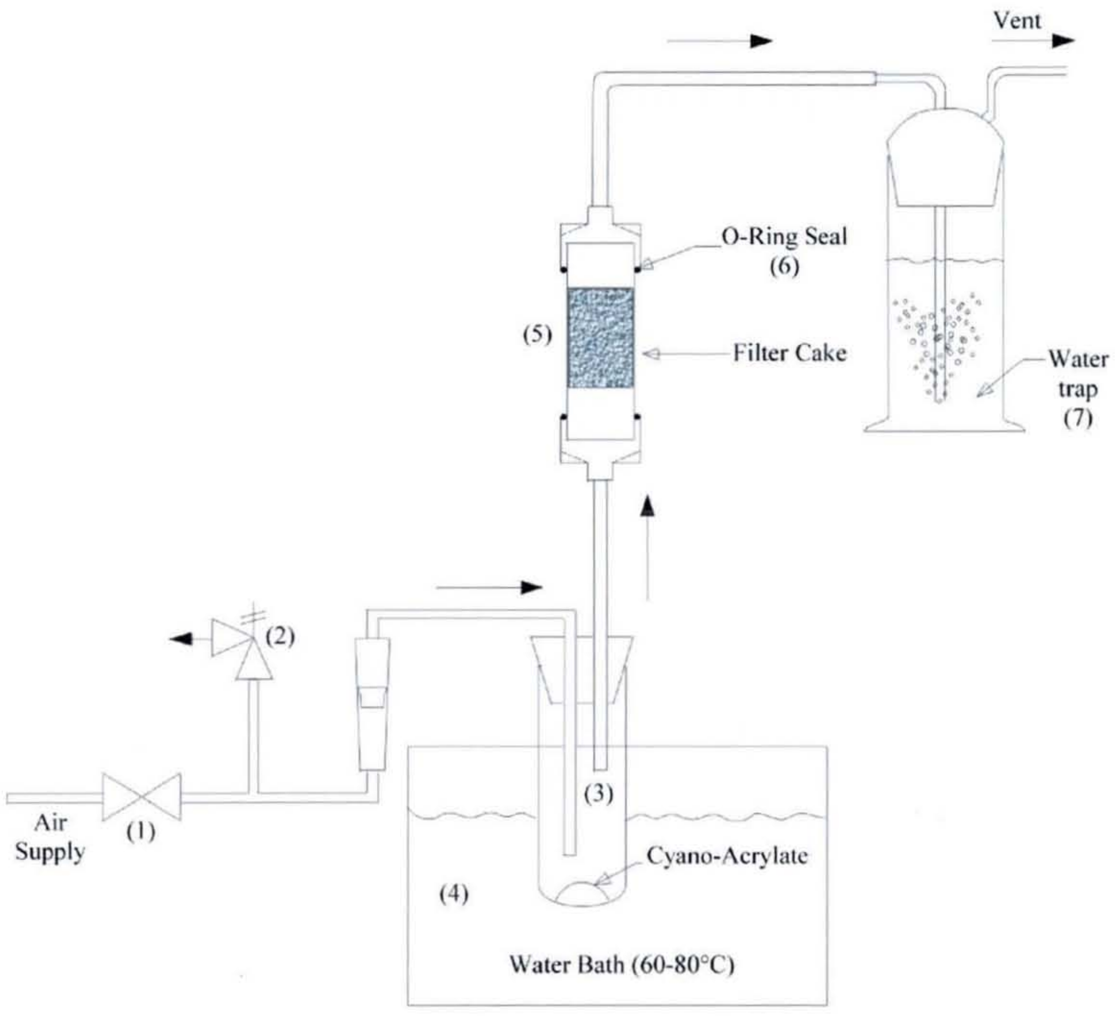


Figure 4.8 - Apparatus used for cyano-acrylate setting of a filter cake

The apparatus used compressed air (1) to push cyano-acrylate (super glue) vapours through the cake sample, which was contained in the stainless steel tube (5), sealed at both ends by o-rings (6). The tube shown as (5) in Figure 4.8 is the same as that shown in Figure 4.6. In this way, the cake was transferred only once before being at least temporarily set using the cyano-acrylate vapours. The vapours were produced by heating a tube containing the cyano-acrylate (3) in a water bath (4) to ensure a relatively constant temperature high enough to vaporise the liquid glue. On the other side of the cake sample tube was a water trap (7) to ensure that any cyano-acrylate vapours not condensed within the cake structure are removed by the water. A pressure relief valve (2) was also in place to ensure that any excessive pressure was vented before coming into contact with the cyano-acrylate.

A photograph of this apparatus is shown in Figure 4.9. Due to the fact that cyano-acrylate vapours are harmful if inhaled, the second part of the experiment was always performed within a fume cupboard.

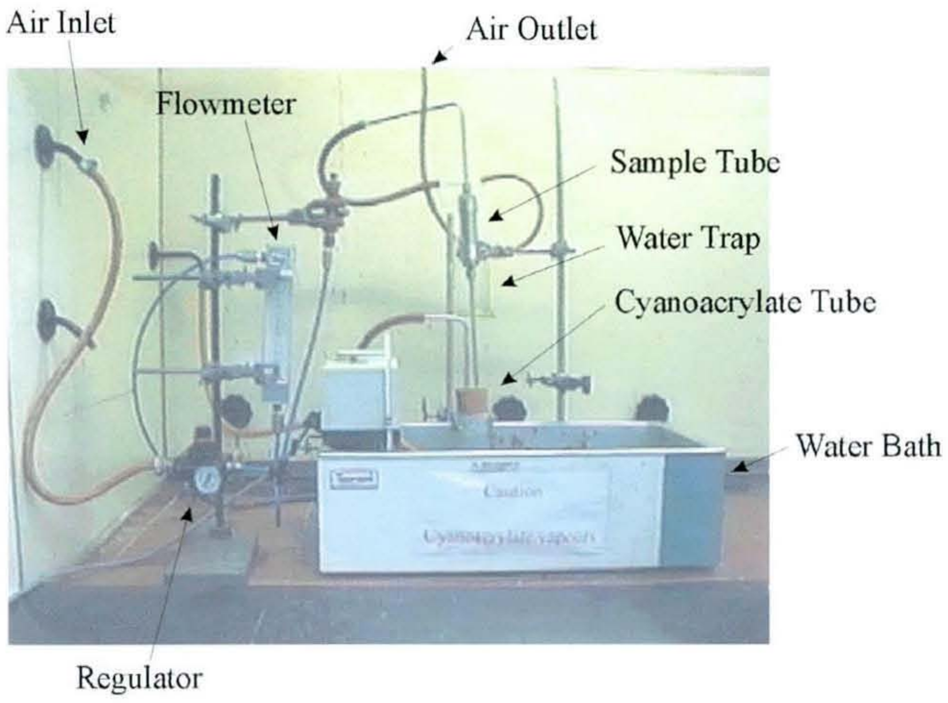


Figure 4.9 – Photograph of the apparatus used for temporarily setting the filter cake using cyano-acrylate vapours

After the cake had been set temporarily using the cyano-acrylate, an ejector (shown in Figure 4.10) was used to push the cake out of the second sample tube and into a small plastic sample (HDPE) jar. The base of the jar had been removed, with the screw-threaded lid now at the bottom, as shown in Figure 4.11. This jar was the apparatus used to hold the filter cake sample whilst being embedded in epoxy resin (described in more detail in Section 4.2.2.

The final part of apparatus (not shown here) was a disc cutter using an aluminium oxide cutting wheel with a width of 0.5 mm. This enabled a very fine section of the embedded cake to be taken before being processed by the scanning electron microscope.

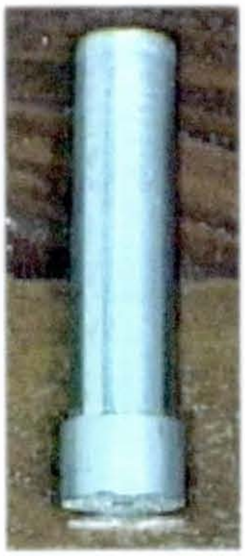


Figure 4.10 – Photograph of the ejector used to push cake out of the second sample tube (after having been set using cyano-acrylate vapours)

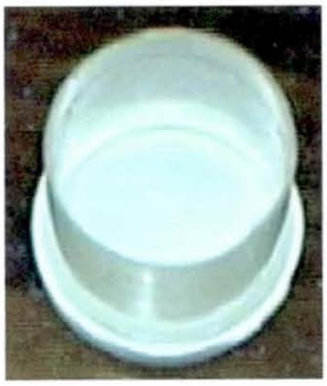


Figure 4.11 – Photograph showing a top view of the sample jar used to hold the epoxy resin whilst embedding a filter cake sample

4.2 Experimental Method

As with the experimental apparatus, the experimental method is also described in the constituent stages required to produce a stable, sectioned filter cake.

4.2.1 Cake filtration method

Initially, the required amount of distilled water was weighed using the electronic balance. The required amount of the solid mineral was then added to the water, using the stirrer to achieve a satisfactory wetting and mixing of the mineral.

The two minerals chosen for the work were calcite and talc. These were chosen as representative non-compressible and compressible filter cakes respectively. Coupled with this, it was also possible to characterise the mineral properties. Both minerals have also been used previously in filtration characterisation experiments. Analyses of the minerals used in the experimental work can be found in Chapter 5.

After a number of trials, a concentration of 20% by volume was chosen as a representative, manageable suspension for the calcite experiments in this work. This suspension comprised 800g of water with approximately 516g of calcite. For the experiments filtering talc suspensions, a 20% by volume suspension was found to be too thick for practical purposes, and a suspension of 10% by volume (243g of talc added) was used.

In the case of talc, the suspension was then stirred for 20 minutes to ensure complete mixing and wetting of the mineral suspension. This was not necessary when using the calcite, as the material readily mixed with the water and did not easily sediment in the short period time between stirring and the commencement of the experiment. With the inclusion of agitation for the talc suspensions, sedimentation was found not to be a major factor during the time scales over which the experiments were performed.

After mixing, valves (1) and (4) (see Figure 4.1) were opened, and with valves (2) and (3) closed, the suspension under examination was introduced into the system through the 'slurry input' funnel at the top of the equipment. Valves (1) and (4) were then closed. It should be noted that the sampling probe was retained at the top of the apparatus using the support pin that constitutes part of the apparatus.

The system was pressurised by opening valve (2). With this completed and the PC made ready, the filtrate outlet valve (3) was opened, allowing filtrate to exit from the vessel, through the filter medium into a collection vessel placed upon the balance (as shown in Figure 4.2). After sufficient filtrate had been collected (indicated by a small volume of air starting to escape through the filter cake), the air inlet valve (2) was closed, and the system vented using valve (4).

At the end of the filter cycle, the sampling probe was carefully pushed into the cake to take a sample. The vessel was then disassembled by loosening the flange bolts at the base of the apparatus. Once the rod and sampling probe had been removed from the vessel, the cake sample was pushed, using the ejector disk, into the second stainless steel tube (Figure 4.6). This was then placed in an oven at approximately 85°C for two hours to remove the majority of moisture present in the sample. As well as the core sample from the filter cake, used for structural analysis, a further sample of the cake was taken, weighed, and dried in an oven for six hours in order to calculate the cake porosity.

4.2.2 Cake fixing method

After a sample of the cake had been taken, it was necessary for it to be temporarily set using the cyano-acrylate vapour fixing agent. This was achieved by heating the cyano-acrylate in a water bath previously set at approximately 60°C (see Figure 4.8), at which point it progressively vaporised. The vapour was then pushed through the cake sample using low-pressure compressed air. The flow rate of the air through the sample was maintained at a level lower than the flow rate of the filtrate moving through the cake as it was being formed, so as not to disrupt the internal structure of the cake being studied. At the same time, it was necessary to maintain a sufficient flow rate to allow the cyano-acrylate vapours to travel through the height of the cake sample. The cyano-acrylate vapours were passed through the cake sample for one hour, until all the liquid setting agent had evaporated. After this time, the cake sample was much more stable and was ready for the next stage in the operation.

Whilst the cyano-acrylate was percolating through the cake structure, the water bath was also used to heat separate containers of epoxy resin and hardener. This heating process was necessary to lower the viscosities of both compounds. The two compounds are commercially available and the current work used Ciba Araldite® CY1301 Epoxy and HY1300 Hardener.

Once the cyano-acrylate had filled the pore structure of the cake to a suitable degree, the sample was discharged from the metal tube into the straight-sided plastic container shown in Figure 4.11. The straight sides were necessary in order to discharge the cake once embedded in epoxy resin. The epoxy and hardener were then mixed and poured almost immediately (to avoid any premature setting) *around* the cake sample so as to minimise the disturbance to the structure. The container with the cake sample and resin mixture was then placed in a vacuum chamber and the air expelled for a minimum of ten minutes. This partially removed air from the pore spaces within the cake and allowed the epoxy resin to enter the voids and thus embed inside the cake structure. This process was repeated to ensure that as much of the air inside the cake sample had been replaced with resin. The previous heating of the separate components ensured that the viscosity of the final embedding liquid was as low as possible to allow as high a penetration as possible of the resin into the cake structure.

The embedding agent was allowed to cure for at least twelve hours before the sample was removed from its container. After being removed, the sample was first 'squared off' using a hacksaw, leaving a block of resin containing the cake sample, as shown in Figure 4.12. It can be seen from the shape of the cake sample in Figure 4.12 that the cake remains intact during the embedding process, shown by the still circular shape of the sample. After this process, the sample was cut into a number of thin (2-5 mm) slices using an aluminium oxide cutting wheel (357 CA) mounted in a Struers Accutom-5, moving at between 0.1 and 0.15 mm/s). The positions of the cuts were recorded for future reference before further processing.

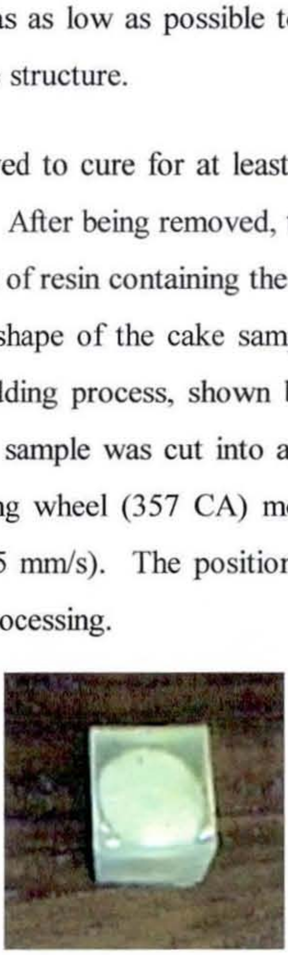


Figure 4.12 – Photograph showing a cake having been embedded in epoxy resin and squared off prior to sectioning

The slices of cake were then cleaned using distilled water, to remove any cutting fluid and loose cake material and prepared for examination under a scanning electron microscope (SEM). The fixing process applied to the sample also added contrast to the resulting image, due to the different properties (especially opacity) of the embedding agent and the particulate matter under scrutiny.

After sectioning the cake, there was the possibility that the subsequent samples would require polishing, but after initial examination, this was deemed unnecessary and possibly detrimental to the surface of the section.

Once the sample had been photographed under the scanning electron microscope (SEM), image analysis techniques were used to create a data file, which could be "read" by the software mentioned in Chapter 2. This enabled both fractal and physical descriptions of the sample to be made.

The results of the experimental trials are discussed in Chapter 5, whilst the fractal measurements of the filter cakes are discussed in Chapter 6.

5 Experimental Trials

Experiments have been carried out to produce and sample cakes formed by the pressure filtration of mineral suspensions. The work focused on the filtration performance of calcite (calcium carbonate) and talc suspensions, filtered under constant pressure conditions between 100 and 600 kPa.

5.1 Dead-end filtration of mineral suspensions

In the experiments, the filtrate volume with time data was logged using an electronic balance connected to a PC as shown in Figure 4.2, and subsequently imported into a spreadsheet application. The data was then converted from a time-mass to time-volume basis, using the density of the filtrate, in this case distilled water, with a density of 1000 kg m^{-3}. This in turn was used to produce a plot of time per unit volume (t/V) against filtrate volume (V). From this line, a value for the specific cake resistance, $\bar{\alpha}$ can be calculated, using Eq. (1.13). A typical example of the plot used to calculate $\bar{\alpha}$ is shown in Figure 5.1, using a calcite suspension filtered at 200 kPa.

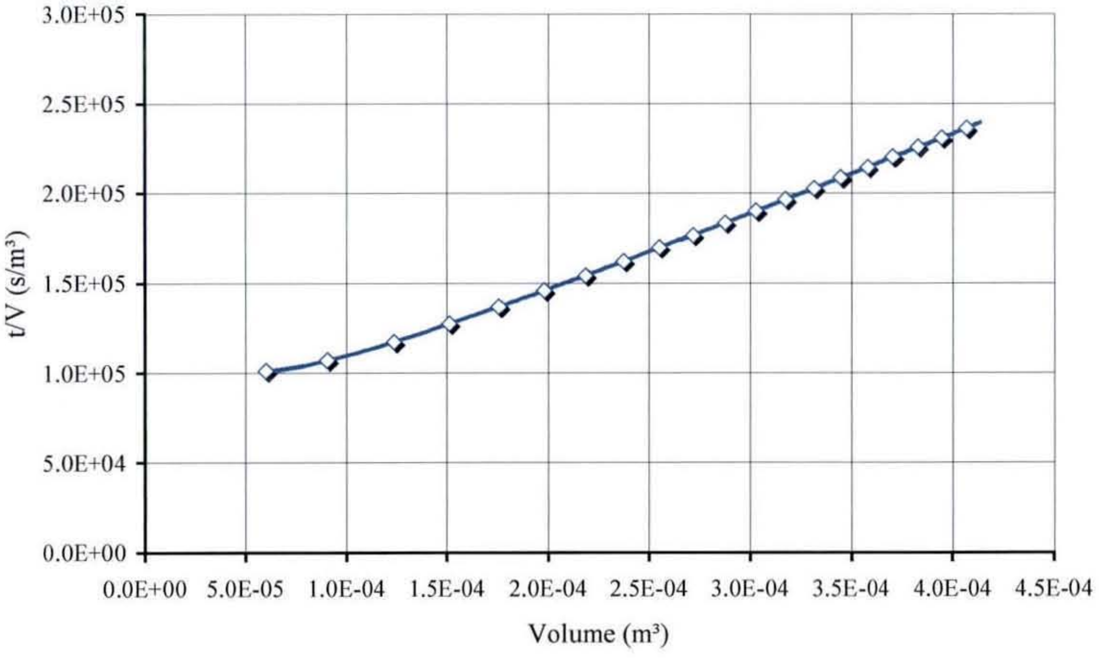


Figure 5.1- t/V vs. V for typical calcite filtration experiment performed at 200 kPa

The value for $\bar{\alpha}$ obtained from this plot was evaluated as $1.15\text{x}10^{10}$ m kg^{-1}. All measurements regarding the physical properties of filter cake were calculated using the effective feed concentration (Eqn. 1.9) rather than slurry concentration.

Not all individual *points* are shown for the data in the experimental plots, as the sample time for the filtrate volume was set at below 1 second, in order to give the most accurate results. For example, the line shown in Figure 5.1 contains only twenty of the total of 233 data points, which would be too numerous to display.

As well as the value of $\bar{\alpha}$, other physical data were obtained from the filtration experiments, namely; effective feed concentration (c), average porosity ($\bar{\varepsilon}$), filter medium resistance (R_m) and moisture ratio (m). The experiment from which Figure 5.1 was obtained yielded the following properties:

Effective feed concentration	1035 kg m^{-3}
Porosity	0.601
Filter medium resistance	$1.12\text{x}10^{11}$ m^{-1}
Moisture Ratio, m (Eqn. 1.9)	1.584

5.1.1 Reproducibility

As a part of the experimental programme, it was necessary to ascertain the reproducibility of the results obtained using the equipment described in Chapter 4. This reproducibility was demonstrated during the experimental schedule by plotting the time-mass data of experiments performed under the same conditions. Figure 5.2 shows the time-mass data for calcite suspensions filtered at 300 kPa pressure. This chart shows that the experiments can be considered reproducible.

Figure 5.3 shows the reproducibility of the experimental method using the t/V against V plot for a range of pressures. In this case, the variation in time per volume with filtrate volume can clearly be seen, with a clear sequence in the experimental results.

Figure 5.4 shows the time to complete filtration for each of the experiments shown in Figure 5.3. Figure 5.4 shows that the time for the same volume of filtrate to pass through the filter cell had a reciprocal relationship with pressure under which the experiments were performed. The total suspension fluid volume used in the experiments was 800ml, with an average 435ml filtrate passing through the cell into the balance.

As the data in Figure 5.3 and Figure 5.4 show, the filter cell used for this work proved to be reliable apparatus.

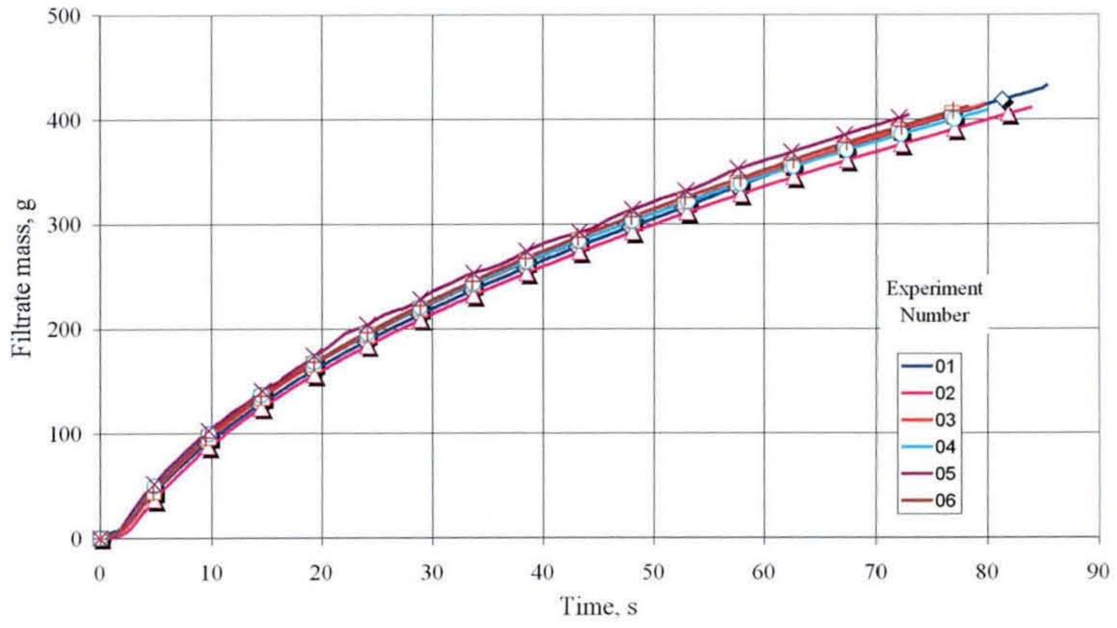


Figure 5.2 - Graph showing the reproducibility of experiments performed using calcite suspensions at 300 kPa pressure

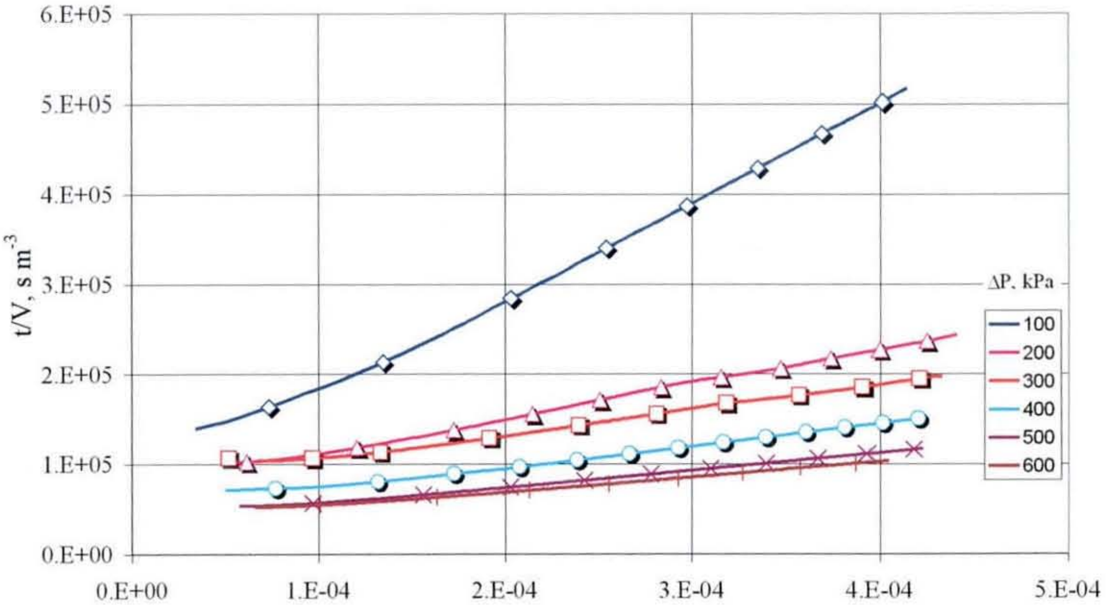


Figure 5.3 – Graph showing typical t/V against V data for experiments performed at a range of pressures

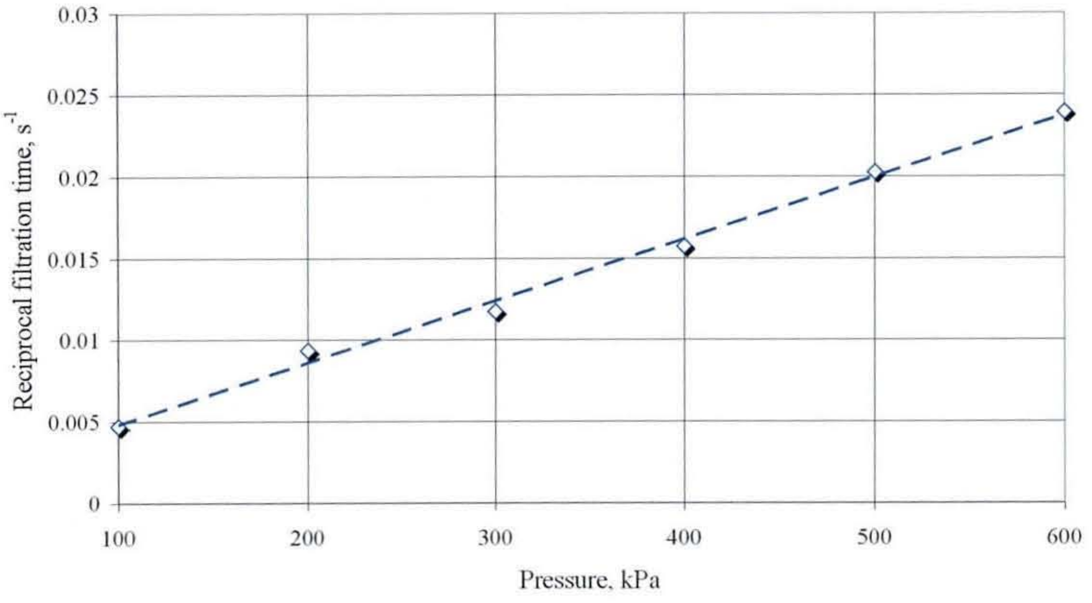


Figure 5.4 - Variation in experimental run time with pressure

5.1.2 Compressibility of filter cakes

In order to calculate the compressibility of the cakes formed from suspensions of the minerals under investigation, experiments were performed varying only the filtration pressure, with all other parameters remaining constant. After the series of tests, the cake resistance at unit pressure drop, α_0 and compressibility index, n were calculated using Eq. (1.15), from a plot of ln $\bar{\alpha}$ vs. ln ΔP.

Figure 5.5 shows the plot of ln $\bar{\alpha}$ vs. ln ΔP for the experiments performed using calcite suspensions.

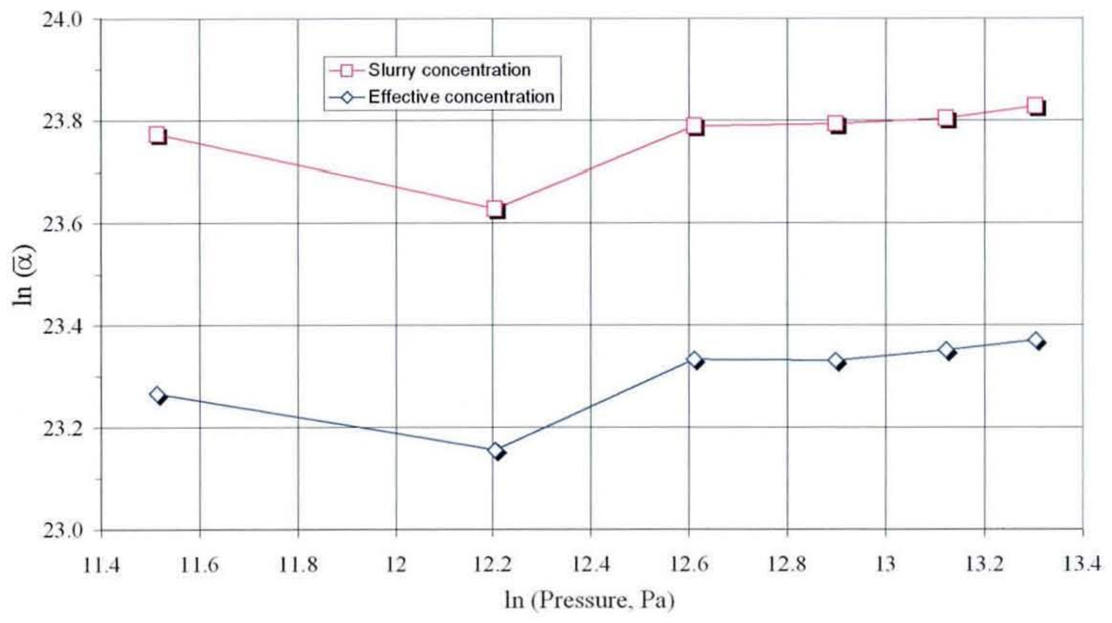


Figure 5.5 - ln $\overline{\alpha}$ vs. ln ΔP to calculate specific cake resistance at unit pressure drop for calcite at constant pressures of 100 to 600 kPa pressure

The results yielded from the data shown in Figure 5.5 gave a value for cake resistance, α_0 of 4.69x10^9 m kg^{-1} Pa^{-n}, with the compressibility index, n calculated as 0.08. Both of these results were obtained using the effective feed concentration for each experiment. Results showed that using the effective, as opposed to actual, feed concentration yielded more accurate results, as indicated by a higher correlation coefficient (r^2) value for the data (0.46 compared to 0.25). Whilst the values for the correlation coefficients were low, removing the data generated from experiments performed at 200 kPa gave an r^2 value for the data from the effective feed concentration of 0.96. This is considerably higher than if the 200 kPa data was included. The removal of this data also changed the value of α_0 to 6.78x10^9 m kg^{-1} Pa^{-n} and n to 0.05. It was assumed from the data shown in Figure 5.5 that the results from the filtration trials carried out at 200 kPa pressure drop could be attributed to experimental error such as the manual measurement and control of filtering pressure. The results from these analyses showed that the filter cakes formed from calcite suspensions were essentially incompressible.

The application of the same calculations was also applied to the filtration of talc suspensions, again with constant concentrations and varying pressures between 100 and 600 kPa.

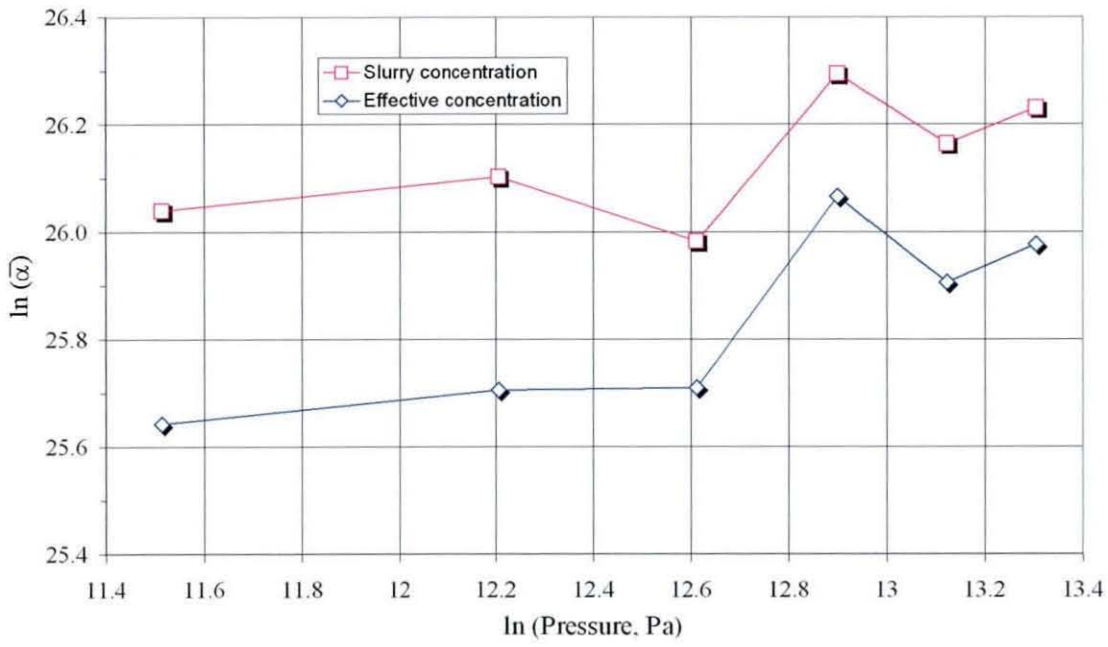


Figure 5.6 - ln $\bar{\alpha}$ vs. ln ΔP to calculate specific cake resistance at unit pressure drop for talc at constant pressures of 100 to 600 kPa pressure

Figure 5.6 shows the data for the filtration of talc corresponding to Figure 5.5. In this case the value of α_0 was calculated to be 1.15×10^{10} m kg^{-1} Pa^{-n}, whilst the compressibility index, n was calculated as 0.21. The value of the correlation coefficient for the data was 0.66 when calculated using the effective feed concentration and 0.40 when calculated using the slurry feed concentration. This again showed that the effective feed concentration was the more reliable figure to use when calculating the physical properties of the systems under scrutiny. These results also showed that the talc filter cakes could be considered representative of compressible systems.

Figure 5.7 also shows that the filter cakes formed from the filtration of talc suspensions were more compressible than those formed from calcite suspensions under the same conditions. Figure 5.7 presents the variation in average porosity, $\bar{\varepsilon}$, with pressure. As this figure shows, the porosity of the calcite filter cakes varies only slightly with increase in pressure, whilst that of the talc filter cakes decreases as the pressure increases (especially in the range 100 to 400 kPa). These conclusions are in agreement with the original choice of the calcite and talc as forming incompressible and compressible filter cakes from mineral suspensions.

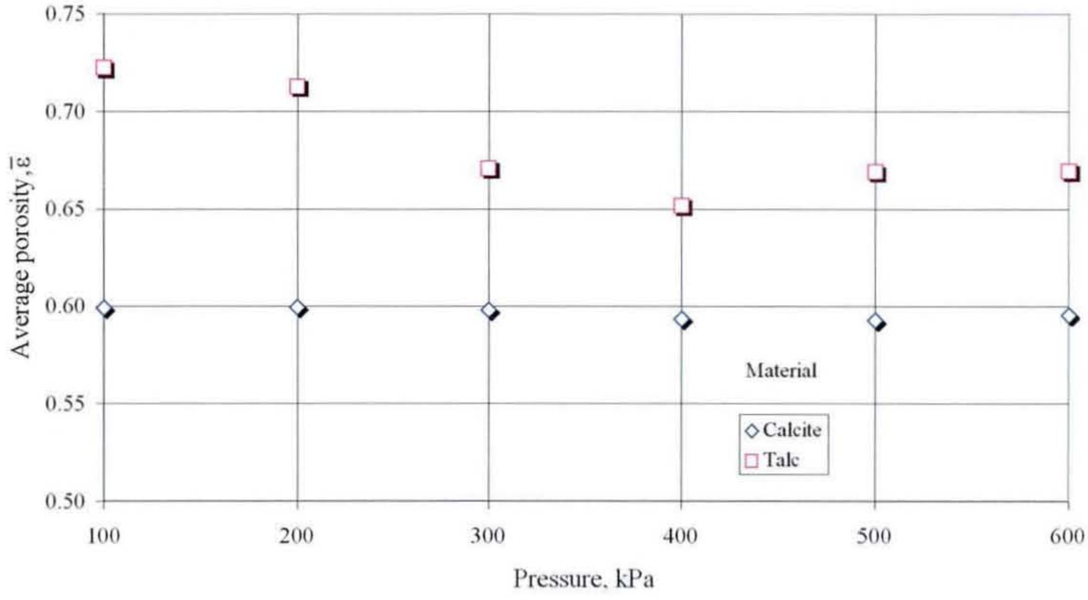


Figure 5.7 - Variation in porosity with pressure for calcite and talc filter cakes

A full set of summaries for the dead-end filtration experiments performed are shown in Appendix C.

5.2 Mineral characterisation

The minerals were also characterised in terms of size prior to filtration. Test were carried out using a Coulter® Counter LS with the minerals in a suspension of distilled water (the same carrier fluid as the filtration trials). Results from these tests give the average particle size (d_{50} by volume) of calcite as 15.47 μm, whilst the d_{50} of the talc particles was calculated to be 17.27 μm. The calcite was found to have had slightly narrower distribution than the talc, with a d_{10} of 4.22 μm, compared to the talc with a d_{10} of 4.32 μm. The d_{90} of the two materials also showed this, with d_{90} (calcite) = 45.40 mm and d_{90} (talc) = 50.95 μm.

In addition to the particle size distributions of the two materials, the density of the two minerals was also required for the experimental procedures. Calcite was found to be 2580 kg m^{-3}, whilst that of the talc was 2742 kg m^{-3}.

A full set of the results from these tests, both experimental results and particle size distributions, can be found in Appendix C.

5.3 Conclusions

The experimental results showed that calcite could be used as a representative material forming incompressible cakes under dead-end filtration, whilst talc was shown to produce cakes that were moderately compressible under the same conditions.

The procedure and results proved that the method, whilst basic, did provide reproducible results that could be used as the foundation for comparison with the fractal measurement techniques described elsewhere in this work.

6 Spatial Characterisation of Filtration

In the previous chapter, the more traditional characteristics of filtration were examined, including the compressibility and porosity of the cakes. Chapter 6 explores the spatial nature of the minerals and formed filter cakes used for the trials. This was achieved by analysing both fractal and Euclidean space properties of the particles and cakes.

6.1 Analysis techniques

Both the individual particles and samples of the cakes formed were analysed using a variety of techniques. In all cases, images were taken by scanning electron microscope (SEM), scanned and digitised by PC.

The original image was initially scanned and cropped to provide a suitable working area for further analysis. Figure 6.1 shows a typical first image that was obtained from an SEM for a calcite filter cake (filtered at 600 kPa). Figure 6.2 shows the SEM image for an individual talc particle.

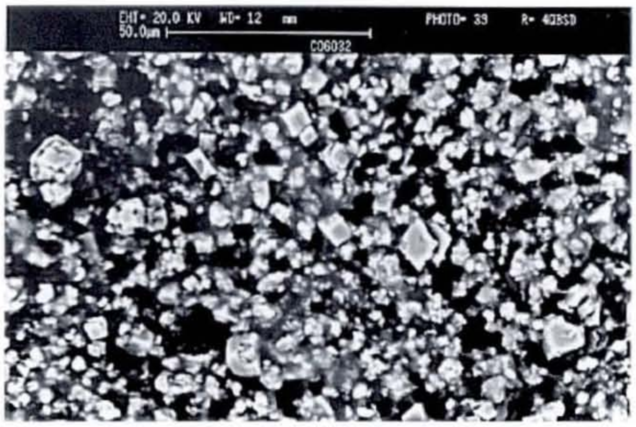


Figure 6.1 – Typical calcite filter cake scanned from SEM

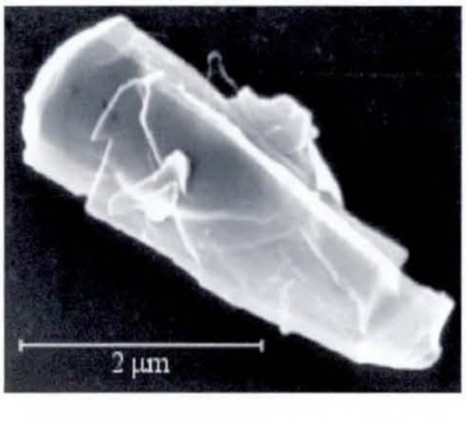


Figure 6.2 - Talc particle scanned from SEM

After the images were imported into the computer, image analysis software (Visilog® or Scion Image®) was used to produce a 'threshold' image of the material in question. This threshold technique changed any grey-scale values above a certain level (the threshold) to white and any grey-scale values below the threshold to black. In this way, a monochrome image of the structure was formed. Figure 6.3 shows the filter cake section in Figure 6.1 after the application of a threshold, whilst Figure 6.4 shows the talc particle in Figure 6.2 after the same treatment.

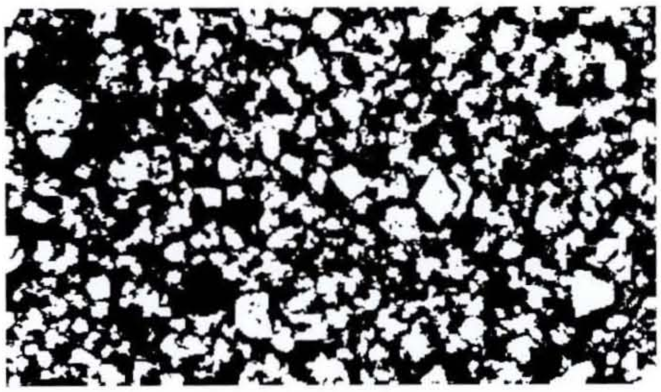


Figure 6.3 - Figure 6.1 after application of threshold technique

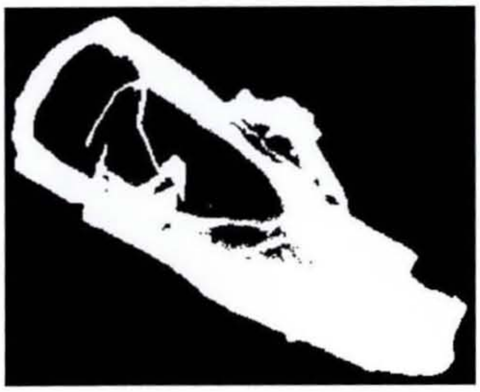


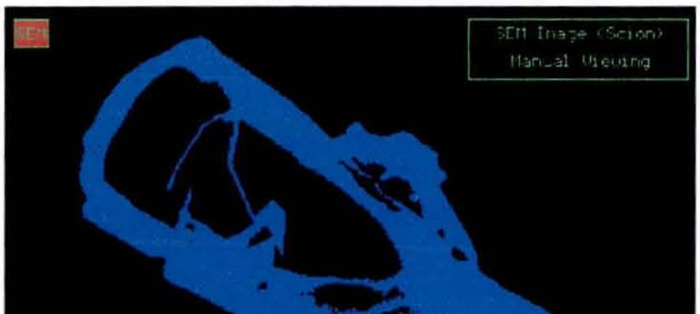
Figure 6.4 - Figure 6.2 after application of threshold technique

Once these monochrome, or 'binary', images were created using the image analysis package, they were saved to computer disk in a text format, with digits representing black and white space within the image. A header was also applied to these files, containing information on the type, size and borders of the images. Figure 6.5 shows this header information with the start of the pixel data for a typical particle.

```
SEM Image (Scion)
Size:
H  260 W  325
Borders:
T 005 B 000 L 000 R 000
0000100000000000000000000000000000
0000110000000000000000000000000000
0001110000000000000000000000000000
0011100000000000000000000000000000
0111100000000000000000000000000000
0011100000000000000000000000000000
0001100000000000000000000000000000
0001100000000000000000000000000000
0001110000000010000000000000000000
0011111000000011100000000000000000
0111111100000111110000000000000000
0011111111111111111110000000000000
0011111111111111111110000000000000
0001111111111111111111000000000000
0001111111111111111111100000000000
0001111111111111111111110000000000
0011111111111111111111110000000000
```

Figure 6.5 - Binary file header and pixel information

The text files were then read by the software created for this work (see Chapter 2) and converted back into a recognisable form. Figure 6.6 shows the particle in Figure 6.2 decoded by the program. Although, in terms of surface detail, the resolution appears to have been reduced in the transition between the original SEM and the image imported into the computer program, the detail of the boundary (necessary for the type of analysis performed in this case) remains intact. The procedure for the digitisation of the filter cake images was the same as that for the individual particles, although with different analyses being performed.



```
SEM Image (Scion)
Manual Viewing

C:\TP\TEST DATA\talc.dat
Press any key to continue
```

Figure 6.6 - Representation of Figure 6.2 in Turbo Pascal® program

6.2 Analysis of individual particles

Initially, the individual mineral particles were characterised using a number of techniques to analyse both their boundaries and overall shape.

6.2.1 Particle Boundary Characterisation

Some work was performed to measure the fractal dimension of individual particles from a sample of the mineral under observation. The program described in Chapter 2 had the ability to read binary files containing particle information and measure the fractal dimension of the particle concerned using the structured walk technique. Samples of both calcite and talc were analysed in this way. The results of those analyses are presented here.

The structured walk technique used was essentially the same as that described in Chapter 2 for the seed agglomeration simulated structures. The centre of gravity of the object and maximum and minimum Feret diameters were first determined, and a point furthest from the centre of gravity used as a start point for the walk. A structured walk analysis was then performed, again using the steplength range 0.08 to 0.32 times the maximum Feret diameter, with the surface fractal dimension (D_s) calculated using a Richardson Plot.

The results showed that the average surface fractal dimension of the calcite particles was somewhat greater than that for the talc particles (1.109 cf. 1.092). In real terms, considering that the surface fractal dimension measured must, by definition lie between 1.0 and 2.0, this represented a difference of approximately 15% between the two minerals (average talc $D_s = 0.109$ cf. average calcite $D_s = 0.092$). Table 6.1 shows the surface fractal dimensions for the two types of particle measured using the technique described above. The 'I.D.' column refers to the identification given to each of the particles. Further details on these can be found in Figure 6.11 and Appendix D.

Table 6.1 shows that there were two particles in particular (6250 and 6296) with fractal dimensions significantly different from the rest of the objects studied. This demonstrated that there was variation of fractal dimension within the minerals, as well as between the two minerals. The results given here show that more samples of the mineral should be investigated to give a clearer, statistically valid average boundary fractal dimension for the two materials.

Table 6.1 - Surface fractal dimensions of calcite and talc particles using $\lambda = 0.08F$ to $0.32F$

Calcite		Talc	
I.D.	D_s	I.D.	D_s
6248	1.111	6292	1.059
6249a	1.095	6293	1.077
6249b	1.092	6294	1.089
6249c	1.079	6295	1.047
6250	1.253	6296	1.203
6251	1.081	6297	1.100
6252	1.055	6298	1.066

Although other fractal characterisations (notably those of particle agglomerates, detailed in Chapter 3) did not consider the analysis of structured walk fractal dimensions by the subtraction of the lower limit of D_s, these other characterisations gave results where trends were clearly visible.

In the case of the characterisation of mineral particles, the particles themselves could not be considered to be true fractal (that is to say self-similar) objects. With the images used for this section of work, as the scale of scrutiny was increased (i.e. the objects were effectively being examined at a higher magnification), the shape of the particles became more uniform. This was probably due to the removal of the images from the original source (i.e. the scanning electron microscope) and their placing into the characterisation application (the computer program). Using this method (the only workable solution available) meant that only a finite amount of information concerning the particles could be communicated between the physical (SEM) and computational (computer program) analysis methods.

It would not have been a practical exercise to measure the perimeter of the shapes at various magnifications under the microscope or to scan the scanned images at higher resolutions in order to provide boundaries that were more accurate. This would require the apparent size of the particle to become ever bigger in the computer application. The most obvious solution to this problem (and the solution used in this work) was to size the particles so that they filled a reasonable amount of space on the computer screen, to convey as much detail as possible without falling outside the boundaries of the application. However, in theory, the analysis of the higher resolution images would have been the method by which more details of the perimeter would have been apparent, resulting in a more accurate fractal measurement. With the improvements in computational power and using different software from that chosen for this work, it might well be possible to analyse the minerals in this way.

Thus, under the scale of scrutiny available (the loading of digitised images into the computer program), the best way of comparing the results was to use a lower range of steplengths ($0.02F$ to $0.08F$), effectively examining the particles at a higher magnification.

When the lower range of steplengths was used, the average surface fractal dimension of the calcite particles was 1.129 (increased from the results generated using the original range of steplengths), with the average fractal dimension of the talc particles equal to 1.055 (decreased from the previous results). Again taking into consideration the constraints of the surface fractal dimension being between 1.0 and 2.0, the further analyses showed a much greater distinction between the two minerals, with the calcite particles having a higher average surface roughness than the talc particles.

Whilst the original analyses (with greater steplengths) yielded a difference in surface fractal dimension between the talc and calcite particles of 15%, closer scrutiny yielded results with a difference between the two of approximately 57% (average talc $D_s = 0.055$ cf. average calcite $D_s = 0.129$).

These results also showed that the range of fractal dimensions was wider for the calcite particles than the talc particles. Using the range $\lambda = 0.08F$ to $0.32F$ yielded a standard deviation of fractal dimensions for the calcite of 0.066 and for the talc 0.052. When the lower range of steplengths was employed, the standard deviation for calcite remained almost the same at 0.065. However, the standard deviation for the talc fractal dimensions fell to 0.026, considerably lower than that for the calcite.

This would seem to indicate that the original measurement parameters of 0.08 to 0.32 times the Feret diameters did not necessarily yield the best results in terms of differentiating between the two minerals in this case. In those systems whose detail is finer than the lower limit of the steplengths used to measure it, or where the object is not truly self-similar from a large scale, finer scrutiny, by the use of smaller steplengths may be of benefit.

The values for the surface fractal dimensions agree well with literature values [Peleg and Normand, 1985] that suggest a range of fractal dimensions for natural systems of between 1.05 and 1.36.

The fractal dimensions for each of the particles measured at 0.02*F* to 0.08*F* are shown in Table 6.2. Clear differences can be seen between these results and those shown in Table 6.1.

Table 6.2 - Surface fractal dimensions for calcite and talc particles using λ = 0.02F to 0.08F

Calcite		Talc	
I.D.	D_s	I.D.	D_s
6248	1.180	6292	1.108
6249a	1.186	6293	1.056
6249b	1.054	6294	1.056
6249c	1.071	6295	1.037
6250	1.208	6296	1.056
6251	1.137	6297	1.026
6252	1.064	6298	1.049

In order to ensure that the fractal dimensions generated from this (and indeed all other measurement techniques), both the correlation coefficient and the critical values for the correlation coefficient were calculated using statistical tables built into the program structure [White et al, 1991]. This ensured that any 'fractal rabbits', as described in Section 1.4.3 would be noted due to the fact that the data points from the Richardson plot would not all belong to the same population. In all cases, the probability that the data belonged to the same population was > 99.9%. This check was a particularly important action when using steplengths outside of the recommended range 0.08*F* to 0.32*F*.

6.2.2 Particle aspect ratio

In addition to the surface fractal dimension, it was also possible, using the maximum and minimum Feret diameters, to calculate the aspect ratio of each of the particles under examination.

The aspect ratio was defined as the ratio of maximum to minimum Feret diameters (measured in pixels). Table 6.3 shows the aspect ratio (A.R.) for the calcite and talc particles characterised above.

Table 6.3 - Aspect ratios measured for calcite and talc particles

Calcite		Talc	
I.D.	A.R.	I.D.	A.R
6248	1.569	6292	1.499
6249a	1.874	6293	1.269
6249b	1.296	6294	1.595
6249c	1.226	6295	2.455
6250	1.465	6296	1.733
6251	1.322	6297	3.344
6252	1.509	6298	2.239

The results from the calculation of aspect ratio show that the average aspect ratio of the talc particles was higher than that of the calcite particles, although this figure may be somewhat skewed by the particularly high aspect ratio of particle I.D. 6297. The average aspect ratio for the calcite particles in Table 6.3 was 1.466, whilst the average for the talc particles was 2.019 (1.798 with the removal of particle I.D. 6297).

6.2.3 Particulate structures

It can be seen from Figures 6.7 to 6.10 that the shape of the two different types of mineral particle varied quite dramatically. The calcite tended to be of a rhombohedral nature, and from examples such as Figure 6.8 would appear, at least in the dry state, that small constituent particles connect together to form the whole. Figure 6.7 also demonstrates an almost crystalline structure, where smaller rhombohedral shapes cover the surface of the individual particle. The talc particles, however, were more likely to be individual, elongated particles, with smooth surfaces.

These observations were borne out to some extent in the results of the fractal boundary and aspect ratio analyses (Tables 6.1 to 6.3). In some instances (e.g. Figure 6.10) the results were skewed somewhat by appendages on the particles, both in terms of the calculation of fractal dimension and aspect ratio.

Figures 6.7 to 6.10 show typical calcite and talc particles, with the fractal dimension as measured at $\lambda = 0.08F$ to $0.32F$.

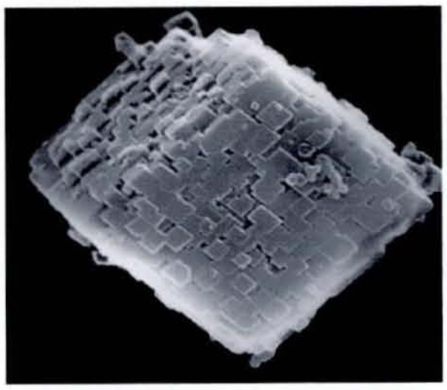


Figure 6.7 –Calcite particle, perimeter fractal dimension = 1.055

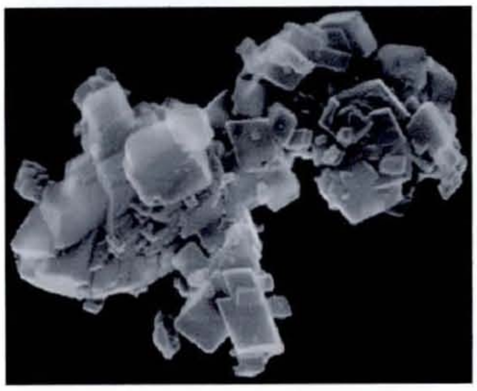


Figure 6.8 - Calcite particle, perimeter fractal dimension = 1.253

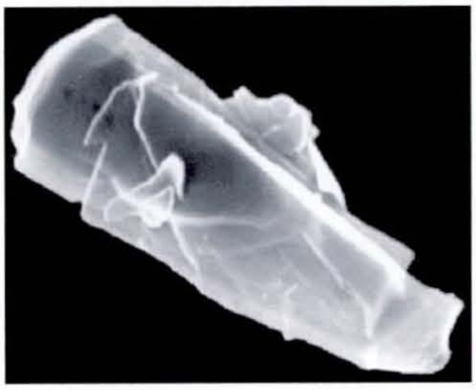


Figure 6.9 - Talc particle, perimeter fractal dimension = 1.047

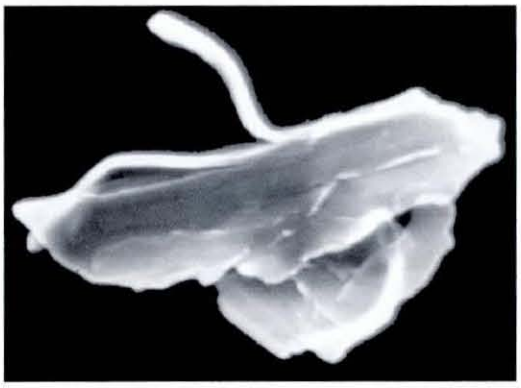


Figure 6.10 - Talc particle, perimeter fractal dimension = 1.203

6.3 Characterisation of sectioned filter cakes

The filter cakes were formed and sampled using the experimental method described in Chapter 4 before being digitised for use in the computer program as described in Section 6.1. Two micrographs were created from each section of filter cake under scrutiny in order to obtain a representative sample of each slice. In total, 38 micrographs were used to obtain the sample results detailed here.

After the digitisation process, the samples were analysed using the box counting technique described in Chapter 1 to give overall density fractal dimensions. The two dimensional porosity of the samples was also analysed, in order to correlate this with the experiments performed.

Unfortunately, all attempts to section cakes formed by the dead-end filtration of talc suspensions failed, due to the difficulty in removing all the moisture from the internal structure of the cake. Whilst the method appeared successful, with a relatively strong cake sample removed from the sample tube described in Chapter 4, each of the samples disintegrated upon sectioning with the cutting wheel. Inspection of the cake fragments revealed that there was a large amount of moisture still present in the structure of the cake, which the drying process had not removed. The moisture could not be removed even when the drying time of the process was increased to 24 hours. As the moisture had not been fully removed, neither the cyano-acrylate nor the epoxy resin used to set the samples fully penetrated the pores of the cake. This in turn meant that upon sectioning, the cutting wheel caused the particles within the cake to separate. A number of attempts were made to perform this operation, without success. The following results, therefore, refer only to the calcite cakes. Chapter 5 however, contains the information gained from the filtration of talc suspensions (compressibility index etc.) and thus serves as a start point for future work involving the sampling of formed cakes.

It was found during experimentation that the best results from the sectioned cakes were found at the base and top of each of the samples. Figure 6.11 gives a graphical representation of the position of the samples taken from each of the cakes, to aid further analyses of their structures. The thickness of each of the slices was approximately 3 mm.

It should be noted from Figure 6.11 that, with the exception of results at 400 kPa, at least one sample was taken from the top and bottom of each cake sampled. It was not always possible to take further samples from all the cakes formed, as problems were experienced with the sectioning procedure, as explained above for the talc cakes. A synopsis of the results for each of the samples can be found in Appendix D.

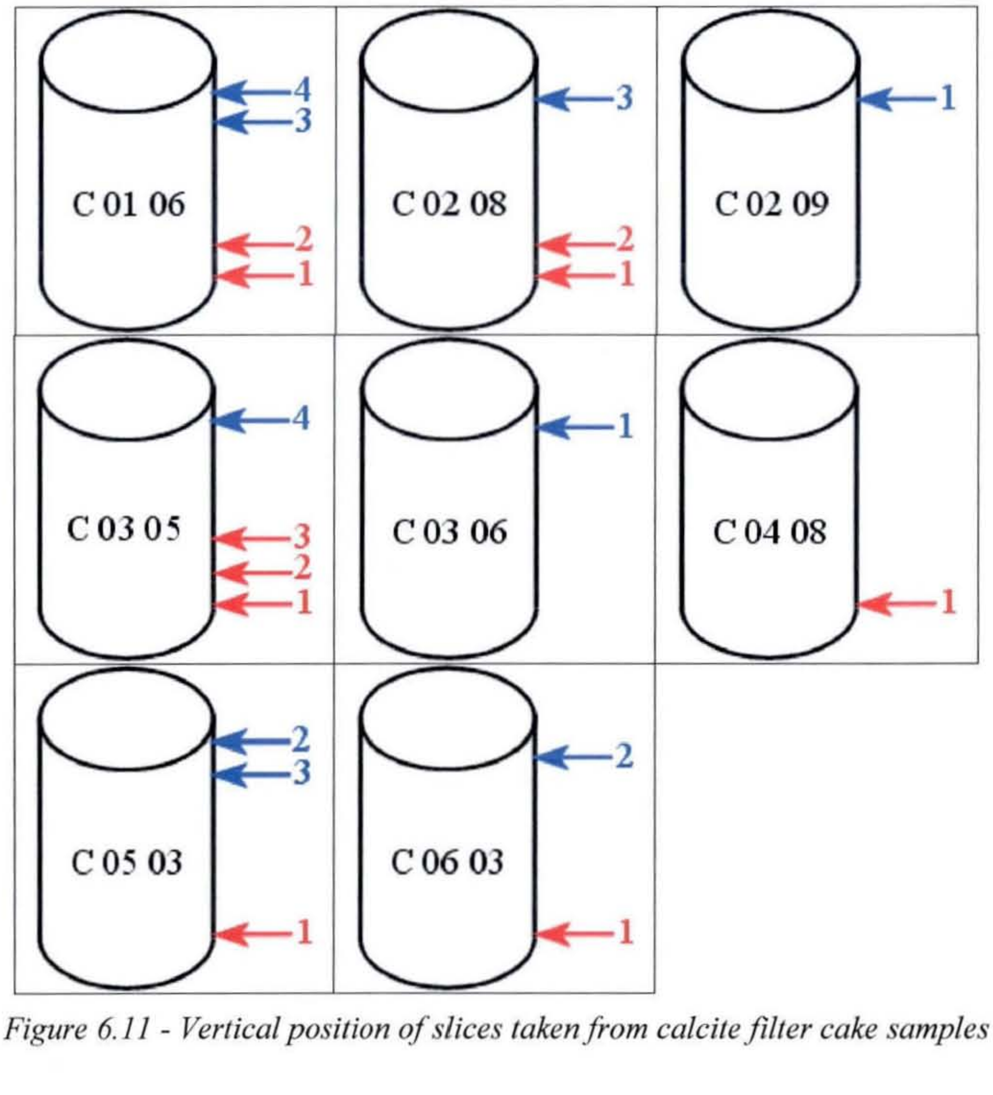


Figure 6.11 - Vertical position of slices taken from calcite filter cake samples

The codes shown in the sample representations in Figure 6.11 give the experiment number and are explained in Appendix D

6.3.1 Fractal analysis

The box counting fractal dimension was calculated using the method described in Sections 1.4.1.8 and 2.4.5, after laying a grid over the digitised object such as that shown in Figure 6.3. Figure 6.12 shows just such a grid (with the length of the boxes, *b*, set at 0.01*F* laid over the sectioned filter cakes shown in Figure 6.1. In each case, the start position for the boxes was at the top left hand corner of the image, extending down and to the right.

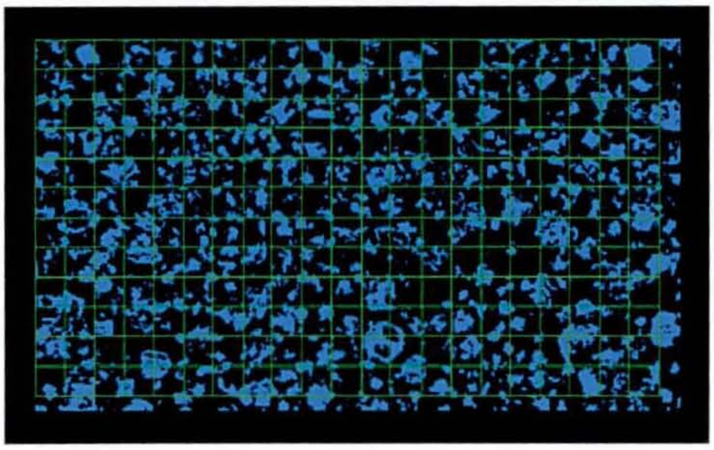


Figure 6.12 - Box counting grid at 0.10F laid over the filter cake shown in Figure 6.1

Figure 6.13 shows the box counting fractal dimension (an average of all of the samples analysed at each pressure) against pressure for the formed filter cakes. There was perhaps a slight overall trend of increasing fractal dimension with increase in pressure. This trend, however, was small - with a difference between the minimum and maximum results of only 0.4%. The difference between the fractal dimensions of the samples rose to 6% when the lower dimension boundary of 2.0 was subtracted from the results (as described in Section 6.2.1). This still did not represent a great change in fractal dimension with pressure.

The results from the measurement of the fractal dimensions agree with the results shown in Chapter 5, which showed that cakes formed by the dead-end filtration of calcite suspensions could be considered incompressible.

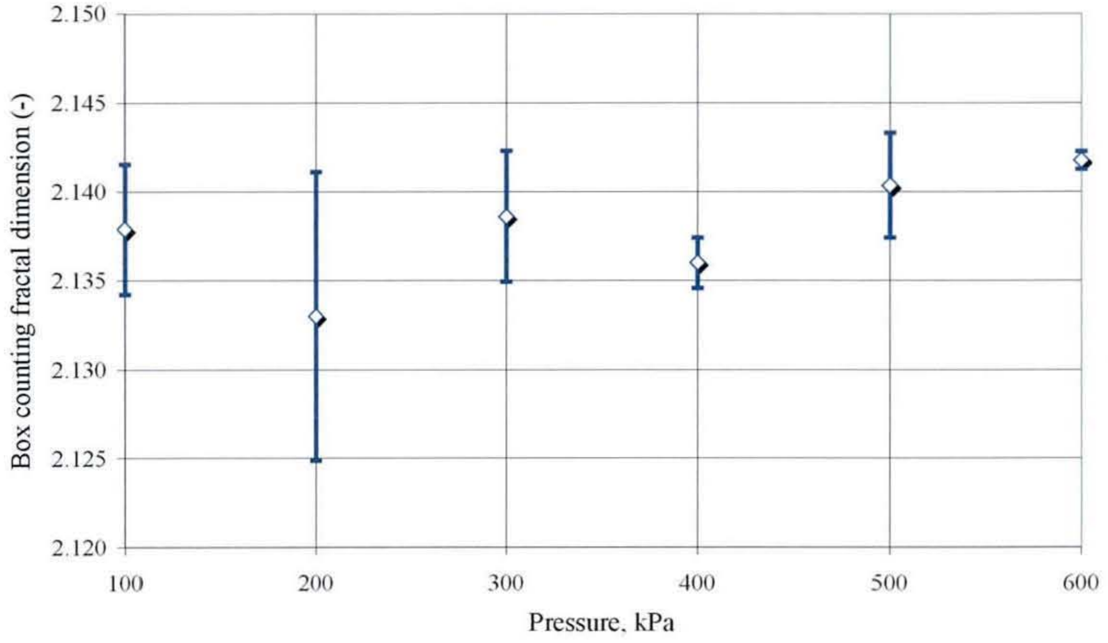


Figure 6.13 - Box counting fractal dimensions for calcite filter cake samples, with ± one standard deviation shown as error bars

As can be seen from the standard deviation for the samples characterised at 200 kPa, this data appears to be the least reliable of the set. As discussed in Chapter 4, the filtration data used for the calculation of the various physical properties of the same cakes was also the least reliable at 200 kPa, possibly due to the manual control and measurement of the filtering pressure.

Figure 6.14 shows the change in box counting fractal dimension with the vertical position of the slice within the cake sample. In this case, the averages of the box counting fractal dimensions were taken for each end of the sample and plotted against pressure. Figure 6.14 shows that there was generally little variation, and seemingly no overall trend between the box counting fractal dimension for the top and bottom of the cake sample.

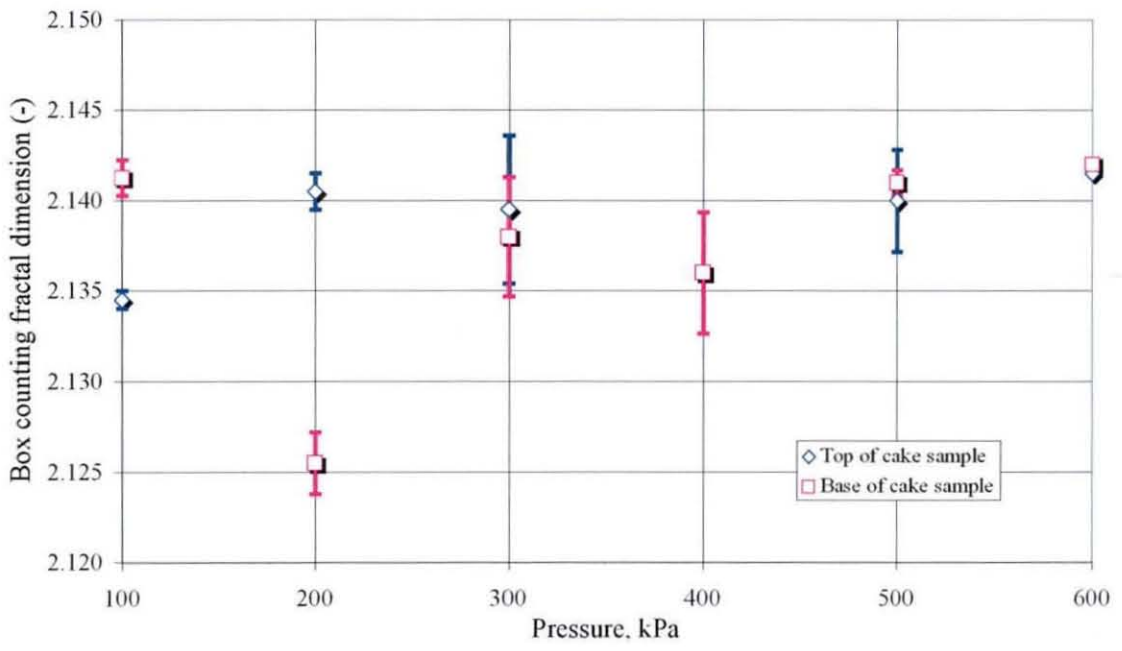


Figure 6.14 - Box counting fractal dimensions showing variation with vertical position within cake samples, with ± one standard deviation shown as error bars

For the correlation between fractal measures of the filter cakes and their physical characteristics, Figure 6.15 shows that there may be a relationship between the box counting fractal dimension and the specific cake resistance, $\overline{\alpha}$, as calculated from the experimental results. The values for $\overline{\alpha}$ are taken from Figure 5.5, the specific cake resistance for calcite, calculated using the effective feed concentration. Although an absolute correlation was not observed, the results for filtration at 100, 300, 500 and 600 kPa appear to suggest a correlation between fractal dimension and cake resistance. It should be noted from Figure 5.5 that the experimental results for filtration performed at 200 kPa did not form a linear relationship with the other results. In addition to this, it was only possible to take one sample from the filter cakes formed at 400 kPa.

Figure 6.16 shows the relationship between the average box counting fractal dimensions and permeabilities for cakes formed from suspensions filtered at 100 to 600 kPa. As can be seen, the apparent correlation between the two is the inverse of that shown when correlating the fractal dimension with cake resistance. In the case of specific cake resistance, the relationship showed a rising fractal dimension with increasing resistance. Figure 6.16 shows that the fractal dimension decreased with increasing permeability.

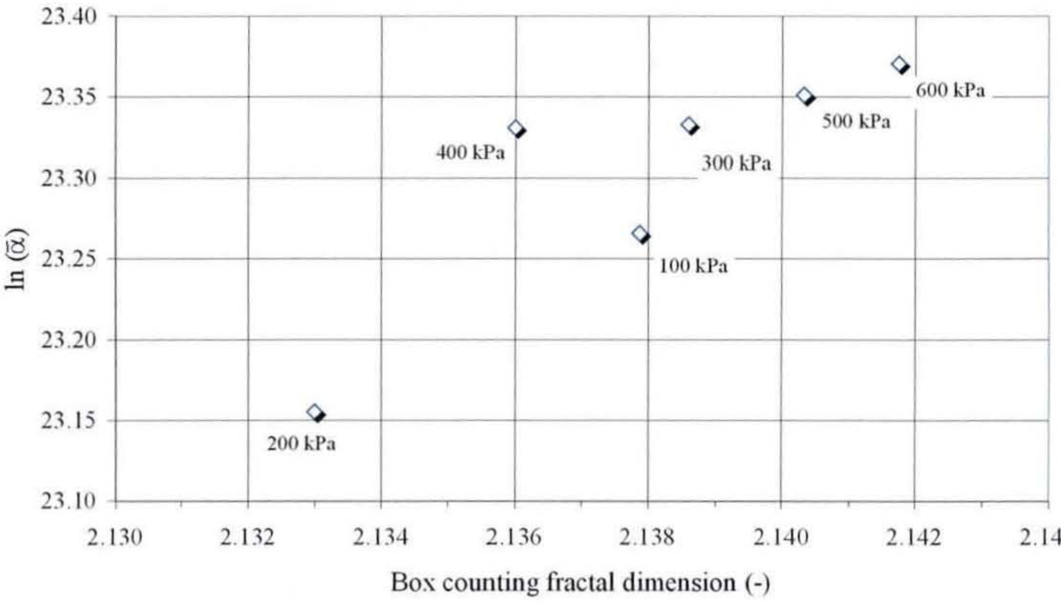


Figure 6.15 - Graph showing relationship between box counting fractal dimension and specific cake resistance for calcite suspensions filtered at constant pressure between 100 and 600 kPa

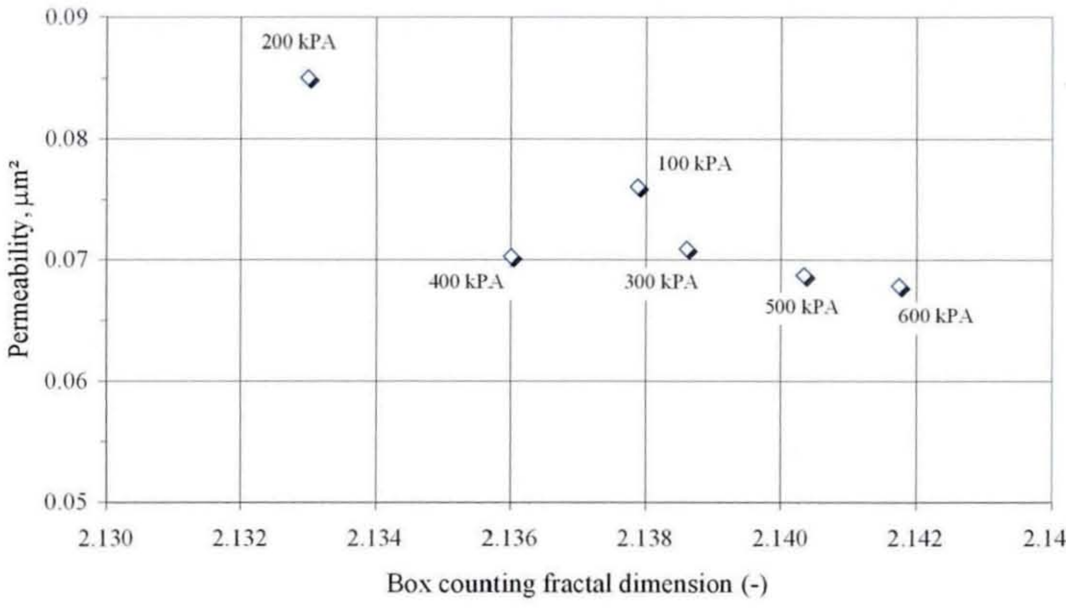


Figure 6.16 - Graph showing relationship between box counting fractal dimension and cake permeability for calcite suspensions filtered at constant pressure between 100 and 600 kPa

As can be seen from Figure 6.17, there was however, found to be no relationship between the box counting fractal dimension and the porosities measured from the experimental trials. Figure 6.17 shows the box counting fractal dimension plotted against the porosities obtained from the experimental trials.

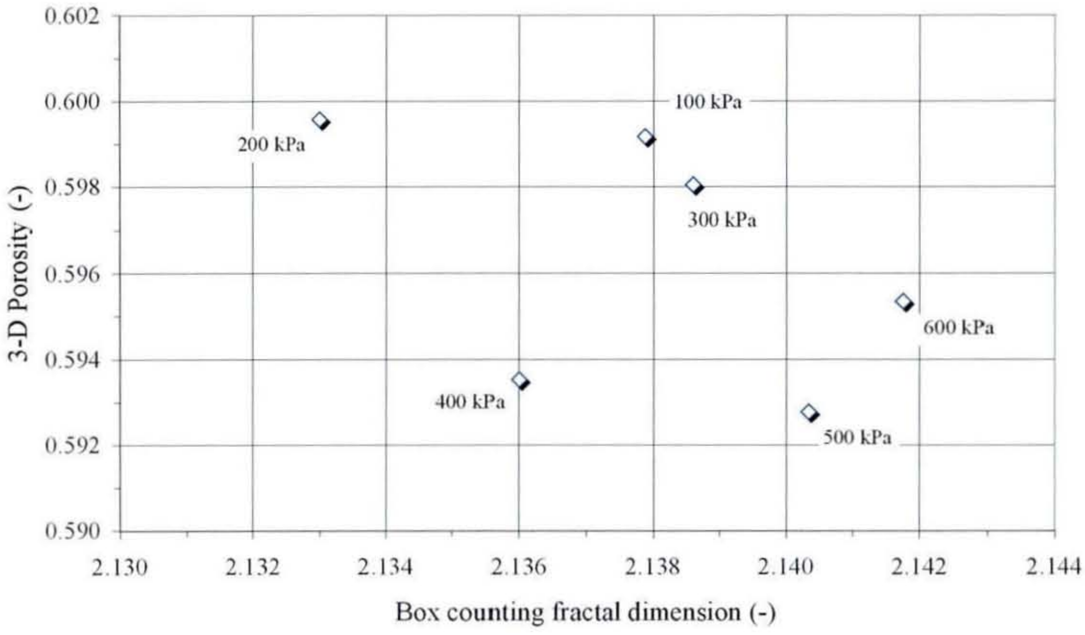


Figure 6.17 - Graph showing relationship between box counting fractal dimension and actual cake porosity for calcite suspensions filtered at constant pressure between 100 and 600 kPa

6.3.2 Physical analysis

In addition to the fractal analysis of the cake structures, their physical characteristics were also measured in the form of a two dimensional porosity. This involved the digitisation of the cake image (as described in Section 6.1) followed by the calculation of the area of 'empty' space within the structure as a percentage of the whole. The computer program completed this task by evaluating the ratio of the number of pixels containing information (material) to the overall number of pixels in the image. The 'digital' porosity yielded by this method varied from the porosities measured in Chapter 5 in that the digital porosities do not represent the fraction of void volume in the object (i.e. actual cake porosities), only the fraction of void area in the plane under scrutiny.

It would theoretically be possible to calculate the volume porosity using the measurement of digital porosity by taking infinitely thin slices of the cakes and performing the equivalent of an integration to make the transformation from two to three dimensions.

Figure 6.18 shows the results gained from the measurement of digital porosities within the cake structure. As with the fractal measures, no obvious trend is visible with the change in pressure, with a wide range of scatter.

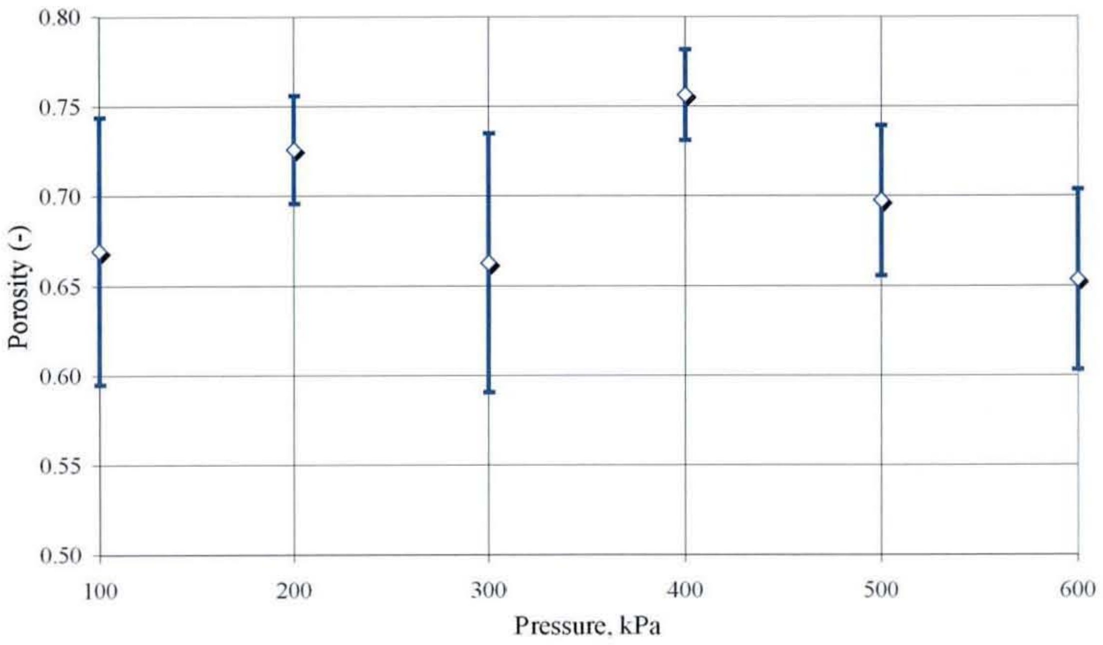


Figure 6.18 – 2-D porosities for calcite filter cake samples, with ± one standard deviation shown as error bars

Figure 6.19 shows the variation in digital porosity with position in the cake sample. It can be seen that there was a slight variation in that the porosities at the top of the cake were overall slightly lower than for those samples taken from the base of the samples. This pattern would concur with the work outlined in Chapter 1 (e.g. Tiller [1953]) that the porosity of filter cakes varies throughout their depth, with decreasing porosity nearer the filter medium due to forces applied above, by both the bulk fluid and solids already in the cake.

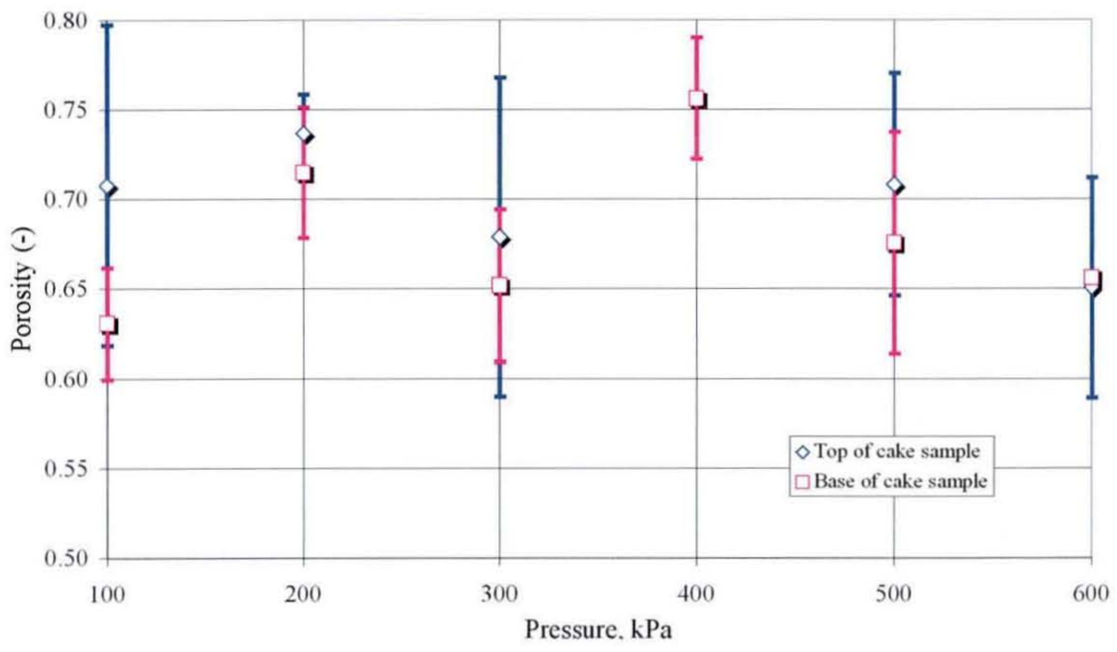


Figure 6.19 - 3-D Porosities showing variation with vertical position within cake samples, with ± one standard deviation shown as error bars

As with the porosities obtained from the experimental trials, no correlations were found between the box counting fractal dimension and the porosities obtained from the analysis of the SEM's taken of the cake samples. Figure 6.20 shows the box counting fractal dimension plotted against the porosities obtained from the SEM's of the cake samples.

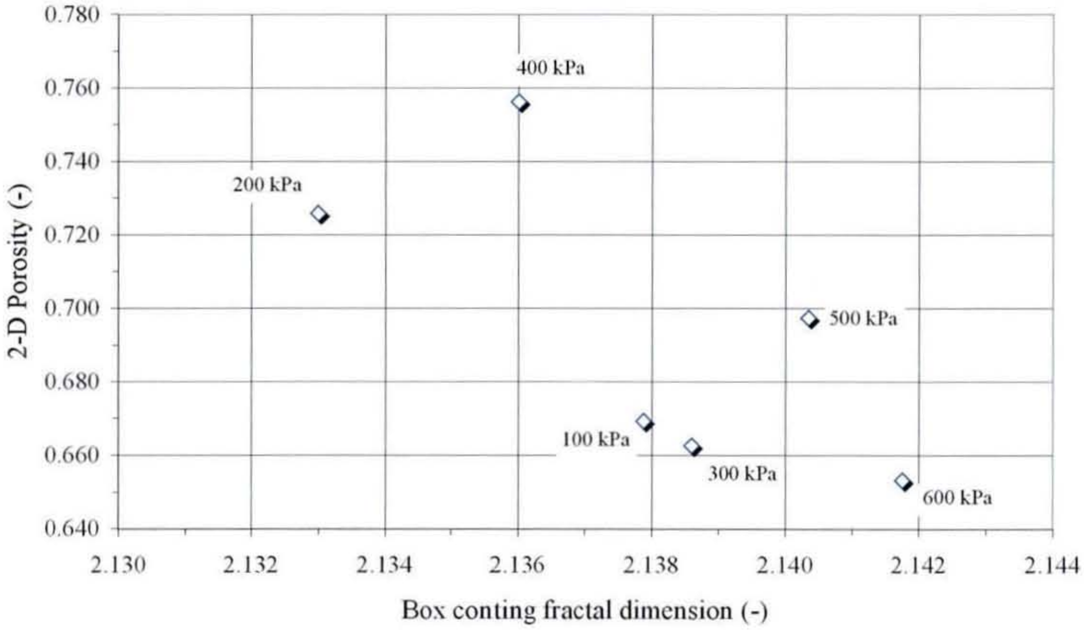


Figure 6.20 - Graph showing relationship between box counting fractal dimension and digital cake porosity for calcite suspensions filtered at constant pressure between 100 and 600 kPa

6.4 Correlation between simulated and experimental results

Due to the nature of fractal characterisation, only certain fractal techniques could be applied to individual systems. For example, systems without a central starting point, or obvious and relevant centre of gravity did not benefit from the measurement of the enclosing circle fractal dimension, as this is better suited to the calculation of fractal measure from constituent particles (e.g. flocs).

This work has attempted to provide an indication of the fractal dimension(s) of various structures and compare them to the physical or Euclidean characteristics exhibited therein. The study of a number of systems, both simulated and experimental has shown that there was not one unifying fractal measurement that could be used to describe both environments.

As such, this work has included the correlation between fractal and Euclidean spatial properties for the individual systems, as well as comparing the Euclidean properties available for both simulated and experimental structures.

6.4.1 Euclidean comparison of simulated and experimental structures

The common, measurable property of both the simulated and experimental structures was the amount of space as a fraction of the whole contained within the structure of the objects under scrutiny, referred to throughout this work as porosity.

The porosities of the four types of simulated structure (two and three dimensional seed agglomerates and two and three dimensional filtration simulations) were analysed to determine the relationship between the parameters used to construct the agglomerates and the characteristics of the final structures.

The results from the measurement of porosity for the simulated agglomerates, previously discussed in this work are repeated here for clarity in order to offer a direct, visual comparison of the data. A full analysis of the results can be found in Chapter 3. Figure 6.21 (Figure 3.39) shows the porosities measured for two dimensional seed agglomerates, whilst Figure 6.22 (Figure 3.40) represents the same measure for agglomerates constructed using the three-dimensional seed model. The two sets of results, whilst showing the same general trend cannot be compared directly, since the porosity was measured in the dimensional space in which the structures were created (i.e. two or three dimensions). These results did however prove that the parameters used to create the agglomerates had a definite effect on the structures formed.

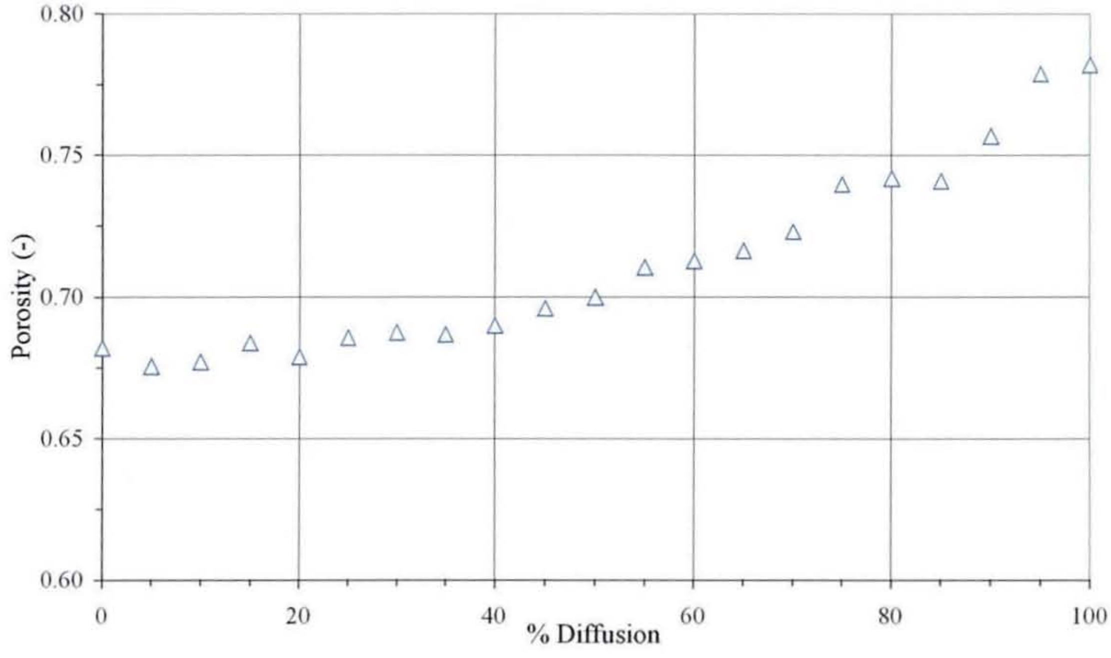


Figure 6.21 - Porosities of 2-D seed agglomerates

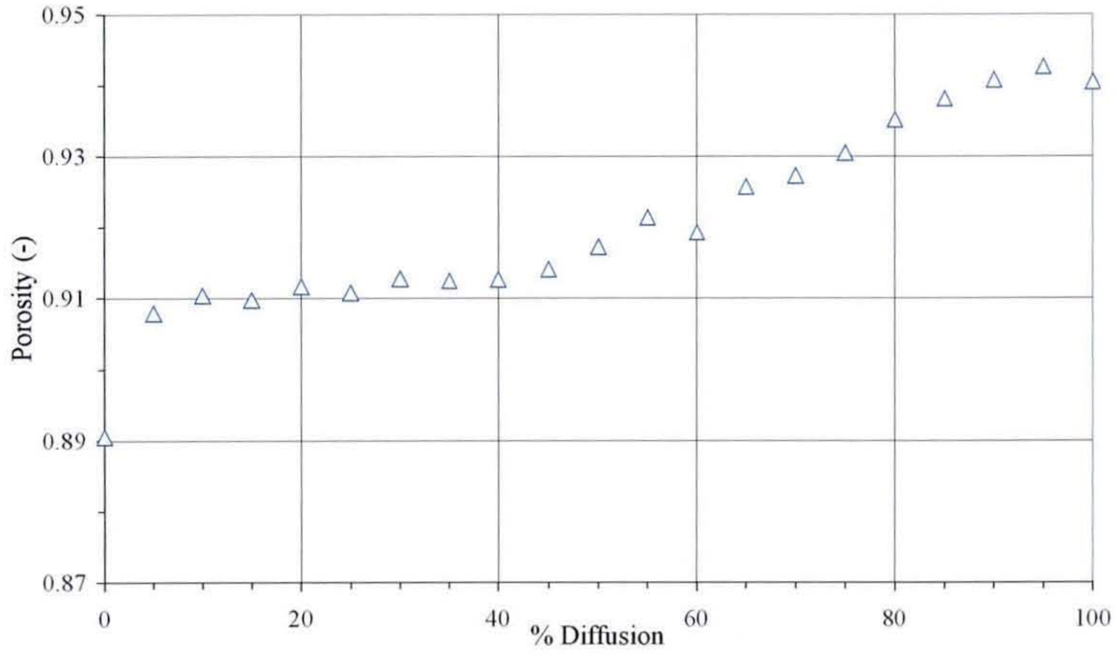


Figure 6.22 - Porosities of 3-D seed agglomerates

The filtration simulations created also showed that the parameters used to control the motion of particles attaching to a growing virtual 'filter cake' had a definite effect on the properties of the structure. Figure 6.23 (Figure 3.71) shows the porosities of the two dimensional filtration simulations, whilst Figure 6.24 (Figure 3.74) shows the porosities for the three dimensional filtration simulations.

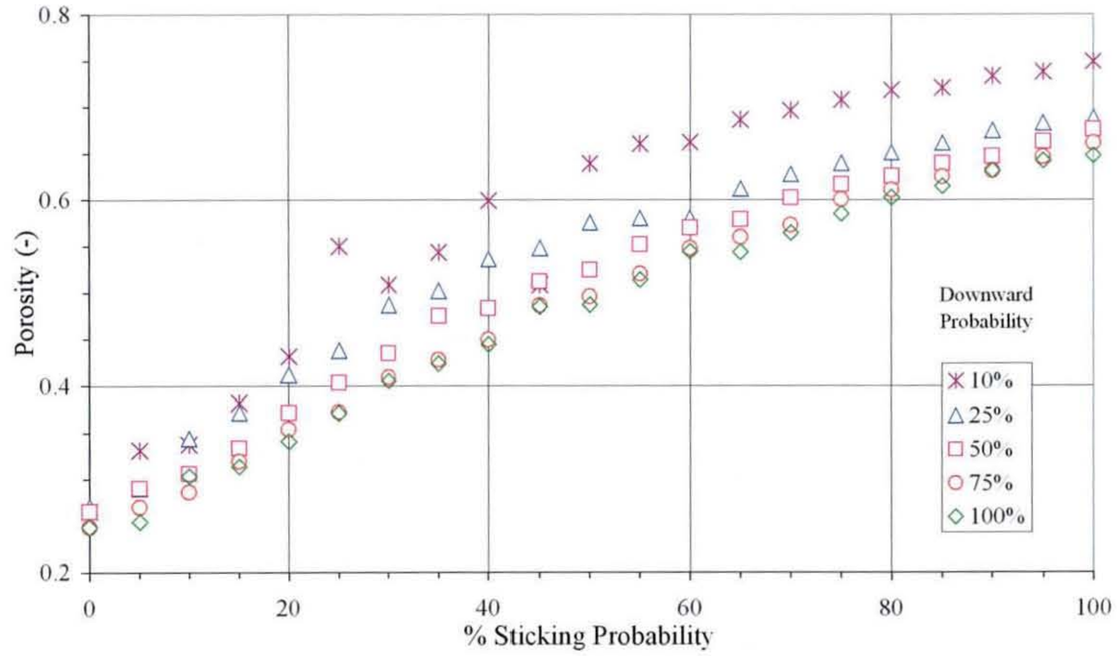


Figure 6.23 - Porosities of 2-D filtration simulations

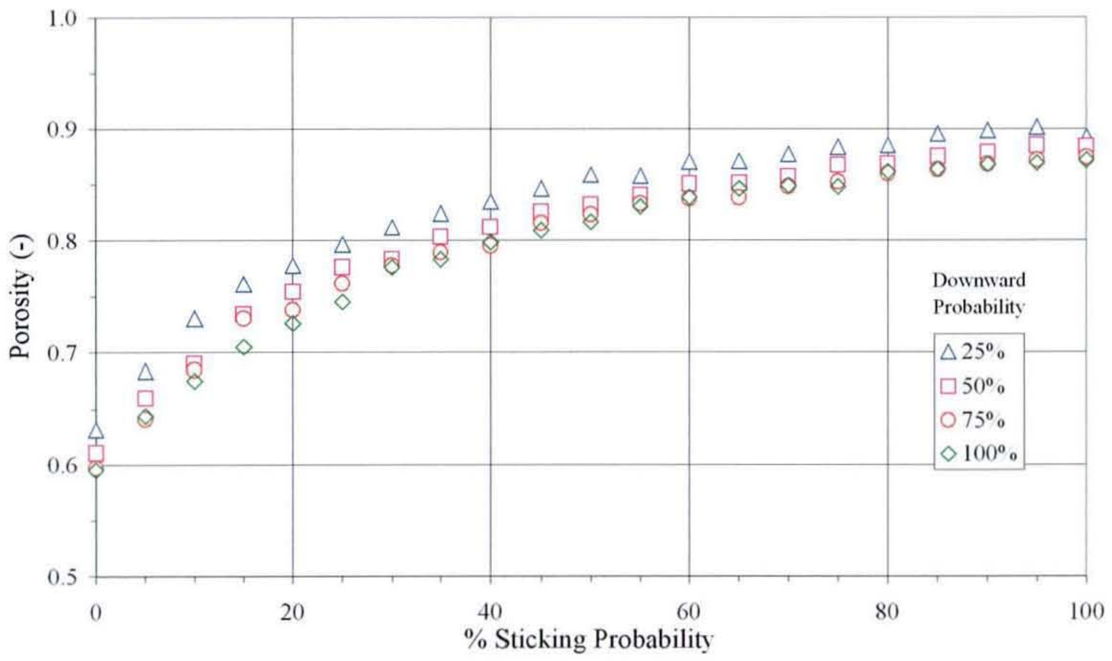


Figure 6.24 - Porosities of 3-D filtration simulations

Figure 6.25 shows that the porosities obtained from the experimental trials were within the same range as those measured for the three dimensional filtration simulations shown in Figure 6.24.

Comparing the results shown in Figure 6.24 (Figure 3.74) and Figure 6.25 (Figure 5.7), it can be seen that the porosities obtained from the experimental work performed with both talc and calcite suspensions showed similarity with the three-dimensional porosities obtained from the analyses of the filtration simulations performed at low sticking probabilities (≤15%). Porosities in both sets of data were in the region of 0.6 to 0.75. The experimental results show that there was not a wide variation of porosity with increased pressure, whilst the porosities measured from the simulations increased with increasing sticking probability and to some extent decreasing downward probability. From these results, it can be surmised that the cakes formed by the filtration of calcite suspensions contained particles which tended not to rearrange within the cake.

The results for talc, on the other hand show that the porosity of the filter cakes did vary with pressure (as discussed in Chapter 5). These results also showed comparisons with the porosities obtained from the filtration simulations. However, in this case, the porosities of the cakes decreased with the increase of pressure drop across the cell. In the case of talc, the results still correlate with the simulations performed using low sticking probabilities, although across a larger range than the results for the calcite filter cakes. These results would tend to suggest that at the higher pressures under which the experiments were performed, the particles fall into a base arrangement, such that no further movement of the particles would have been possible. At the low end of the range of pressures tested, the particles formed arrangements (at least on the basis of porosity comparison) that had more open structures. This cake behaviour was mimicked in the filtration simulations with the porosity following the inverse relationship with sticking probability (i.e. as the sticking probability rose, so did the measured porosity). In the case of the talc experiments, the equivalent sticking probability ranged from 0% ($\bar{\varepsilon} \approx 0.65$) to approximately 20% ($\bar{\varepsilon} \approx 0.75$). These results would tend to suggest that the particles within the talc cakes were able to rearrange within the structure, causing lower porosities at higher pressures. The relationship with regards the sticking probability suggested that the lower sticking probabilities were equivalent to higher filtration pressures, allowing the particles to rearrange within the cakes structure.

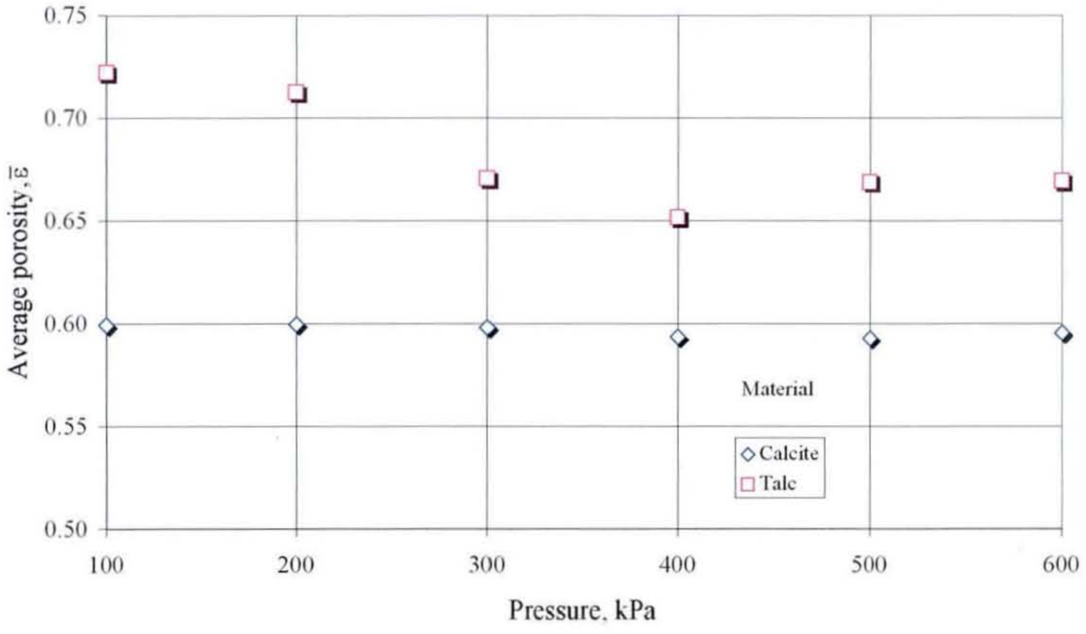


Figure 6.25 - Porosities of filter cakes formed by the dead-end filtration of calcite and talc suspensions at pressures ranging from 100 to 600 kPa

With regards to the variations of porosity within the cakes structure (discussed in Section 6.3), variations were seen within the structure of the simulated as well as experimental filter cakes. Figure 6.26 shows the porosity profile for a typical three-dimensional filtration simulation. As with modern filtration theory, the porosity is shown to decrease with increasing distance away from the filtration surface.

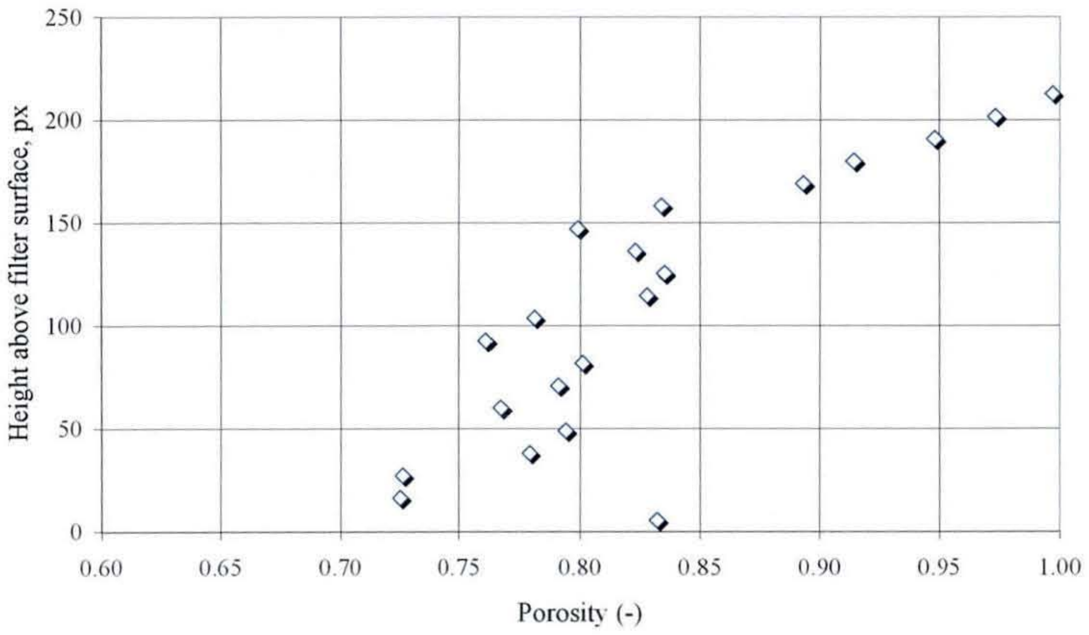


Figure 6.26 - Variation in porosity with cake height shown for a typical 3-D filtration simulation

6.4.2 Fractal comparison of simulated and experimental structures

As mentioned in the introduction to Section 6.4, it was not possible to use one fractal measure on both the simulated and experimental structures. The closest fractal dimensions that were available to compare the two were the three density fractal dimensions, namely enclosing circle / sphere, radius of gyration and box counting dimension. The enclosing circle and radius of gyration measurements were used to classify the density fractal dimensions of the seed agglomeration trials, whilst the box counting fractal dimension was used to classify the fractal dimension of the filtration experiments.

The only fractal measure that was appropriate for the filtration simulations carried out was the measurement of the perimeter of the surfaces of the virtual filter cakes. Unfortunately, due to the time scale and computational power available at the time the simulations were unable to closely model the natural structures formed by the experimental trials. Had this been possible, the box counting dimension could have been applied to slices of the virtual filter cakes formed. The measurement of the surface of the filter cakes formed in the experimental trials may have aided the comparison between experimental and simulated structures, although investigation of slices of the cakes indicated that this would not necessarily have been a valid exercise.

The only fractal data available from the experimental trials was the measurement of the box counting fractal dimension of the cake slices. These results showed little variation with filtration pressure. Had the experimental trials with talc (the compressible cake-forming mineral) been more successful, fractal results may have yielded more information about the structure of the cakes.

As has been shown in Figure 6.26, it is entirely feasible to make measurements of cake properties along the height of the filtration simulation. Due to the nature of the simulations created for this work, and the computational power available, it was not practical for density fractal dimensions of the structures created for the filtration simulations to be measured. Recent advances in computational power might now have made it possible to create filtration simulations that could be characterised using the box counting technique to measure their density fractal dimension. In addition, the characterisation of the individual mineral particles, as well as the cakes formed by the filtration of their suspensions should enable these simulations to be linked more closely to the minerals under investigation with regards to shape characteristics etc.

6.5 Comparison with other fractal measures of filtration

Other authors [Bayles et al, 1987; Bayles, 1988; Bayles et al, 1989; Schmidt, 1995] have used various fractal dimensions to analyse sectioned filter cakes, with varying degrees of success.

Bayles et al [1989] measured the perimeter of particles within the formed filter cakes, to give an equivalent to specific surface area, which they related to the permeability and tortuosity of the sampled cakes. Unfortunately, it was not possible to measure the individual particles within the cake structure, as the boundaries were not well enough defined. This did not, however affect the measurement of the box counting dimension, as this relied upon the distribution of mass within the structure and was thus unaffected by particle boundary resolution.

Schmidt [1995] used the box counting technique (as was used in the current work) to measure the fractal dimension of filter cake structure, but found no correlation between the fractal dimension and the physical characteristics of the cake (specifically looking at the distance from the filter surface) or the mechanism under which it was created (namely pressure). It is Schmidt's work that is most closely followed in the current study, although the current study has found an apparent correlation between the fractal dimension and the distance from the filter bed, as well as for the filtering pressure.

The measurement of the fractal dimension of all but the simplest natural systems (e.g. individual particles) has proven to be a very subjective matter, with workers offering different methods for measuring the same fractal dimension. Other methods involve extrapolating one type of dimension into another, as in the case of applying a one dimensional fractal measure to a two dimensional system when implying that the addition of 1.0 to a perimeter fractal dimension will give a surface fractal dimension [Bayles, 1988].

This work has attempted to use the most widely accepted fractal methods, and those most suited to the measurement undertaken – for example, it was found that the enclosing circle fractal dimension, whilst yielding results, was not suitable for the characterisation of particulate systems.

6.6 Conclusions

The results have demonstrated that it was feasible to measure a number of spatial properties, both fractal and Euclidean, of the minerals and cakes formed from the dead-end filtration of their suspensions, investigated as part of this work. Some of the results obtained from these measurements have also been correlated to the results from the simulations of filter cakes performed using the software written for this work.

Characterisation of the individual mineral particles has shown that the two materials under scrutiny exhibited different shape properties, both fractal and Euclidean. Calcite particles were shown to have a higher average surface fractal dimension (i.e. rougher surface), whilst possessing an aspect ratio closer to unity than the talc particles studied. It is thus quite possible that the characteristic shape of the constituent particles, as measured by the methods described in Section 6.2, could have some bearing on the overall properties of the filter cakes formed during dead-end or other filtration.

With regards to the spatial measurements of the filter cakes formed from the mineral suspensions, only one mineral suspension was characterised. Results showed however, that whilst there was little variation between filter cakes formed at varying pressures, there was variation within the cake structure itself. This variation was illustrated by the difference between the physical characteristics of the samples taken from the top and bottom of the cakes.

These variations would tend to agree with other work performed with regards to 'modern' filtration theory that attempts to quantify the way in which the structure within filter cakes varies with the distance from the base of the cake.

The absence of variation with pressure seen in the characteristics of the sectioned filter cakes would also agree with classical filtration theory, with calcite suspensions forming near-incompressible cakes under dead-end filtration. These conclusions are also supported by the results shown in Chapter 5, with the compressibility index for calcite calculated as 0.08.

It was also shown, although more work is needed to verify the results, that there was a suggestion of a correlation between the fractal dimension as measured from digitised images of sectioned filter cakes and the physical properties of the cake taken from the filtration experiments (specific cake resistance and permeability).

In this particular instance, the fractal characterisation of the cakes under scrutiny yielded only a small amount of information with regards to their internal structure. It has, however, proved that it was possible to produce meaningful fractal measurements of naturally formed (rather than simulated) systems.

Some correlation between the experimental and simulation work was found, although the model used to create the virtual cakes needs to be expanded to provide better comparison with natural systems. Two major areas in which the simulation work could be extended include increasing the size of the virtual filtration cell and including the possibility of shapes other than perfect circles / spheres being used in the simulations. Both of these expansions should be possible with the recent increasing power of computational systems.

More work would now be required to improve the sectioning procedure (described in Chapter 4) in order to obtain further information as to the internal structure of the cakes.

SEM's of both the individual particles and filter cakes produced from their suspensions are shown in Appendix D, along with summaries of their fractal and physical properties.

7 CONCLUSIONS

In recent years, fractal mathematics have been applied to a wide range of physical, numerical and theoretical systems. In general, the self-similarity of fractal objects has been used to provide a novel method by which the properties of the system under scrutiny can be measured.

7.1 Synopsis of work performed

- A computer application was written for the project which was not only able to simulate particle agglomerates in both two and three dimensions but to measure their fractal and physical properties. In addition to these functions, the same application was also able to measure various properties of a range of experimental data, in the form of digitised scanning electron micrographs. A wide range of fractal measures were researched and specific procedures chosen for their suitability (in conjunction with Euclidean and physical measures) in characterising either the simulated structures or the digitised results from the experimental trials.

- Apparatus was constructed in order to perform experiments involving the dead-end filtration of the suspensions of two different minerals selected to provide examples of compressible and incompressible filtration. The results from these experiments were analysed in terms of both the physical and spatial characteristics of the cakes formed, using recognised filtration theory and digital examination techniques. Results demonstrated reliable experimental procedures and rugged computational methodology.

- Measures of the sectioned filter cakes showed an apparent link between experimental conditions and their fractal and Euclidean properties. These results appeared to vary with both the pressure under which the cakes were formed and the position of the sample within the cake.

- Using the computer application written, agglomerates in both two and three dimensions were constructed in order to ascertain the relationship between parameters governing particle behaviour and the fractal and physical properties of the formed structures. These simulations included both seed agglomeration models and structures more closely modelling filtration systems. Results from the measurement of the simulated structures showed that the parameters used to govern the motion of individual particles attaching to a growing agglomerate had a definite effect on the physical and fractal properties of the completed structure.

- Results from the Euclidean analysis of the simulated filter cakes confirmed that the parameters used to control their construction could form virtual cakes with properties similar to those created from experimental trials.

7.2 Further work

- Due to the increase in computational power since the inception of this study, applications can now be written to improve the properties of the particles used to create the virtual agglomerates, in addition to adding motion control algorithms to better imitate their motion inside the virtual filter cell. The particles used for this work were restricted to circular / spherical objects, moving under only two influences. Whilst the results from the simulations showed that the methods by which the agglomerates were created yielded measurable differences in both physical and fractal properties, expansion of the program can more closely relate these to filtration systems. The application programmed for this work was constructed in such a way that with small additions / adjustments, systems other than those outlined here can easily be simulated and measured.

- Improvements in the experimental set-up would include the ability to successfully section more of the length of the cake, and to develop a suitable method by which to sample the cakes formed from the filtration of talc (or other material) suspensions. Both of these goals would include the further development of the drying and sectioning process.

- Whilst the filtration method itself is sound, more results could be obtained by filtering under a wider range of pressures, or indeed by the constant rate, rather than constant pressure filtration of mineral suspensions. The former of these would, for example, only require experimentation in intervals of 50 kPa, rather than 100 kPa as in the current work, whilst the latter would require the upgrading of the data acquisition system to include control of the overall filtration pressure.

7.3 Overall conclusions

This work has attempted to lay the foundations for further study into the fractal properties of filtration systems, using both dead-end and other experimental methods. Whilst only a relatively small number of experiments were performed, there now exists a rugged method by which to measure the fractal dimension of the resulting filter cakes.

This work has also gone some way to mimicking filtration systems and comparing the results of fractal and physical measurements obtained to experimental work performed using a dead-end filtration cell.

8 Nomenclature

Where broad definitions of the units are given (e.g. '[Length]', this refers to the fact that the unit may be measured in S.I. units (e.g. metres) or in the case of computerised analysis other non-physical units (e.g. graphical pixels, 'px').

Superscripts within the definition denote the specific method of calculating the fractal dimension (1,2 and 4), simulation parameters (3) or topic (5):

<u>Key to superscript labels</u>

1	Box counting technique
2	Co-ordinate averaging technique
3	Monte Carlo simulation
4	Structured walk technique
5	Filtration terminology
6	Fracture surface roughness
7	Agglomerate creation

Symbol	Description	Units
A	Cross-sectional area	m^2
A	Apparent fracture surface [6]	m^2
A_t	Area (mass) of agglomerate	m^2 (kg)
Av	Number of neighbours [2]	-
a, b, c	Distances in Figure 2.9	[Length]
b	Size of boxes [1]	[Length]
c	Bulk phase concentration [3]	kmol m^{-3}
c	Concentration in slurry [5]	kg m^{-3}
C	Number of co-ordinates used [2]	-
C_1, C_2	Constants [6]	-
D	Fractal dimension (general usage)	-

Subscripts used for specific types of fractal dimensions

a	Molecular tiling
b	Box counting
e	Enclosing circle / sphere
L	Profile dimension
rg	Radius of gyration
s	Structured walk
S	Surface dimension

D_F	Degrees of freedom of particle motion	°
E_a	Activation energy	J
f	Friction factor	-
f_m	Occupation probability [3]	-
F	Feret diameter	[Length]
G_0	Surface area of a smooth particle	m^2
h	Perpendicular distance	[Length]
h_i	Height of airway	m
H	Characteristic roughness parameter	-
I_g	Moment of inertia	$[\text{Length}]^4$
J_R	Correction factor for specific cake resistance	-
$J(x)$	Pore size distribution function	-
k	Permeability	m^2
k_b	Boltzman constant	$J\ K^{-1}$
K_s	Surface shape factor	-
l	Length (of fractal)	[Length]
L	Steplength [2]	[Length]

L	Depth of filter cake [5]	m
L	True perimeter length [6]	m
L_0	Apparent perimeter length	m
m	Hit rate	-
M	Mass (of fractal)	kg
n	Number of boxes [1] or steps [4]	-
n	Compressibility index [5]	-
N	Number of molecules	-
N_A	Avagadro's number	6.02×10^{23} mol^{-1}
p_c	Probability of adjacent movement [3]	-
Δp	Pressure drop	Pa
P	Length of perimeter [4]	[Length]
P	Pressure [5]	Pa
P_D	Downward probability [3]	-
P_S	Pressure exerted on solids	Pa
Pe	Peclet number	-
Q	Flowrate	m^3 s^{-1}
r, R	Radius	[Length]
R_c	Resistance of filter cake	m kg^{-1}
R_g	Radius of gyration	m
R_m	Resistance of filter medium	m^{-1}

R_S	Material surface roughness	-
s	Sticking probability 3	-
s	Solids weight fraction in feed slurry [5]	-
S	Hausdorff measure of surface	-
S	True fracture surface [6]	m^2
δt	Thickness of strip	m
t	Filtration time	s
T	Absolute temperature	K
V	Volume of filtrate	m^3
V_m	Monolayer coverage	m^2
w	Mass of cake per unit area	kg m^{-2}
$\overline{x}$	Average micropore size	m
x, y, z	Co-ordinate system	-

Greek symbols

α	Specific cake resistance [6]	m kg^{-1}
α, β	Angle subtended between two points [7]	° / Radians
$\overline{\alpha}$	Average specific cake resistance	m kg^{-1}
α_0	Specific cake resistance at unit pressure drop	$m^{(n+1)}$ s^{2n} $kg^{-(1+n)}$
α	Fraction of final steplength [4]	-
φ	Density factor [3]	-
ε	Porosity	-

$\bar{\varepsilon}$	Average porosity	-
κ	'Lacunarity' (constant)	-
λ	Steplength	[Length]
μ	Viscosity	Pa s
η	Level of scrutiny [6]	-
θ	Direction of particle motion	$^\circ$
ρ_s	Solid phase density	kg m^{-3}
σ	Particle cross section	m^2
ω	Profile structure factor [6]	-
Ψ(F)	Random function (zero to D_F)	-

9 References

9.1 Referenced material

Following is a list of articles referenced in the main body of this thesis.

Adler, J; Allen, M, "Obtaining fractal dimensions and examining boundaries by co-ordinate averaging", *Part & Part Systems Characterization*, **11**, 6, 418, 1994

Almy, C; Lewis, WK, "Factors deteriming the capacity of a filter press", *Ind Eng Chem*, **4**, 528, 1912

Antonucci, PL; Barbelri, R; Aricò, AS; Amoddeo, A; Antonucci, V, "Fractal surface characterization of chalcogenide electrodeposits", *Materials Sci & Eng B, Solid State materials for advanced technology*, **38**, 12, 9, 1996

Banik, J; McPherson, MJ; Calizaya, F; MoussetJones, P, "Application of fractal analysis to mine airways", *Proc of the US Mine Ventilation Symposium*, 125, 1993

Bayles, GA, "New pore characterization techniques for flow dynamics in porous media", *Thesis, University of Pittsburgh*, 1988

Bayles, GA; Klinzing, GE: Chiang, S-H, "Fractal mathematics applied to flow in porous systems", *Part & Part Systems Characterization*, **6**, 168, 1989

Bayles, GA; Klinzing, GE; Chiang, S-H, "Determination of the fractal dimensions of coal filter cakes and their relationship to cake permeability", *Part Sci and Technol*, **5**, 4, 371, 1987

Clark, NN; Maeder, AJ; Reilly, S, "Data Scatter in Richardson Plots", *Part and Part Systems Characterization*, **9**, 1, 9, 1992

Coulson, JM; Richardson, JF, "Filtration", *Chem Eng*, **2**, Fourth Edition, 282, Elsevier Science Ltd., 1991

Dubuc, B; Zucker, SW; Tricot, C; Quiniou, J-F; Wehbi, D, "Evaluating the fractal dimension of surfaces", *Proc of the Royal Soc of London - Series A*, **425**, 113, 1989

FE Kruis, J van Dendern, H Buurman, B Scarlett, "Characterization of agglomerated and aggregated aerosol particles using image analysis", *Part & Part Systems Characterization*, **11**, 6, 426, 1994

Feder, J, *Fractals*, Plenum Press, 1988

Fryer, JG; Laurence, SJ, "The fractal analysis of agglomerates comprising particles with irregular shape and size distribution", *Final year R&D project*, Loughborough University of Technology, 1995

Giano, M; Patierno, O, "Monte carlo simulation of aggregation process", *Chem Eng Comm*, **121**, 219, 1993

Gottsleben, F; Hesse, D, "Possibilities and limitations of the fractal description of pore tetxures", *Hungarian J Ind Chem*, **20**, 2, 121, 1992

Gray, WA, *The Packing of Solid Particles*, Chapman and Hall, 1968

Gutsch, A; Pratinis, SE; Löffler, F, "Agglomerate structure and growth rate by trajectory calculations of monomer-cluster collisions", *J Aerosol Sci*, **26**, 2, 187, 1995

Hancock, DJ; Richards, MJ; Adler, J, "Fractal differentiation between wear and contaminant particles found in mining machinery lubrication systems", *Part & Part Systems Characterization*, **10**, 6, 308, 1993

Herd, CR; McDonald, GC; Hess, WM, "Morphology of carbon-black aggregates: Fractal versus euclidean geometry", *Rubber Chem and Technol*, **65**, 107, 1992

Jaroniec, M; Gilpin RJ; Choma J, "Correlation between microporosity and fractal dimension of active carbons", *Carbon*, **31**, 2, 325, 1993

Jiang,Q ; Logan, BE, "Fractal dimensions for aggregates from shear devices", *Journal / American Water Works Association*, **88**, 2, 100, 1996

Jullien, R, "The fractal geometry of aerosol aggregates", *J Aerosol Sci*, **25**, Suppl 1, 301, 1994

Kaye, BH, *A Random Walk Through Fractal Dimensions*, John Wiley & Son Ltd, 1994

Kaye, BH; Trottier, RA; Clark, GC, "Characterizing the fractal structure of fineparticle profiles using the concepts of geometrical probability", *Part and Part Systems Characterization*, **9**, 209, 1992

Kim, HS; Scarlatos, PD, "Fractal dimension of aggregated sediments", *Proc - National Conference on Hydraulic Eng*, **1**, 1184, 1993

Koenders, MA; Wakeman, RJ, "Filter cakes formation from structured suspensions", *Trans IChemE*, **75**, 309, 1997

Kolb, M, "Fractal characterization of porous media: A model calculation", *J Non-Crystalline Solids*, **121**, 227, 1990

Kudoh, M; Hu, X; Ohno, K; Kawazoe, Y, "New cluster-cluster aggregation model for formation process of fractal structure in Sol-Gel transition of SiO_2", *J Crystal Growth*, **128**, 1-4 pt 2, 1162, 1993

Landman, KA; White, LR; Eberl, M, "Pressure filtration of flocculated suspensions", *AIChE Journal*, **41**, 7, 1687, 1995

Larousse dictionary of science and technology, Chambers, 1999

Lenormand, R; Zarcone, C, "Capillary fingering: Percolation and fractal dimension", *Transport in Porous Media*, **4**, 599, 1989

Li, L; Chan, P; Zollinger, DG, "Quantitive analysis of aggregate shape based on fractals", *ACI Materials Journal*, **90**, 4, 357, 1993

Logan, BE; Kilps, JR, "Fractal dimensions of aggreagtes formed in different fluid mechanical environments", *Water Research*, **29**, 2, 443, 1995

Ludlow, DK; Vosen, WM, "Characterization of synthetic-coal char particles using fractal dimension analysis", *Part & Part Systems Characterization*, **10**, 6, 313, 1993

Marner, B; Schmickler, W, "Model for the growth of a two-dimensional cluster", *J Physical Chem*, **93**, 8, 3186, 1989

Meakin, P, "Fractal aggregates", *Advances in Colloid and Interface Sci*, **28**, 4, 249, 1988

Meakin, P, "The effects of reorganization processes on two-dimensional cluster-cluster aggregation", *J Colloid and Interface Sci*, 187, 1986

Meakin, P, "Models for colloidal aggregation", *Annual Review of Physical Chem*, **39**, 237, 1988

Meakin, P; Skjeltorp, A, "Application of experimental and numerical models to the physics of multiparticle systems.", *Advances in Phys*, **42**, 1, 1, 1993

Meeten, GH, "A Dissection method for analysing filter cakes", *Chem Eng Sci*, **48**, 13, 2391, 1993

Namer, J; Ganczarczyk, JJ, "Fractal dimensions and shape factors of digested sludge aggregates", *Water Pollution Research J Canada*, **29**, 4, 4411994

Peleg, M; Normand, MD, "Mechanical stability as the limit to the fractal dimension of solid particle silhouettes", *Powder Technol*, **43**, 187, 1985

Pencea, SC; Dumitrascu, M, "Concentration influence on diffusion limited cluster aggregation", *Materials Research Soc Symposium Proc*, **367**, 373, 1995

Pimienta, C; Dubuc, B; Tawashi, R, "Surface fractal dimension and the quantification of roughness of titanium implant material", *Cells and Materials*, **4**, 4, 379, 1994

Russ, JC, "Measurement of the fractal dimension of surfaces", *J Computer-Assisted Microscopy*, **3**, 3, 127, 1992

Ruth, BF, "Correlating Filtration Theory with Industrial Practice", *Ind Eng Chem*, **38**, 564, 1946

Schmidt, E, "Experimental investigations into the compression of dust cakes deposited on filter media", *Filtration and Separation*, **32**, 8, 789, 1995

Schmidt, E; Löffler, F, "The analysis of dust cake structures", *Part Systems Characterization*, **8**, 105, 1991

Schmidt, E; Löffler, F, "Preparation of dust cakes for microscopic examination", *Powder Technol*, **60**, 173, 1990

Shirato, M; Sambuichi, M, "Theoretical and experimental studies in cake filtration", *Mem Fac Eng*, Nagoya University, **37**, 1, 38, 1985

Shirato, M; Sambuichi, M; Kato, H; Aragaki, T, "Internal flow mechanism in filter cakes", *AIChemE Journal*, **15**, 3, 405, 1969

Stamatakis, K; Tien, C, "Cake formation and growth in cake filtration", *Chem Eng Sci*, **46**, 8, 1917, 1991

Sutherland, DN, "Chain formation of fine particle aggregates", *Nature*, **226**, 1241, 1970

Svarovsky, L, Solid-Liquid Separation, 3rd Edition, Butterworths, 1990

Tarleton, ES; Hancock, DL, "Imaging of filter cakes through electrical impedance tomography", *Filtration and Separation*, **33**, 6, 491, 1996

Tiller, FM, "The role of porosity in flitration, Part 1", *Chem Eng Prog*, **49**, 467,1953

Titus, WJ, "A one-dimensional realization of a general model for cluster-cluster aggregation", *American J Phys*, **57**, 12, 1131, 1989

Trottier, RA, "Fractal Description of Fineparticle Systems", *School of Graduate Studies, Laurentian University*, 1987

Vaughan, NP; Brown, RC, "Observations of the microscopic structure of fibrous filters", *Filtration and Separation*, **33**, 8, 741, 1996

Víquez, S; Rodríguez-Lugo, V; Vázquez,Polo, G; Castaño, VM, "Measuring two-dimensional fractal patterns: The role of the definition of dimension", *Computational Materials Sci*, **4**, 2, 172, 1995

Vold, MJ, "Computer simulation of floc formation in a colloidal suspension", *J Colloid Sci*, **18**, 684,1963

Wakeman, RJ, "A numerical integration of the differential equations describing the formation of and flow in compressible filter cakes", *Trans IChemE*, **56**, 258, 1978

White, J; Yeats, A; Skipworth, G, *Tables for statisticians*, 3rd Edition, Stanley Thornes Ltd., 1991

Willmer, SA; Tarleton, ES; Holdich, RG, "Understanding filter cake formation through electrical impedance measurements", *IChemE Research Event*, **2**, 883, 1995

Witten, TA; Sander, LM, "Diffusion limited aggregation, a kinetic critical phenomenon", *Physical Review Letters*, **47**, 1400, 1981

Xie, H; Bhasker, R; Li, J, "Generation of fractal models for characterization of pulverized materials", *Minerals & Metallurgical Processing*, **10**, 1, 36, 1993

Xiong, Y; Pratsinis, SE, "Formation of agglomerate particles by coagulation and sintering. Part I. A two-dimensional solution of the population balance equation", *J Aerosol Sci*, **24**, 3, 283, 1993

Xu, W; Zerda, TW; Gerspacher, M, "Surface fractal dimension of graphitized carbon black particles", *Carbon*, **34**, 2, 165, 1996

9.2 General bibliography

The following articles, whilst not directly referenced in the text were used in the composition of this thesis.

Avnir, D; Farimand, MD; Pfeifer, P, "Surface geometric irregularity of particulate materials: The fractal approach", *J Colloid and Interface Sci*, **103**, 1, 112, 1985

Bai, D; Shibuya, E; Nakagawa, N; Kato, K, "Fractal characteristics of gas-solids flow in a circulating fluidized bed", *Powder Tech*, **90**, 205, 1997

Bourgeois, FS; Lyman, GL, "Morphological analysis and modelling of fine coal filter cake microstructure", *Chem Eng Sci*, **52**, 7, 1151, 1997

Chan, SK; Ng, K, "Geometrical characteristics of a computer generated three dimensional packed column of equal and unequal sized spheres - With special reference to wall effects", *Chem Eng Comm*, **48**, 215, 1986

Chellam, S; Wiesner, MR, "Fluid mechanics and fractal aggreagtes", *Water Research*, **27**, 9, 1493, 1993

Gagnepain, JJ; Roques-Carmes, C, "Fractal appproach to two-dimensional and three-dimensional surafce roughness", *Wear*, **109**, 119, 1986

Hasegawa,M; Liu, J; Okuda, K; Nunobiki, M, "Calculation of the fractal dimensions of machined surface profiles", *Wear*, **192**, 1-2, 40, 1996

Hwang, K-J; Lu, W-M, "Dynamic analysis on constant pressure filtration of power law slurry", *J Chem Eng of Japan*, **29**, 1, 65, 1996

Kamijo, M; Nakazawa, M; Shimizu, Y, "Idenfication of complicated shape objects fractal charcteristics variables - Categorizing dust particles on LSI wafer surface", *Systems and computers in Japan*, **27**, 6, 82, 1996

Kaye, BH; Clark, GG, "Fractal description of extra terrestrial fineparticles", *Part Characterization*, **2**, 143, 1985

Kaye, BH; Clark, GG; Kydar, Y, "Strategies for evauating boundary fractal dimensions by computer-aided image-analysis", *Part and Part Characterization*, **11**, 6, 411, 1994

Kaye, BH; Clark, GG; Leblanc, JE; Trottier, RA, "Image analysis procedures for characterizing the fractal dimension of fineparticles", *Part Characterization*, **4**, 2, 63, 1987

Li, DH; Ganczarczyk, JJ, "Fractal geometry of particle-particle aggregates generated in water and wastewater treatment processes", *Environmental Sci and Tech*, **23**, 11, 1385, 1989

Majumdar, A; Tien, CL, "Fractal characterization and simulation of rough surfaces", *Wear*, **136**, 313, 1990

Möhler, O; Nauman, K-H; Schöck, W, "Dynamic behaviour of fractal soot aerosols", *J Aerosol Sci*, **25**, Suppl 1, 303, 1994

Nigmatulin, RR; Dissado, LA; Soutougin, NN, "A fractal pore model for archie's law in sedimentary rocks", *J Phys D: Applied Physics*, **25**, 1, 32, 1992

Nyeki, S; Colbeck, I, "The measurement of the fractal dimension of individual in situ soot agglomerates using a modified millikan cell techique", *J Aerosol Sci*, **25**, 1, 75, 1994

Reeve, R, "A warning about standard errors when estimating the fractal dimension", *Computers and Geophysics*, **18**, 1, 89, 1992

Roberge, PR; Trethewey, KR, "Fractal dimension of corroded aluminium surfaces", *J Applied Electrochemistry*, **25**, 10, 962, 1995

Rushton, AG; Akers, RJ, "The prediction of the mass of a fractal aggregate from its projection in two diemsnions", *1994 IChemE Research Event*, 1184, 1994

Schmidt, E; Löffler, F, "Preparation of dust cakes - Translation from Praparation von Staubkuchen", *Health and Safety Executive (Staub Reinhaltung der Luft*, **49**, 429, 1989

Tanaka, M, "Fracture toughness and crack morphology in indentation fracture of brittle materials", *J Materials Sci*, **31**, 3, 749, 1996

Tory, EM; Church, BH; Tam, MK; Ratner, M, "Simulated random packing of equal spheres", *Canadian J Chem Eng*, **51**, 484, 1973

Walker, AJ; Svarovsky, L., "Development of a computer model for predicting the filtration characteristics of a suspension", *Filtration and Separation*, **31**, 1, 57, 1994

Zerda, TW; Yang, H; Gespader, M, "Fractal dimensions of carbon-black particles", *Rubber Chemistry and Tech*, **65**, 130, 1992

APPENDIX A

Computer program listing

APPENDIX A - PROGRAM LISTING (AS AT 23RD AUGUST 2000)

The units comprising the main program are given here in alphabetical order. Other units that each requires can be found in the *uses* statement at the top of each section.

AGGLOM.PAS - Main controlling program

```
program agglom;

{$N+,E+}
{$M 65520, 0, 163840}

uses Crt,Graph,    disk,funcs,look,make,menus,screeninfo,variables;

begin
     {************************DEFAULTS************************}
     {           Other defaults setup in 'MENUS.PAS'           }
     {}NumGen:='On'; Debug:='Off'; Wait:=10; Dimensions:=3;    {}
     {}Auto:='Manual'; Runs := 20; N:=1;                       {}
     {}Shape:='Circle'; Model:='F'; Diff_Frac:=50;             {}
     {}Fl:=470; LWall:=10; RWall:=440; Roof:=40;               {}
     {}DownProb:=75; StickProb:=75;                            {}
     {}Sides:=4; Quality:='Reg'; Rotate:=0;                    {}
     {}                                                        {}
     {       Comment out unwanted simulation size below:       }
     {}Number:=10;  MinSize:=50; Maxsize:=50; seed:=50;        {}
     {}Number:=80;  MinSize:=20; Maxsize:=20; seed:=20;        {}
     {}{Number:=1000; MinSize:=15; Maxsize:=15; seed:=10;       {}
     {}{Number:=800; MinSize:=10; Maxsize:=10; seed:=10;        {}
     {}Number:=1999; MinSize:=8; Maxsize:=10; seed:=5;          {}
     {}{Number:=800; MinSize:=3; Maxsize:=3; seed:=3;          {}
     {}                                                        {}
     {}Root:=CurrDir+'\'; Name:='agglom';                      {}
     {}If Root='C:\TP\TEST\' then Root:=CurrDir+'\DATA\';      {}
     {}If Root='D:\TP\TEST\' then Root:=CurrDir+'\DATA\';      {}
     {}RData:=Root;
     {}Save:=Root+I2S(Dimensions)+'D\'+Name+'.dat';            {}
     {}PoreFile:=Root+'MONTE\'+Name+'.por';                    {}
     {}POV:=Root+'3D\povray\'+Name+'.pov';                     {}
     {}GYR:=Root+'3D\'+Name+'.gyr';                            {}
     {}DimTxt:=I2S(Dimensions)+' Dimensions'; Twist:='On';     {}
     {}RayHeight:=180; RayWidth:=240; Frames:=10;              {}
     {}FileType:='M';                                          {}
     {}Method:='P'; Fraction:=0.1; Ripple:=20; Convert:=990;   {}
     {}Range:='Yes'; LowFrac:=0.02; UppFrac:=0.08; Gap:=0.005; {}
     {}Range:='Yes'; LowFrac:=0.08; UppFrac:=0.32; Gap:=0.02;  {}
     {}View:='F'; WalkDir:='R'; PoreMeth:='P'; PropMeth:='S';  {}
     {}EncMeth:='B';                                           {}
     {************************DEFAULTS************************}

     CheckGraph; {Check for Graphics Drivers}
     InitGraph(GraphDriver, GraphMode,Path);
     CloseGraph;

 {***************************MAIN PROGRAM BLOCK*****************************}
 repeat
        N:=1;
        PoreMeasure:=False;
     MakeWindow(9);
     TextColor(White);
     Writeln('                 Main Menu');
```

```
    Writeln;
    TextColor(Green);
    Writeln(' 1.  Build Agglomerate');
    Writeln(' 2.  Display Stored Agglomerate');
    Writeln(' 3.  Analyse Agglomerate');
    Writeln(' 4.  Default Settings');
    Writeln(' 5.  Ray Trace Settings');
    Writeln;
    Writeln(' D   DOS Shell');
    Writeln(' R   Rename Automatic files');
    Writeln(' X   Exit Program');
    Window(6,17,70,19);
    Write('Enter option : ');
    option:=upcase(Readkey);
    Write(option);
    Delay(200);
    ClrScr;
    Window(16,3,62,11);

    Case option of
      '1' : Simulate;
               '2' : Display;
      '3' : Analyse;
      '4' : Setup;
      '5' : Trace_Menu;
      'D' : DosShell;
      'R' : Ren;
      'X','0' : Exit;
    end;
    if Return then Return:=false;
    until false;
end.
```

BOXCOUNT.PAS - Routine for calculation of the box counting fractal dimension.

```
unit boxcounting;

interface
{$N+,E+}
uses Crt,Graph, disk, funcs, mover, screeninfo, swalk, variables;

procedure SEMBox;

implementation
{**************************************************************************}
procedure InfoBox;

begin
   SetColor(LightGreen);
   SetTextJustify(CenterText,TopText);
   OutTextXY(554,10,'Analyzing...');
   SetTextJustify(LeftText,TopText);
   SetLineStyle(0,0,1);
   Rectangle(485,30,622,160);
   SetTextJustify(CenterText,TopText);
   OutTextXY(559,35,Title);
   SetTextJustify(LeftText,TopText);
   SetColor(Green);
   OutTextXY(490,55,'Img. Height :');
   OutTextXY(490,75,'Img. Width :');
   OutTextXY(560,55,R2S(HImage,7,0));
   OutTextXY(560,75,R2S(WImage,7,0));
   OutTextXY(490,95,'Fraction :');
   OutTextXY(490,115,'Box Size :');
   OutTextXY(490,135,'Boxes to go :');
end;
{**************************************************************************}
procedure ResultBox;
begin
     SetTextJustify(CenterText,TopText);
     OutTextXY(554,35,Title);
     OutTextXY(554,155,chr($AF)+' D = '+R2S(-linb,1,3)+' '+chr($AE));
     SetColor(Green);
     SetTextJustify(CenterText,TopText);
     Rectangle(485,190,622,204);
     SetFillStyle(SolidFill,Black);
     FloodFill(553,200,Green);
     if (R2S(LowFrac,1,3)='0.080') and (R2S(UppFrac,1,3)='0.320')
        and (R2S(Gap,1,3)='0.020') then
     OutTextXY(553,190,'Default Range') else
     OutTextXY(553,190,R2S(lowfrac,1,3)+' '+chr($AF)+' '+R2S(uppfrac,1,3)
               +' @ '+R2S(Gap,1,3));
     if (not multi) or (Debug='On') then Continue;
end;
{**************************************************************************}
procedure Boxes;

var
   I, Pix, Code : integer;
   Fig : char;
   Found : String;
   EndX, EndY : longint;
   Mass : Array [1..50] of Single;
   N, Top, Left, Across, Down, X, Y, Done : Integer;
```

```
begin
     If WImage >= HImage then
        begin FeretMax:=WImage-RB-LB; FeretMin:=HImage-TB-BB end
     else
        begin FeretMax:=HImage-TB-BB; FeretMin:=WImage-RB-LB end;
     if Range='Yes' then LowFrac:=LowFrac-Gap;
     Increments:=round((UppFrac-LowFrac)/Gap);
     InfoBox;

For N:=increments downto 1 do begin

     If Debug='On' then begin SemDraw; Infobox end;

     {Set size of box (steplength) by Feret fraction}
     Fraction:=LowFrac+(N*Gap);
     Steplength[N]:=Fraction * FeretMin;

     Across:=Trunc(WImage/Steplength[N]);
     Down:=Trunc(HImage/Steplength[N]);

{     if Across*Down>200 then OutTextXY(560,135,'   > 200');}

     If Debug='On' then begin
        SetColor(LightMagenta);
        for X:=0 to Across do
            line(Trunc(X*Steplength[N])+LB+ExtraX,TB+ExtraY-1,

Trunc(X*Steplength[N])+LB+ExtraX,Trunc(Down*Steplength[N])+TB+ExtraY-1);
        for Y:=0 to Down do
            line(LB+ExtraX,Trunc(Y*Steplength[N])+TB+ExtraY-1,

Trunc(Across*Steplength[N])+LB+ExtraX,Trunc(Y*Steplength[N])+TB+ExtraY-1);
        end;

     X:=0;  Y:=0;

     SetColor(Green);
     OutTextXY(560,95,R2S(Fraction,8,3));
     OutTextXY(560,115,R2S(Steplength[N],7,2));

     SetColor(LightBlue);
     Mass[N]:=0; Done:=0;

     repeat
     repeat
     Inc(Done);
     SetColor(Green);
{     if (Across*Down-Done)<200 then begin
        DelText(560,135,'   > 200');}
        OutTextXY(560,135,R2S(Across*Down-Done,7,0));
{         end;}
     Assign(Storage,save);
     Reset(Storage);
     Found:='No';
     {Move down}
     Top:=(TB-1)+Trunc(Y*Steplength[N]);
     for J:=1 to Top do Readln(Storage);
         for J:=Top to Top+Trunc(Steplength[N]) do begin
             {Move right}
             Left:=LB+Trunc(X*Steplength[N]);
```

```
              for K:=1 to Left do Read(Storage,Fig);     {Skips outside RadEnc}
              for K:=Left to Left+Trunc(Steplength[N]) do begin
              {Only go through routine if no mass has been found yet}
              if Found='No' then begin
                  Pix:=GetPix;
                  if (Pix=1) and (Found='No') then begin
                     Mass[N]:=Mass[N]+1;
                     Found:='Yes';
                     end;                        {End found pixel}
              if (Debug='On') and (Pix=1) then
PutPixel(K+ExtraX+LB+1,J+ExtraY,LightBlue);
              end;
              end;                               {End left to right}
          Readln(Storage);
          end;                                    {End top to bottom}
     Close(Storage);
     If Debug='On' then delay(Wait);
{      if (Across*Down-Done)<200 then }DelText(560,135,R2S(Across*Down-
Done,7,0));
     Inc(Y);
     until Y=Down;
     Y:=0;
     Inc(X);
     until X=Across;

     SetColor(Green);
     DelText(560,95,R2S(Fraction,8,3));
     DelText(560,115,R2S(Steplength[N],7,2));
     DelText(560,135,R2S(Perim[N+1],7,2));
end;                                            {END N:=1 to increments}

     for N:=Increments downto 1 do begin
     StepLength[N]:=ln(StepLength[N]); Mass[N]:=ln(Mass[N]);
     end;

     if Range='Yes' then LowFrac:=LowFrac+Gap;

     SetTextJustify(CenterText,TopText);
     DelText(554,10,'Analyzing...');

     if (not multi) or (Debug='On') then Continue;

     if Range='Yes' then begin
     J:=1;

     GraphIt(Increments,1,'Box Counting','ln Box Size','ln
Mass',Steplength,Mass);

     ResultBox;
     end;
end;                                 {End 'Steps'}
{**************************************************************************}
procedure SEMBox;

begin
     Boxes;
end;
{**************************************************************************}
end.
```

CARLOMOV.PAS - Routine for movement of particles under Monte Carlo simulation.

```
unit carlomove;

{$N+,E+}

interface

uses Crt,Graph,funcs,mover,rolling,screeninfo,variables;

var
   Xend,Yend,Zend : integer;
   Roll1,Roll2,RollFrom,RollDir : integer;
   XDist,YDist,Zdist : single;
   Xbest,Ybest,Zbest,mindist : single;
   ZRad : single;
   Distance : single;
   FlDist : single;

procedure Roll2D(Q : integer);
procedure CarloStart;
procedure FreeFall(D,Q,step : integer);
procedure CarloStep(D,Step : integer);
procedure CTest2D(I,J : integer);
procedure CTest3D(I,J : integer);
procedure CarloAttach2D(I : integer);
procedure CarloAttach3D(I : integer);

implementation
var
   numtests : integer;
   DeltaX,DeltaY,DeltaZ : single;
   A,B,C,Alpha,Beta,Gamma : single;
   Contact, ContactX, ContactZ : Single;
   Roll : Array [1..4] of Integer;

{***************************************************************************}
procedure RollTest2D(X,Y : single; Q : integer);

var
   K,Best,d : integer;
   BestFix : single;

begin
     Rest:=False;
     BestFix:=4*maxsize;

     Rest:=((Y+Size[I])>=Fl)
     or ((X-Lwall)<=Size[I]+1)
     or ((RWall-X)<=Size[I]+1);

     if rest then begin
        if (Y+Size[I])>=Fl then condition:='b';         {'b' for bottom}
        if ((X-Lwall)<=Size[I]+1)
        or ((RWall-X)<=Size[I]+1) then condition:='w';  {'w' for wall}
        exit;
        end                                             {end rest}

     else for K:=I-1 downto 1 do begin
          d:=Size[I]+Size[K];
          if (K<>Q) and (K<>Roll1) and (K<>Roll2) then begin
          XDist:=X-PosX[K]; YDist:=Y-PosY[K];
          Distance:=sqrt(sqr(XDist)+sqr(YDist));
          if (Distance<=d) then begin
             Rest:=True;
             if Distance<BestFix then begin
                BestFix:=Distance;
                Best:=K
                end;                              {End distance<BestFix}
             end;                                 {End distance<d}
          end;                                    {End K<>Q & K<>Roll1}
     end;                                         {End for K:=1 to I-1 (not
rest)}

     if Rest then begin
        K:=Best;
        Condition:='p';
        Roll2:=Roll1;
        Roll1:=Q;
        RollFrom:=K;
        if ((X>PosX[K]) and (RollDir=-1))
           or ((X<PosX[K]) and (RollDir=1))
           or (X=PosX[K]) then Condition:='n';   {'n' for nested}
        end;                                      {End rest}
end;                                              {End procedure}
{****************************************************************************}
procedure Roll2D(Q : integer);

var
   K,X,Y : single;

const M=0.01;

begin
     NumTests:=0;
     Rest:=False;
     Combined:=Size[I]+Size[Q];
     DeltaY:=PosY[Q]-PosY[I];
     if PosX[I]<>PosX[Q] then begin
        if PosX[I]>PosX[Q] then RollDir:=1          {Clockwise}
        else RollDir:=-1;                           {Anti-Clockwise}
        DeltaX:=RollDir*(PosX[I]-PosX[Q]);
        Contact:=RollDir*(Pi/2-arctan(DeltaY/DeltaX));
        K:=Contact-(RollDir*M);
        repeat
            K:=K+(RollDir*M);
            X:=PosX[Q]+Sin(K)*Combined;
            Y:=PosY[Q]-Cos(K)*Combined;
            if Debug='On' then begin
               PutPixel(round(X),round(Y),Green+RollDir);
               Delay(wait);
               end;
            if NumTests>0 then RollTest2D(X,Y,Q);
            numtests:=numtests+1;
            until Rest or ((K*RollDir)>(Pi/2));
            If not rest then Condition:='f';
     PosX[I]:=round(X); PosY[I]:=round(Y);
     end                                  {End Contact <> 0}
     else condition:='n';
end;
{****************************************************************************}
procedure CarloStart;
```

```
var
   L, R : integer;

begin
     L:=LWall+Size[I]+1; R:=RWall-Size[I]-1;
     CellRad:=(R-L) div 2;                                  {Define CellRadius}
     CellCen:=L+CellRad;                                    {Define CellCentre}
     Ystart:=Roof+Size[I];
     if Dimensions=3 then begin
     repeat
        XStart:=round(L+Random(CellRad*2));
        ZStart:=round(L+Random(CellRad*2));
        XDist:=abs(XStart-CellCen);
        ZDist:=abs(ZStart-CellCen);
        Distance:=sqrt(sqr(XDist)+sqr(ZDist));
        until Distance<CellRad;
     end
     else begin
          XStart:=L+Random(CellRad*2);
          end
end;
{***************************************************************************}
procedure FreeFall(D,Q,step : integer);

var
   Dist,Direction,Angle,Swing : single;
   XBound, YBound, ZBound : boolean;

begin
     if Debug='On' then begin
        SetColor(LightMagenta);
        Circle(Xstart,Ystart,Size[I]);
        line(Xstart,Ystart,Xend,Yend);
        end;

     If D=3 then begin
         Xend:=Xstart-XRoll;
         Yend:=Ystart;
         Zend:=Zstart-ZRoll;
         flag:=false;
         Xin:=Xstart; Yin:=Ystart; Zin:=Zstart;    {Sets 'Test' parameters}
         Xout:=Xend; Yout:=Yend; Zout:=Zend;
         Xstart:=Xend; Ystart:=Yend; Zstart:=Zend;
         for J:=1 to I-1 do CTest3D(I,J);
         if flag then begin
            condition:='n'; Rest:=True; end;
         XDist:=XEnd-CellCen; ZDist:=ZEnd-CellCen;
         Dist:=sqrt(sqr(XDist)+sqr(ZDist));
         if Dist>=CellRad then begin
            condition:='w'; Rest:=true; end;
         end
     else begin
         Xend:=Xstart+RollDir;
         Yend:=Ystart;
         flag:=false;
         Xin:=Xstart; Yin:=Ystart;          {Sets 'Test' parameters}
         Xout:=Xend; Yout:=Yend;
         Xstart:=Xend; Ystart:=Yend;
         for J:=1 to I-1 do CTest2D(I,J);
         if flag then begin
```

```
          condition:='n'; Rest:=(True); end;
          if (Xend=LWall+Size[I]) or (Xend=RWall-Size[I]) then begin
          condition:='n'; Rest:=true; end;
          end;

         if Debug='On' then begin
         delay(Wait);
         SetColor(Black);
         Circle(Xstart,Ystart,Size[I]);
         end;

      Xin:=Xstart; Yin:=Ystart;          {Sets 'Test' parameters}
      Xout:=Xend; Yout:=Yend;
      if D=3 then begin Zin:=Zstart; Zout:=Zend; end;

      Xstart:=Xend; Ystart:=Yend;
      if D=3 then Zstart:=Zend;

end;
{****************************************************************************}
procedure CarloStep(D,step : integer);

var
   Direction,Angle,Elev,Depth,Slice : single;
   Swing : single;
   InBound, XBound, YBound, ZBound : boolean;

begin
      repeat
      Swing:=(100-DownProb)/100*360;
      Direction:=(Random*Swing-Swing/2+90)/DegRad;
      Yend:=round(Ystart + step * Sin(Direction));
      If D=3 then begin
         Beta:=random*360/DegRad;
         W:=step * Cos(Direction);
         ZEnd:=round(Zstart+W*sin(Beta));
         XEnd:=round(Xstart+W*cos(Beta));
         XDist:=abs(CellCen-Xend); ZDist:=abs(CellCen-Zend);
         Distance:=sqrt(sqr(XDist)+sqr(ZDist));
         InBound:=Distance<=CellRad;
         YBound:=(Yend-Size[I])>Roof;
         end
      else begin
         Xend:=round(Xstart + step * Cos(Direction));
         Inbound:=(Xend-Size[I]>LWall) and (Xend+Size[I]<RWall);
         YBound:=((Yend-Size[I])>Roof);
         end;
      until InBound and YBound;

      if Debug='On' then begin
         SetColor(Green);
         line(Xstart,Ystart,Xend,Yend);
         if Wait>20 then begin
            Circle(Xstart,Ystart,Size[I]);
            delay(Wait);
            SetColor(Black);
            Circle(Xstart,Ystart,Size[I]);
            end;
         end;

      Xin:=Xstart; Yin:=Ystart;          {Sets 'Test' parameters}
      Xout:=Xend; Yout:=Yend;
      Xstart:=Xend; Ystart:=Yend;
      if D=3 then begin Zin:=Zstart; Zout:=Zend; Zstart:=Zend; end;

end;
{*************************************************************************}
procedure CTest2D(I,J : integer);

begin
      Floor:=False;
      Hit[J]:=False;

      if (I>1) and (I<>J) then begin
         XDist:=Xout-PosX[J]; YDist:=Yout-PosY[J];
         Distance:=sqrt(sqr(XDist)+sqr(YDist));
         Combined:=Size[I]+Size[J];
         Hit[J]:=(Distance<=Combined);
         end;

      YDist:=Fl-Yin;
      if (Yout<>Yin) then XDist:=(Xin-Xout)*(YDist/(Yout-Yin))
      else XDist:=(Xin-Xout);
      {X distance when floor is hit}
      FlDist:=sqrt(sqr(XDist)+sqr(YDist));

      if Hit[J] then Floor:=((Yout+Size[I])>=Fl)and(FlDist<Distance)
      else Floor:=(Yout+Size[I])>=Fl;

      if Hit[J] and (not floor) then NotFloor:=True;

      if not flag then flag:=Hit[J] or Floor;
end;
{*************************************************************************}
procedure CTest3D(I,J : integer);

begin
      Floor:=False;
      Hit[J]:=False;

      if (I>1) and (I<>J) then begin
         XDist:=Xout-PosX[J]; YDist:=Yout-PosY[J]; ZDist:=Zout-PosZ[J];
         Distance:=sqrt(sqr(XDist)+sqr(YDist)+sqr(ZDist));
         Combined:=Size[I]+Size[J];
         Hit[J]:=(Distance<=Combined);
         end;

      YDist:=Fl-Yin;
      if (Yout<>Yin) then begin
         XDist:=(Xin-Xout)*(YDist/(Yout-Yin));
         ZDist:=(Zin-Zout)*(YDist/(Yout-Yin));
         end
      else begin
           XDist:=(Xin-Xout);
           ZDist:=(Zin-Zout);
           end;

      {X distance when floor is hit}
      FlDist:=sqrt(sqr(XDist)+sqr(YDist)+sqr(ZDist));

      if Hit[J] then Floor:=((Yout+Size[I])>=Fl)and(FlDist<Distance)
      else Floor:=(Yout+Size[I])>=Fl;
```

```
     if Hit[J] and (not floor) then NotFloor:=True;
     if Floor then NotFloor:=False;

     if not flag then flag:=Hit[J] or Floor;

end;
{********************MONTE CARLO   A T T A C H   BELOW******************}
procedure CarloAttach2D(I : integer);

var
   Xlen, Ylen : integer;
   Xdir, Ydir : integer;
   X, Y : double;
   Xmove, Ymove : boolean;
   dist, old_dist, best, bestX, bestY : single;
   XFix, YFix, DistFix : single;
   Fraction : single;

begin
     Xlen:=abs(Xin-Xout); Ylen:=abs(Yin-Yout);
     Xmove:= Xlen >= Ylen;
     Ymove:= Ylen > Xlen;

     if Xin >= Xout then Xdir := -1 else Xdir:=1;
     if Yin >= Yout then Ydir := -1 else Ydir:=1;

     DistFix:=4*radius;
     Best:=4*radius;

if not floor then for J:=1 to I-1 do begin

if (hit[J]) then begin

   Dist:=2*radius;
   Combined:=Size[I]+Size[J];
   X:=Xin; Y:=Yin;

if Xmove then begin

   SetColor(Xdir+4);
   repeat
         old_dist:=dist;
         Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J]));
         X:= X + Xdir;
   until ((old_dist > Combined) and (dist <= Combined));
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   X := X - (Xdir*(1+Fraction));
   Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
   end;                    {End Xmove}

if Ymove then begin

   SetColor(Ydir+3);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J]));
         Y:= Y + Ydir;
   until ((old_dist > Combined) and (dist <= Combined));
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   Y := Y - (Ydir*(1+Fraction));
   X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
   end;                    {End Ymove}

   XFix:=X-Xin; YFix:=Y-Yin;
   DistFix:=sqrt(sqr(XFix)+sqr(YFix));

   If DistFix < Best then begin
      Best:=DistFix;
      RollFrom:=J;
      PosX[I]:=Round(X);
      PosY[I]:=Round(Y);
      end;

   end;                    {End 'if Hit[J]... loop}

   end                     {End for J... loop}

else begin                 {Floor ; True}

   Mindist:=4*radius;
   Dist:=2*radius;
   X:=Xin; Y:=Yin;

if Xmove then begin

   SetColor(Xdir+4);
   repeat
         old_dist:=dist;
         Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
         dist:=abs(Y-Fl);
         X:= X + Xdir;
   until ((old_dist > Size[I]) and (dist <= Size[I]));
   Fraction:=(Size[I]-Dist)/(Old_dist-Dist);
   X := X - (Xdir*(1+Fraction));
   Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
   end;                    {End Xmove}

if Ymove then begin

   SetColor(Ydir+3);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
         dist:=abs(Y-Fl);
         Y:= Y + Ydir;
   until ((old_dist > Size[I]) and (dist <= Size[I]));
   Fraction:=(Size[I]-Dist)/(Old_dist-Dist);
   Y := Y - (Ydir*(1+Fraction));
   X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
   end;                    {End Ymove}

   PosX[I]:=round(X);
   PosY[I]:=round(Y);
   end;                    {End floor loop}

end;                    {End Attach - Do not remove - Leave at bottom}
{*******************PROCEDURE   A T T A C H  3 D  BELOW*******************}
procedure CarloAttach3D(I : integer);
```

```
var
   Xlen, Ylen, Zlen : integer;
   Xdir, Ydir, Zdir : integer;
   X, Y, Z : double;
   Xmove, Ymove, Zmove : boolean;
   combined : integer;
   dist, old_dist, best : single;
   XFix, YFix, ZFix, DistFix : single;
   fraction : single;

begin
     Xlen:=abs(Xin-Xout); Ylen:=abs(Yin-Yout); Zlen:=abs(Zin-Zout);
     Xmove:= (Xlen >= Ylen) and (Xlen >= Zlen);
     Ymove:= (Ylen > Xlen) and (Ylen >= Zlen);
     Zmove:= (Zlen > Xlen) and (Zlen > Ylen);

     if Xin >= Xout then Xdir := -1 else Xdir:=1;
     if Yin >= Yout then Ydir := -1 else Ydir:=1;
     if Zin >= Zout then Zdir := -1 else Zdir:=1;

     best:=2*radius;
     DistFix:=4*radius;

if not floor then for J:=1 to I-1 do begin

if hit[J]  then begin

     Dist:=2*CellRad;
     X:=Xin; Y:=Yin; Z:=Zin;
     Combined:=Size[I]+Size[J];

if Xmove then begin
   SetColor(Xdir+4);
   repeat
         old_dist:=dist;
         Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
         Z := Zin + Zdir * abs((X - Xin) * (ZLen/XLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J])+sqr(Z-PosZ[J]));
         X:= X + Xdir;
   until ((old_dist > Combined) and (dist <= Combined));
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   X := X - (Xdir*(1+Fraction));
   Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
   Z := Zin + Zdir * abs((X - Xin) * (ZLen/XLen));
   end;                         {End Xmove}

if Ymove then begin
   SetColor(Ydir+3);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
         Z := Zin + Zdir * abs((Y - Yin) * (ZLen/YLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J])+sqr(Z-PosZ[J]));
         Y:= Y + Ydir;
   until ((old_dist > Combined) and (dist <= Combined));
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   Y := Y - (Ydir*(1+Fraction));
   X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
   Z := Zin + Zdir * abs((Y - Yin) * (ZLen/YLen));
   end;                         {End Ymove}
```

```
if Zmove then begin
   SetColor(Zdir+5);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Z - Zin) * (XLen/ZLen));
         Y := Yin + Ydir * abs((Z - Zin) * (YLen/ZLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J])+sqr(Z-PosZ[J]));
         Z:= Z + Zdir;
   until ((old_dist > Combined) and (dist <= Combined));
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   Z := Z - (Zdir*(1+Fraction));
   X := Xin + Xdir * abs((Z - Zin) * (XLen/ZLen));
   Y := Yin + Ydir * abs((Z - Zin) * (YLen/ZLen));
   end;                         {End Zmove}

   XFix:=X-Xin; ZFix:=Z-Zin; YFix:=Y-Yin;
   DistFix:=sqrt(sqr(XFix)+sqr(YFix)+sqr(ZFix));

   If DistFix < best then begin
      Best:=DistFix;
      RollFrom:=J;
      PosX[I]:=round(X);
      PosY[I]:=round(Y);
      PosZ[I]:=round(Z);
      end;
   end;                      {End 'if Hit[J]... loop}
   end                       {End 'for J... loop}

else begin             {Floor := True}

   Dist:=2*CellRad;
   X:=Xin; Y:=Yin; Z:=Zin;

if Xmove then begin
   SetColor(Xdir+4);
   repeat
         old_dist:=dist;
         Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
         Z := Zin + Zdir * abs((X - Xin) * (ZLen/XLen));
         dist:=fl-Y;
         X:= X + Xdir;
   until ((old_dist > Size[I]) and (dist <= Size[I]));
   Fraction:=(Size[I]-Dist)/(Old_dist-Dist);
   X := X - (Xdir*(1+Fraction));
   Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
   Z := Zin + Zdir * abs((X - Xin) * (ZLen/XLen));
   end;                      {End Xmove}

if Ymove then begin
   SetColor(Ydir+3);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
         Z := Zin + Zdir * abs((Y - Yin) * (ZLen/YLen));
         dist:=fl-Y;
         Y:= Y + Ydir;
   until ((old_dist > Size[I]) and (dist <= Size[I]));
   Fraction:=(Size[I]-Dist)/(Old_dist-Dist);
   Y := Y - (Ydir*(1+Fraction));
   X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
   Z := Zin + Zdir * abs((Y - Yin) * (ZLen/YLen));
   end;                    {End Ymove}

if Zmove then begin
   SetColor(Zdir+3);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Z - Zin) * (XLen/ZLen));
         Y := Yin + Ydir * abs((Z - Zin) * (YLen/ZLen));
         dist:=fl-Y;
         Z:= Z + Zdir;
   until ((old_dist > Size[I]) and (dist <= Size[I]));
   Fraction:=(Size[I]-Dist)/(Old_dist-Dist);
   Z := Z - (Zdir*(1+Fraction));
   X := Xin + Xdir * abs((Z - Zin) * (XLen/ZLen));
   Y := Yin + Ydir * abs((Z - Zin) * (YLen/ZLen));
   end;                    {End Zmove}

   PosX[I]:=round(X);
   PosY[I]:=round(Y);
   PosZ[I]:=round(Z);

   end;                    {End floor loop}

end;
{**************************************************************************}
end.
```

DISK.PAS – Provides all major disk read and write functions

```
unit disk;

{$N+,E+,B-}

interface

uses Crt,Dos,Graph,funcs,screeninfo,variables;

procedure AggStore;
procedure Store3D(PosX,PosY,PosZ,Size : array of integer);
procedure FiltStore;
procedure POVRay;
procedure POVAnim;
procedure Retrieve;
procedure RetrieveGyr;
procedure Fibre;
procedure Fryer;
procedure SEM;
procedure PoreSave;
procedure DosShell;
procedure Ren;
procedure CheckGraph;
procedure CheckSubs;
procedure CreateSub;
procedure CloseData(Ave : Single; Q : integer);
function  CurrDir : String;

implementation

var
   I,Code : integer;
   floor : integer;            {Height of POV-Ray floor (y-axis)}
   Variable : single;
   FMaxZ, FMinZ : single;

{***************************************************************************}
procedure CharRead(m,n,r : integer);

var
   Num : string;
   Fig : char;

begin
Num:='';
For I:=1 to m do read(Storage,Fig);      {Reads 'm' unwanted characters}
For I:=1 to n do begin                   {Reads 'n' characters wanted}
    read(Storage,Fig);
    if (Fig<>' ') and (Fig<>#10) and (Fig<>#13) then Num:=Num+Fig;
    end;
if (r=1) and (Fig<>#13) then Readln(Storage);            {Moves onto next
line}
Val(Num,Variable,Code);
end;
{***************************************************************************}
procedure AggStore;

begin
     DimTxt:=I2S(Dimensions) + ' Dimensions';
     Assign(Storage,save);
    Rewrite(Storage);
    Writeln(Storage,Title);
    Writeln(Storage,DimTxt);
    Writeln(Storage,'Seed : '+TimeStr);
    Writeln(Storage,TimeTxt);
    Writeln(Storage,'Seed Size : '+I2S(Seed));
    Writeln(Storage,'Min. Particle Size : '+I2S(MinSize));
    Writeln(Storage,'Max. Particle Size : '+I2S(MaxSize));
    writeln(Storage,number);
    for I:=1 to number do begin
        writeln(Storage,PosX[I]);
        writeln(Storage,PosY[I]);
        if Dimensions=3 then writeln(Storage,PosZ[I]);
        writeln(Storage,Size[I]);
        end;
    close(Storage);
end;
{***********************************************************************}
procedure store3D(PosX,PosY,PosZ,Size : array of integer);

begin
    DimTxt:=I2S(Dimensions)+' Dimensions';
    Assign(Storage,save);
    Rewrite(Storage);
    Writeln(Storage,Title);
    Writeln(Storage,DimTxt);
    Writeln(Storage,'Timer seed : '+TimeStr);
    Writeln(Storage,TimeTxt);
    Writeln(Storage,'Seed Size : '+I2S(Seed));
    Writeln(Storage,'Min. Particle Size : '+I2S(MinSize));
    Writeln(Storage,'Max. Particle Size : '+I2S(MaxSize));
    writeln(Storage,number);
    for I:=0 to number-1 do
        begin
            writeln(Storage,PosX[I]);
            writeln(Storage,PosY[I]);
            writeln(Storage,PosZ[I]);
            writeln(Storage,Size[I]);
        end;
    close(Storage);
end;
{***********************************************************************}
procedure FiltStore;
begin
    DimTxt:=I2S(Dimensions) + ' Dimensions';
    Assign(Storage,save);
    Rewrite(Storage);
    Writeln(Storage,Title);
    Writeln(Storage,DimTxt);
    Writeln(Storage,'Seed : '+TimeStr);
    Writeln(Storage,TimeTxt);
    Writeln(Storage,SimInfo);
    Writeln(Storage,'Min. Particle Size : '+I2S(MinSize));
    Writeln(Storage,'Max. Particle Size : '+I2S(MaxSize));
    writeln(Storage,number);
    for I:=1 to number do begin
        writeln(Storage,PosX[I]);
        writeln(Storage,PosY[I]);
        if Dimensions=3 then writeln(Storage,PosZ[I]);
        writeln(Storage,Size[I]);
        end;
    Close(Storage);
end;
{**************************************************************************}
procedure POVStore;
begin
case FileType of

    'N' : begin
    for I:=1 to number do begin
        Writeln(Storage,'object { ');
        Write(Storage,' sphere { <');
        write(Storage,I2S(-CentreX+PosX[I])+' ');
        write(Storage,I2S(CentreY-PosY[I])+' ');
        write(Storage,I2S(CentreZ-PosZ[I])+'> ');
        writeln(Storage,I2S(Size[I])+' }');
        Writeln(Storage,'texture { color red 1 phong 1}');
        Writeln(Storage,'}');
        end;
    end;                            {End seed agglomerates}
    'M' : begin
    for I:=1 to number do begin
        Writeln(Storage,'object { ');
        Write(Storage,' sphere { <');
        write(Storage,I2S(PosX[I])+' ');
        write(Storage,I2S(Fl-PosY[I])+' ');
        write(Storage,I2S(PosZ[I])+'> ');
        writeln(Storage,I2S(Size[I])+' }');
        Writeln(Storage,'texture { color red 1 phong 1}');
        Writeln(Storage,'}');
        end;
    end;                            {End monte carlo agglomerates}
    end;                            {End 'case'}
    Writeln(Storage);
    Writeln(Storage,'// Define Floor');
    if FileType= 'M' then begin
        Writeln(Storage,'object {');
        Writeln(Storage,'intersection { Disk_Y');
        Writeln(Storage,'scale <'+I2S(CellRad div 2)+' .5 '
        +I2S(CellRad div 2)+'>');
        Writeln(Storage,'translate <'+I2S(CellCen)+' 0 '+I2S(CellCen)+'> }');
        Writeln(Storage,'texture { color Silver phong 1}');
        Writeln(Storage,'}');
        end;
        Writeln(Storage,'object {');
        Writeln(Storage,' plane { <0 1 0> '+I2S(floor)+' }');
        Writeln(Storage,' texture {color green 1 }');

    Writeln(Storage,'}');
    Close(Storage);
end;
{**************************************************************************}
procedure POVRay;

var
   highest,depth : integer;              {depth of camera}
   Look,light : string;
   Fig : char;

begin
case FileType of
```

```
     'N' : begin
           CofG(number);
           Look:=R2S((CofGX-CentreX),3,0)+' '+R2S((CofGY-CentreY),3,0);
           depth:=round(Diameter(PosX,PosY,PosZ,number)*7/4);      {Allows
camera to see all agglomerate}
           Light:=R2S((CofGX-CentreX+20),3,0)+' '
             +R2S((CofGY-CentreY+20),3,0)+' '
             +I2S(-depth+30);
           camera(round(CofGX),round(CofGY));
           sun(round(CofGX+20),round(CofGY-20));
           POV:=RemLet(Save,3)+'pov';
           floor:=240;
           for I:=1 to number do if PosY[I] > floor then floor:=PosY[I];
           floor:=CentreY-(floor+10);      {Floor is 10 lower than bottom}
           Assign(Storage,POV);
           Rewrite(Storage);
           Writeln(Storage,'// '+Title);
           Writeln(Storage,'// Timer seed : '+TimeStr);
           Writeln(Storage,'// '+TimeTxt);
           Writeln(Storage,'// Seed Size : '+I2S(Seed));
           end;                 {End Seed Agglomerates}

     'M' : begin
           CellCen:=LWall+(RWall-LWall) div 2;
           CellRad:=(RWall-LWall);
           for J:=1 to number do if PosY[J] < highest then highest:=PosY[J];
           {Define X & Y positions for camera & look}
           Look:=I2S(CellCen)+' '+I2S((Fl-Highest) div 2);
           depth:=round(CellRad*0.9);      {Allows camera to see all
agglomerate}
           Light:=I2S(CellCen+20)+' '
             +I2S((Fl-Highest) div 2+20)+' '+I2S(-depth+30);
           camera(CellCen,Fl-Highest div 2);
           sun(CellCen+20,Fl-Highest div 2+20);
           POV:=RemLet(Save,3)+'pov';
           Assign(Storage,POV);
           Rewrite(Storage);
           Writeln(Storage,'declare Cylinder_Y = quadric {');
           Writeln(Storage,'<1.0 0.0 1.0>');
           Writeln(Storage,'<0.0 0.0 0.0>');
           Writeln(Storage,'<0.0 0.0 0.0>');
           Writeln(Storage,'-1.0');
           Writeln(Storage,'}');
           Writeln(Storage,'#declare Disk_Y = intersection {       /* Capped
cylinder, Length in y axis */');
           Writeln(Storage,'quadric { Cylinder_Y  }');
           Writeln(Storage,'plane { <0.0 1.0 0.0> -1 inverse }');
           Writeln(Storage,'plane { <0.0 1.0 0.0>  1 }');
           Writeln(Storage,'}');
           Writeln(Storage,'#declare Silver = color red 0.196078 green 0.6
blue 0.8');
           Writeln(Storage,'// '+Title);
           Writeln(Storage,'// Timer seed : '+TimeStr);
           Writeln(Storage,'// '+TimeTxt);
           end;              {End Monte Carlo Agglomerates}
     end;            {End FileType Case}

     Writeln(Storage,'// Min. Particle Size : '+I2S(MinSize));
     Writeln(Storage,'// Max. Particle Size : '+I2S(MaxSize));
     Writeln(Storage,'// '+I2S(number)+' Particles');
     Writeln(Storage);
     Writeln(Storage,'// Define Camera Position');
     Writeln(Storage,'camera {');
     Writeln(Storage,' location <'+Look+' '+I2S(-depth)+'>');
     Writeln(Storage,' look_at <'+Look+' 0>');
     Writeln(Storage,'}');
     Writeln(Storage);
     Writeln(Storage,'// Define light source');
     Writeln(Storage,'object {');
     Writeln(Storage,' light_source{ <'+Light+'> color red 1 green 0.75 blue
0.75 }');
     Writeln(Storage,'}');
     PovStore;
end;
{****************************************************************************}
procedure POVAnim;
var
   highest,depth : integer;             {depth of camera}
   LookX,LookZ,LookY,LightZ : string;
   X,Y,Z : integer;
   Angle : single;
   Batch, Trunk : string;
   Look,light : string;

begin
     {Delete old animation files}
     SwapVectors;
     Exec(GetEnv('COMSPEC'),' /c c:\tp\test\prepare.bat');
     SwapVectors;
Case FileType of
     'N' : begin
           CofG(number);
           {Allow camera to see all agglomerate}
           depth:=round(Diameter(PosX,PosY,PosZ,number)*7/4);
           camera(round(CofGX),round(CofGY));
           sun(round(CofGX+20),round(CofGY-20));
           floor:=240;
           for I:=1 to number do if PosY[I] > floor then floor:=PosY[I];
           floor:=CentreY-(floor+10);      {Floor is 10 lower than bottom}
           Batch:='c:\tp\test\animate.bat';
           Assign(Instruct,Batch);
           Rewrite(Instruct);
           Writeln(Instruct,'@echo off');
           Writeln(Instruct,'cd data\animator');
           Writeln(Instruct,'SET POVRAYOPT=+W'+I2S(RayWidth)+'
+H'+I2S(RayHeight)+
           ' +X +V +LC:\POVRAY\INCLUDE');
           for K:=0 to Frames-1 do begin
               Angle:=(360/Frames)/DegRad*K;
               X:=round(Sin(Angle)*Depth);
               LookX:=I2S(X);
               Z:=round(Cos(Angle)*Depth);
               LookZ:=I2S(-Z);
               LightZ:=I2S(-Z+30);
               if K<10 then POV:=Root+'animator\anim0'+I2S(K)+'.pov'
               else POV:=Root+'animator\anim'+I2S(K)+'.pov';
               Trunk:=RemLet(POV,3);
               Writeln(Instruct,'povray +i'+pov+' +o'+Trunk+'tga %2 %3 %4 %5
%6 %7 > '+
               trunk+'log');
               Assign(Storage,POV);
               Rewrite(Storage);
```

```
                    Writeln(Storage,'// '+Title);
                    Writeln(Storage,'// '+TimeStr);
                    Writeln(Storage,'// Stopwatch : '+TimeTxt);
                    Writeln(Storage,'// Seed Size : '+I2S(Seed));
                    Writeln(Storage,'// Min. Particle Size : '+I2S(MinSize));
                    Writeln(Storage,'// Max. Particle Size : '+I2S(MaxSize));
                    Writeln(Storage,'// '+I2S(number)+' Particles');
                    Writeln(Storage,'// Define Camera Position');
                    Writeln(Storage,'camera {');
                    Writeln(Storage,' location <'+LookX+' 0 '+LookZ+'>');
                    Writeln(Storage,' look_at <0 0 0>');
                    Writeln(Storage,'}');
                    Writeln(Storage);
                    Writeln(Storage,'// Define light source');
                    Writeln(Storage,'object {');
                    Writeln(Storage,' light_source{ <'+I2S(X+25)+' 25 '+LightZ+'>
color red 1 green 0.75 blue 0.75 }');
                    Writeln(Storage,'}');
                    PovStore;                          {Write particle information}
                    end;             {For ..Frames..  }
               end;
       'M' : begin
               CellCen:=LWall+(RWall-LWall) div 2;
               CellRad:=(RWall-LWall);
               for J:=1 to number do if PosY[J] < highest then highest:=PosY[J];
               {Define X & Y positions for camera & look}
               depth:=round(CellRad*0.9);      {Allows camera to see all
agglomerate}
               camera(CellCen,Fl-Highest div 2);
               sun(CellCen+20,Fl-Highest div 2+20);
               Batch:='c:\tp\test\animate.bat';
               Assign(Instruct,Batch);
               Rewrite(Instruct);
               Writeln(Instruct,'@echo off');
               Writeln(Instruct,'cd data\animator');
               Writeln(Instruct,'SET POVRAYOPT=+W'+I2S(RayWidth)+'
+H'+I2S(RayHeight)+
               ' +X +V +LC:\POVRAY\INCLUDE');
               for K:=0 to Frames-1 do begin
                    Angle:=(360/Frames)/DegRad*K;
                    X:=round(CellCen+Sin(Angle)*Depth);
                    LookX:=I2S(X);
                    LookY:=I2S((Fl-Highest) div 2);
                    Z:=round(Cos(Angle)*Depth);
                    LookZ:=I2S(-Z);
                    Look:=LookX+' '+LookY;
                    LightZ:=I2S(-Z+30);
                    Light:=I2S(X+20)+' '+I2S((Fl-Highest) div 2+20)+' '+LightZ;
                    if K<10 then POV:=Root+'animator\anim0'+I2S(K)+'.pov'
                    else POV:=Root+'animator\anim'+I2S(K)+'.pov';
                    Trunk:=RemLet(POV,3);
                    Writeln(Instruct,'povray +i'+pov+' +o'+Trunk+'tga %2 %3 %4 %5
%6 %7 > '+
                    trunk+'log');
                    Assign(Storage,POV);
                    Rewrite(Storage);
                    Writeln(Storage);
                    Writeln(Storage,'declare Cylinder_Y = quadric {');
                    Writeln(Storage,'<1.0 0.0 1.0>');
                    Writeln(Storage,'<0.0 0.0 0.0>');
                    Writeln(Storage,'<0.0 0.0 0.0>');
```

```
                Writeln(Storage,'-1.0');
                Writeln(Storage,'}');
                Writeln(Storage,'#declare Disk_Y = intersection {        /*
Capped cylinder, Length in y axis */');
                Writeln(Storage,'quadric { Cylinder_Y  }');
                Writeln(Storage,'plane { <0.0 1.0 0.0> -1 inverse }');
                Writeln(Storage,'plane { <0.0 1.0 0.0>  1 }');
                Writeln(Storage,'}');
                Writeln(Storage,'#declare Silver = color red 0.196078 green 0.6
blue 0.8');
                Writeln(Storage,'// '+Title);
                Writeln(Storage,'// '+TimeStr);
                Writeln(Storage,'// Stopwatch : '+TimeTxt);
                Writeln(Storage,'// Min. Particle Size : '+I2S(MinSize));
                Writeln(Storage,'// Max. Particle Size : '+I2S(MaxSize));
                Writeln(Storage,'// '+I2S(number)+' Particles');
                Writeln(Storage);
                Writeln(Storage,'camera {');
                Writeln(Storage,' location <'+Look+' '+LookZ+'>');
                Writeln(Storage,' look_at <'+Look+' 0>');
                Writeln(Storage,'}');
                Writeln(Storage);
                Writeln(Storage,'// Define light source');
                Writeln(Storage,'object {');
                Writeln(Storage,' light_source{ <'+Light+'> color red 1 green
0.75 blue 0.75 }');
                Writeln(Storage,'}');
                PovStore;                       {Write particle information}
                end;          {For ..Frames..  }
            end;

            end;

            Writeln(Instruct,'\povray\dta\dtax anim*');
            Writeln(Instruct,'PLAY anim.fli /s14');
            Close(Instruct);
end;            {ENDs POV_Anim}
{*************************************************************************}
procedure aspect;
{Re-write particle co-ordinates for side and top view}

var
   Input : text;
   New, Header : string;
   X, Y, Z, Size, A, B, C : integer;

begin
     Assign(Storage,save);
     Reset(Storage);
     Readln(Storage,Header);
     Remember:=Save;
     New:='';
     for I:=1 to length(Save)-6 do New:=New+Save[I];
     New:=New+View+'.dat';
     Assign(Input,New);
     Rewrite(Input);
     Writeln(Input,Header);
     for I:=1 to 6 do begin
         Readln(Storage,Header);
         Writeln(Input,Header);
         end;
     Readln(Storage,Number);
     Writeln(Input,Number);
     For I:=1 to Number do begin
         Readln(Storage,X);
         Readln(Storage,Y);
         Readln(Storage,Z);
         Readln(Storage,Size);
         if view='S' then begin C:=X; B:=Y; A:=Z; end
         else begin A:=X; C:=Y; B:=Z;  end;
         Writeln(Input,A);
         Writeln(Input,B);
         Writeln(Input,C);
         Writeln(Input,Size);
         end;                      {End For I:=1 to number...}
         Close(Storage);
         Close(Input);
         Save:=New;
end;
{**************************************************************************}
procedure Retrieve;

begin
     if (dimensions=3) and (view<>'F') and (method='S') then aspect;
     Assign(Storage,save);
     Reset(Storage);
     Readln(Storage,Title);
     if Title[1]='M' then begin
        For I:=1 to 3 do Readln(Storage);
        CharRead(2,3,0); DownProb:=Round(Variable);
        CharRead(3,3,1); StickProb:=Round(Variable);
        SimInfo:='D:'+R2S(DownProb,3,0)+' S:'+R2S(StickProb,3,0);
        For I:=1 to 2 do Readln(Storage);
        end
     else for I:=1 to 6 do Readln(Storage);
     Read(Storage,Number);
     for I:=1 to Number do begin
         read(Storage,PosX[I]);
         read(Storage,PosY[I]);
         if Dimensions=3 then readln(Storage,PosZ[I]);
         read(Storage,size[I]);
         end;
     Close(Storage);
     {Restore original file name}
     if (dimensions=3) and (view<>'F') and (method='S') then Save:=Remember;
end;
{**************************************************************************}
procedure RetrieveGyr;

begin
     Assign(Storage,gyr);
     Reset(Storage);
     readln(Storage,Number);
     for I:=1 to Number do begin
         readln(Storage,PosX[I]);
         readln(Storage,PosY[I]);
         readln(Storage,PosZ[I]);
         readln(Storage,size[I]);
         end;
     Close(Storage);
end;
{**************************************************************************}
```

```
procedure Fibre;

var
   ReadX,ReadY,ReadZ : string;
   X,Y,Z : real;
   Fig : char;
   I,J,code : integer;
   dp, df, scaler : single;
   dendinfo : text;

begin
     J:=0; FMaxZ:=0; FMinZ:=500;
     Assign(Storage,save);
     Reset(Storage);
     Readln(Storage);Readln(Storage);
     CharRead(5,4,1); dp:=Variable;
     scaler:=3/dp;
     CharRead(5,4,1); df:=Variable;

     while not eof(Storage) do begin
           J:=J+1;
           ReadX:=''; ReadY:=''; ReadZ:='';

           For I:=1 to 23 do begin
               read(Storage,Fig);
               ReadX:=ReadX+Fig;
               end;
           read(Storage,Fig);

           For I:=1 to 23 do begin
               read(Storage,Fig);
               ReadY:=ReadY+Fig;
               end;
           read(Storage,Fig);

           For I:=1 to 23 do begin
               read(Storage,Fig);
               ReadZ:=ReadZ+Fig;
               end;

           Val(ReadX,X,code);
           Val(ReadY,Y,code);
           Val(ReadZ,Z,code);

           readln(Storage);

           PosX[J]:=240+round(X*df*scaler);
           PosY[J]:=240+round(Y*df*scaler);
           PosZ[J]:=240+round(Z*df*scaler);
           Size[J]:=round(dp*scaler);

           If PosZ[J]>FMaxZ then FMaxZ:=PosZ[J];
           If PosZ[J]<FMinZ then FMinZ:=PosZ[J];

           end;

     Number:=J;
     Close(Storage);

     Assign(dendinfo,'c:\tp\test\data\dend.inf');
     Rewrite(dendinfo);
     Writeln(dendinfo,'df : '+R2S(df,7,2));
     Writeln(dendinfo,'Z Max : '+R2S(FMaxZ,7,2));
     Writeln(dendinfo,'Z Min : '+R2S(FMinZ,7,2));
     Close(DendInfo);
end;
{***************************************************************************}
procedure Fryer;

var
   ReadX,ReadY,ReadSize : string;
   X,Y,S : integer;
   Fig : char;
   I,J,code : integer;

begin

J:=0;

Assign(Storage,save);
Reset(Storage);
Readln(Storage);Readln(Storage);

while not eof(Storage) do begin
J:=J+1;
ReadX:=''; ReadY:=''; ReadSize:='';

CharRead(15,3,0);
X:=Round(Variable);
CharRead(11,3,0);

Y:=Round(Variable);
CharRead(11,2,1);
S:=Round(Variable);

PosX[J]:=X;
PosY[J]:=480-Y;          {Changes co-ordinate system to 0,0 at BOTTOM Left}
{PosZ[J]:=round(Z);}
Size[J]:=S;

end;

number:=J;
close(Storage);
end;
{***************************************************************************}
procedure SEM;

var
   code : integer;

begin

Assign(Storage,save);
Reset(Storage);
Readln(Storage);Readln(Storage);

CharRead(2,3,0);
Himage:=Round(Variable);

CharRead(3,3,1);
Wimage:=Round(Variable);
```

```
Readln(Storage);

CharRead(2,3,0);
TB:=Round(Variable);

CharRead(3,3,0);
BB:=Round(Variable);

CharRead(3,3,0);
LB:=Round(Variable);

CharRead(3,3,1);
RB:=Round(Variable);

Close(Storage);

ExtraX:=320-((Wimage-RB) div 2)-LB;
ExtraY:=(480-Himage-TB+BB) div 2;
end;
{*************************************************************************}
procedure PoreSave;
var
   X,Y : integer;

begin
     {Account for file headers and return char.}
     HImage:=HImage+6; WImage:=WImage+1;
     SetColor(LightGreen);
     OutTextXY(554,10,'Saving...');
     Append(Keep);
     Writeln(Keep,Name);
     Writeln(Keep,I2S(FurthX+ExtraX));
     Writeln(Keep,I2S(FurthY+ExtraY));
     Close(Keep);
     Assign(DataFile,PoreFile);
     Rewrite(DataFile);
     Writeln(DataFile,'Pore Space Data');
     Writeln(DataFile,'Size:');
     Writeln(DataFile,'H '+R2S(HImage,3,0)+' W '+R2S(WImage,3,0));
     Writeln(DataFile,'Borders:');
     Writeln(DataFile,'T 005 B 000 L 000 R 000');
     for Y:=TB to BB do begin
     for X:=LB to RB do begin
         if GetPixel(X,Y)=Blue then begin
            Write(DataFile,'1');
            end
         else Write(DataFile,'0');
         end;
         Writeln(DataFile);
         end;
     TB:=5; BB:=0; LB:=0; RB:=0;
     Close(DataFile);
     ExtraX:=320-((Wimage-RB) div 2)-LB;
     ExtraY:=(480-Himage-TB+BB) div 2;
     DelText(554,10,'Saving...');
end;
{*************************************************************************}
procedure DosShell;
begin
     SwapVectors;
     Window(1,1,80,25);
```

```
      ClrScr;
      TextColor(LightGray);
      Writeln;
      Writeln('           DOS Shell : Type ''EXIT'' to return to program');
      TextColor(Blink+LightGray);
      Writeln('                      ** Use only DOS commands **');
      Writeln;
      Exec(GetEnv('COMSPEC'),'');
      SwapVectors;
end;
{***************************************************************************}
Procedure Ren;
var
   FileExists : Boolean;
   DirStr,Model, Name, Stem : string;
   DirFile : file;
   Q : integer;

begin
      Numfiles:=0;
      Assign(Storage,Root+'auto_01.dat');
      {$I-}
      Reset(Storage);
      Close(Storage);
      FileExists:=(IOResult = 0);
      {$I+}
      If FileExists then begin
         Model:='';
         Assign(Storage,Root+'auto_01.dat');
         Reset(Storage);
         For K:=1 to 2 do begin
             Read(Storage,Option);
             Model:=Model+Option;
             end;
         Readln(Storage);
         Read(Storage,Dimensions);
         if Model='Ba' then Stem:='Ball' else
         if Model='Di' then Stem:='Diff' else
         if Model='Mo' then begin
            Stem:='';
            for I:=1 to 3 do Readln(Storage);
            CharRead(2,3,0); DownProb:=Round(Variable);
            CharRead(3,3,1); StickProb:=Round(Variable);
            CharRead(21,2,1); MinSize:=Round(Variable);
            CharRead(22,2,1); MaxSize:=Round(Variable);
            Close(Storage);
            Stem:=Stem+I2S(MinSize)+'_'+I2S(MaxSize);
            DirStr:=Root+'Monte\'+Stem;
            {Check for existence of required directory}
            Assign(DirFile,DirStr+'\nul');
            {$I-}
            Reset(DirFile);
            Close(DirFile);
            {$I+}
            {Create directory if necessary}
            If IOResult<>0 then MkDir(DirStr);
            Stem:=Stem+'\';
            for Q:=1 to (3-length(I2S(downprob))) do Stem:=Stem+'0';
            Stem:=Stem+I2S(DownProb);
            for Q:=1 to (3-length(I2S(stickprob))) do Stem:=Stem+'0';
            Stem:=Stem+I2S(StickProb);
            end
         else if Model='5%' then Stem:='Mix05'
              else Stem:='Mix'+Model;

         if Model<>'Mo' then begin
            Stem:=Stem+'_';
            Name:=I2S(Dimensions)+'d\'+stem+'01'
            end
         else Name:='Monte\'+stem+'01';
         Save:=Root+Name+'.dat';
         Assign(Storage,Save);
         {$I-}
         Reset(Storage); Close(Storage);
         FileExists := (IOResult = 0);
         {$I+}
         if FileExists then begin
            K:=1;
            repeat
            Inc(K);
            if K<10 then Name:=stem+'0'+I2S(K)
            else Name:=stem+I2S(K);
            if Model ='Mo' then Save:=Root+'Monte\'+Name+'.dat'
            else Save:=Root+I2S(Dimensions)+'d\'+Name+'.dat';
            Assign(Storage,Save);
            {$I-}
            Reset(Storage);
            Close(Storage);
            FileExists := (IOResult = 0);
            {$I+}
            until not FileExists;
            NumFiles:=K-1;
            end;                    {End Normal FileExists}
J:=0;
repeat
      Inc(J);
      if J<10 then begin
      Assign(Storage,Root+'AUTO_0'+I2S(J)+'.DAT');
      Assign(GyrFile,Root+'AUTO_0'+I2S(J)+'.gyr');
      end
      else begin
      Assign(Storage,Root+'AUTO_'+I2S(J)+'.DAT');
      Assign(GyrFile,Root+'AUTO_'+I2S(J)+'.gyr');
      end;
      {$I-}
      Reset(Storage);
      Close(Storage);
      {$I+}
      If (IOResult = 0) then begin
         if Model='Mo' then Name:=Root+'MONTE\'+Stem
         else begin
              Name:=Root+I2S(Dimensions)+'d\'+Stem;
              Gyr:=Root+I2S(Dimensions)+'d\'+Stem;
              end;
         if J+Numfiles < 10 then begin
            Name:=Name+'0'; Gyr:=Gyr+'0'; end;

         Name:=Name+I2S(J+Numfiles)+'.DAT';
         Gyr:=Gyr+I2S(J+Numfiles)+'.gyr';

         Rename(Storage,Name);
         if (Dimensions=3) and (Model<>'Mo') then Rename(GyrFile,Gyr);
```

```
     end
     else exit;
     until (IOResult<>0);
     end;                    {End Auto FileExists}
end;
{**************************************************************************}
Procedure CheckGraph;
begin
     Assign(Network,'z:\site\tp6\bgi\nul');    {Checks for active network}
     Assign(Local,'c:\tp\bgi\nul');            {Checks for TP on hard drive}
     Assign(Other,'d:\tp\bgi\nul');
     {$I-}
     Reset(Local);
     Close(Local);
     LocalThere := (IOResult = 0);
     Reset(Other);
     Close(Other);
     OtherThere := (IOResult = 0);
     Reset(Network);
     Close(Network);
     NetThere := (IOResult = 0);
     {$I+}

     GraphDir:='L'; Path:='c:\tp\bgi';

     NoDrivers:=(not NetThere) and (not LocalThere) and (not OtherThere);
     If Nodrivers then Crash('No Graphics Drivers Present');
     If OtherThere and (not LocalThere) and (not NetThere) then begin
        {Use 'Other' directory}
        Path:='d:\tp\bgi';
        GraphDir:='L';
        end;
     If NetThere and (not LocalThere) and (not OtherThere) then begin
        {Run from Network}
        Path:='z:\site\tp6\bgi';
        GraphDir:='N';
        end;
end;
{**************************************************************************}
Function CurrDir : String;
var
   s : string;

begin
GetDir(0,s);
CurrDir:=s;
end;
{**************************************************************************}
Procedure CheckSubs;
var
   Srec : SearchRec;
   Error : integer;

begin
{Check for Sub-directories}
FindFirst(Root+'animator',AnyFile,Srec);
if DosError=0 then
FindFirst(Root+'2d',AnyFile,Srec);
if DosError=0 then
FindFirst(Root+'3d',AnyFile,Srec);
if DosError=0 then
```

```
FindFirst(Root+'3d\povray',AnyFile,Srec);
if DosError <> 0 then CreateSub;
end;
{***************************************************************************}
Procedure CreateSub;

begin
Window(6,16,70,21);
ClrScr;
Sound(100);Delay(200);NoSound;
TextColor(Green+Blink); GotoXY(16,1);
Writeln('Required sub-directories do not exist!');
TextColor(Green); GotoXY(22,3);
Write('Create Directories Y/[N] ? ');
Option:=UpCase(ReadKey);
ClrScr;
case Option of
     'Y' : begin
           {Create Sub-directories}
           MkDir(Root+'animator');
           MkDir(Root+'2d');
           MkDir(Root+'3d');
           MkDir(Root+'3d\povray');
           Created:=True;
           end;
     'N' : begin
           TextColor(Green); GotoXY(1,5);
           Write('Please enter a valid directory');
           delay(2500);
           Created:=False;
           end;
     end            {end CASE Y/N}
end;
{***************************************************************************}
procedure CloseData(Ave : single; Q : integer);

begin
Ave:=Ave/Q;
Writeln(DataFile);
Writeln(DataFile,'QuickAverage '+R2S(Ave,1,3));
Close(DataFile);
end;
{***************************************************************************}
end.
```

ENCLOSING.PAS - Enclosing circle routines

```
unit enclosing;

interface
{$N+,E+}
uses Crt,Graph, funcs, mover, screeninfo,variables;

procedure Enclose;
procedure PoreEnc;
procedure SEMEnc;

implementation
var
   EncMin, EncMax, Increment : single;
{****************************************************************************}
procedure Enclose;

var
   I,J : integer;
   Distance : single;
   Mass, RadEnc : Array [1..50] of Single;

begin
      ShowName;

      If Ripple>40 then Ripple:=40;
      CofG(number);
      EncMin:=radius; EncMax:=0;
      for I:=1 to number do begin
          if Dimensions=2 then Distance:=sqrt(sqr(PosX[I]-CofGX)+sqr(PosY[I]-
CofGY))+Size[I]
          else Distance:=sqrt(sqr(PosX[I]-CofGX)+sqr(PosY[I]-
CofGY)+sqr(PosZ[I]-CofGZ))+Size[I];
          if Distance < EncMin then EncMin:=Distance;
          if Distance > EncMax then EncMax:=Distance;
      end;

      Increment:=(EncMax-EncMin)/Ripple;
      SetColor(LightBlue);

      for I:=1 to Ripple do begin
          Mass[I]:=0;
          RadEnc[I]:=EncMin+I*(Increment)+1;
          Circle(round(CofGX),round(CofGY),round(RadEnc[I]));
          if dimensions=2 then for J:=1 to number do begin
                 Distance:=sqrt(sqr(PosX[J]-CofGX)+sqr(PosY[J]-CofGY))+Size[J];
                 if distance < RadEnc[I] then begin
                    Mass[I]:=Mass[I]+Pi*sqr(Size[J]);
                    if debug='On' then Particle(PosX[J],PosY[J],Size[J],Red);
                 end;
          end                                {End 2 dimensions}
          else for J:=1 to number do begin
                 Distance:=sqrt(sqr(PosX[J]-CofGX)+sqr(PosY[J]-
CofGY)+sqr(PosZ[J]-CofGZ))+Size[J];
                 if distance < RadEnc[I] then begin
                    Mass[I]:=Mass[I]+Pi*(4/3)*PowerFn(Size[J],3);
                    if Debug='On' then Particle(PosX[J],PosY[J],Size[J],Red);
                 end;
             end;                            {End 3 dimensions}
      delay(Wait);
```

```
     end;                                {End 'Ripple' repeat}
     for I:=1 to Ripple do begin
     RadEnc[I]:=ln(RadEnc[I]); Mass[I]:=ln(Mass[I]);
     end;

     if (not multi) or (Debug='On') then Continue;

     GraphIt(round(0.8*Ripple),1,'Enclosing Circle','ln Radius','ln
Mass',RadEnc,Mass);

     SetTextJustify(CenterText,TopText);
     SetColor(LightGreen);
     OutTextXY(554,35,Title);
     OutTextXY(554,155,chr($AF)+' D = '+R2S(linb,1,3)+' '+chr($AE));
     number:=number-1;            {Resets number for building}
     if (not multi) or (Debug='On') then Continue;
end;
{***********************************************************************}
procedure PixEnc;
var
   VGo,VStop,HGo,HStop :integer;
   Distance : single;
   Mass, RadEnc : Array [1..50] of Single;
   Fig : Char;

begin
     if Ripple>32 then Ripple:=32;
     Increment:=(EncMax-EncMin)/Ripple;
     SetColor(LightBlue);

     for I:=1 to Ripple do begin
         Mass[I]:=0;
         RadEnc[I]:=EncMin+I*(Increment);
         Circle(round(CofGX)+ExtraX,round(CofGY)+ExtraY,round(RadEnc[I]));
         Assign(Storage,save);
         Reset(Storage);

        for J:=1 to TB-1 do Readln(Storage);      {Skips outside RadEnc}
         for J:=TB to (HImage-BB) do begin
             for K:=1 to LB do Read(Storage,Fig);    {Skips outside RadEnc}
             for K:=LB+1 to (WImage-RB) do begin
                 if (K>LB) and (GetPix=1) then begin
                    Distance:=sqrt(sqr(K-CofGX)+sqr(J-CofGY));
                    if (distance < RadEnc[I]) then begin
                    Mass[I]:=Mass[I]+1;
                    if debug='On' then PutPixel(K+ExtraX,J+ExtraY,Red);
                    end;
                 end;                    {End found pixel}
         end;                            {End left to right}
     Readln(Storage);
     end;                                {End top to bottom}
     Close(Storage);
     delay(Wait);
     end;                                {End 'Ripple' repeat}

     for I:=1 to Ripple do begin
     RadEnc[I]:=ln(RadEnc[I]); Mass[I]:=ln(Mass[I]);
     end;

     if (not multi) or (Debug='On') then Continue;
     GraphIt(Ripple,1,'Enclosing Circle','ln Radius','ln Mass',RadEnc,Mass);

     SetTextJustify(CenterText,TopText);
     SetColor(LightGreen);
     OutTextXY(554,35,Title);
     OutTextXY(554,155,chr($AF)+' D = '+R2S(linb,1,3)+' '+chr($AE));
     number:=number-1;            {Resets number for building}
     if (not multi) or (Debug='On') then Continue;
end;
{**************************************************************************}
procedure PoreEnc;

var
   XDist, YDist : single;

begin
     XDist:=CofGX-FurthX; XDist:=sqr(XDist);
     YDist:=CofGY-FurthY; YDist:=sqr(YDist);
     EncMax:=round(Sqrt(XDist+YDist))+1;
     EncMin:=5;
     PixEnc;
end;
{**************************************************************************}
procedure SEMEnc;

begin
     EncMin:=5; EncMax:=MaxRad;
     PixEnc;
end;
{**************************************************************************}
end.
```

FUNC.PAS - Unit containig all mathematical function (rather than procedures)

```
unit funcs;

interface
{$N+,E+}
uses Crt,Graph, variables;

function X(Angle,Radius:single):integer;
function Y(Angle,Radius:single):integer;
function ArcCos(o,h:integer):single;
function ArcSin(a,h:integer):single;
function AngFind(x,z:single):single;
function Width(n : integer):integer;
function Diameter(X,Y,Z : array of integer; n : integer):integer;
function I2S(Val:integer):string;
function R2S(Val:real; Digit, Decimal:integer):string;
function NumCheck(Num : string):integer;
function RemLet(Name : string; N : integer):string;
function shorten(Name : string):string;
function RofG(number : integer):single;
function powerfn(number, exponent : real) : real;
function t(row,column : integer):single;
function GetPix:integer;
procedure PDist2D(PosX,PosY:integer);
procedure PDist(PosX,PosY,PosZ:integer);
procedure IntSort(Left, Right : integer;
                      var A, B, C, Size : Array of integer);
procedure RealSort(Left, Right : integer;
                      var A, B, C : Array of single;
                      size : Array of integer);
procedure GraphIt(n, fit : integer; Title, Xtitle, Ytitle : string;
                  varX, varY : Array of Single);
procedure ShowName;
procedure Cross(X,Y : integer);
procedure CofG(number : integer);
procedure SEMCofG;
function Cylpore(r1,d,x2,z2,r2 : single; j : integer) : single;

implementation

var
   BigX, BigY, SmallX, SmallY : single;
   Beta1low, Beta1high, Beta0low, Beta0high : single;
   RHY,LHY : integer;
   Left,Right,Top,Bottom : integer;
   XRatio, YRatio : single;

{*************************************************************************}
function X(Angle,Radius:single):integer;
         begin X:= Round((sin(Angle)*Radius)+CentreX); end;
{*************************************************************************}
function Y(Angle,Radius:single):integer;
         begin Y := Round((cos(Angle)*Radius)+CentreY); end;
{*************************************************************************}
function ArcCos(o,h:integer):single;
var
   x : single;

begin
     x:=o/h;
     ArcCos:=ArcTan (sqrt (1-sqr (x)) /x)
     end;
{***************************************************************************}
function ArcSin(a,h:integer):single;
var
   x : single;

begin
     x:=a/h;
     ArcSin:=ArcTan (x/sqrt (1-sqr (x)))
     end;
{***************************************************************************}
function AngFind(x,z:single):single;

{Corrects angles derived from tangents (z<0)
and also adds / subtracts 360ø where necessary}

begin

if z=0 then begin
   if x>=0 then AngFind:=Pi/2 else AngFind:=3*Pi/2;
   Exit;
   end

else a:=arctan(x/z);

if z<0 then a:=a+Pi;
if a<0 then a:=a+Pi*2;
if a>=2*Pi then a:=a-Pi*2;

AngFind:=a;

end;

{***************************************************************************}
function Width(n : integer):integer;

var
   I,temp : integer;
   X,Y,Z,Dist, Size : single;

begin
     temp:=0;
     for I:=1 to n do begin
         X:=PosX[I]-CentreX;
         Y:=PosY[I]-CentreY;
         if Dimensions=3 then Z:=PosZ[I]-CentreZ;
         Dist:=sqr(X)+sqr(Y)+sqr(Z);
         size:=sqrt(Dist);
         if size>temp then temp:=round(size);
         end;
     temp:=temp+maxsize*8;                 {Allows 8 maxsizes at edge}
     width:=temp;
end;
{***************************************************************************}
function diameter(X,Y,Z : array of integer;n : integer):integer;

var
   I,temp : integer;
   Xmax,Ymax,Zmax,Xmin,Ymin,Zmin : integer;
   Xsize,Ysize,Zsize : integer;
```

```
begin
     diameter:=0;
     Xmax:=0; Ymax:=0; Zmax:=0;
     Xmin:=2*CentreX; Ymin:=2*CentreY; Zmin:=2*CentreZ;
     for I:=0 to n-1 do begin
         if X[I] > Xmax then Xmax:=X[I];
         if Y[I] > Ymax then Ymax:=Y[I];
         if Z[I] > Zmax then Zmax:=Z[I];
         if X[I] < Xmin then Xmin:=X[I];
         if Y[I] < Ymin then Ymin:=Y[I];
         if Z[I] < Zmin then Zmin:=Z[I];
         end;
     Xsize:=Xmax-Xmin;
     Ysize:=Ymax-Ymin;
     Zsize:=Zmax-Zmin;

     if Xsize >= Ysize then                         {}
        if Xsize >= Zsize then temp:=Xsize          {}        {Check X/Z}
           else temp:=Ysize                         {}
        else if Ysize >= Zsize then temp:=Ysize     {}
             else temp:=Zsize;                      {}        {Check Y/Z}

     diameter:=temp;
end;
{**************************************************************************}
function I2S(Val:integer):string;

var
   Buffer:String[10];

begin
     str(Val, Buffer);
     I2S:=Buffer;
end;
{**************************************************************************}
function R2S(Val:real; Digit, Decimal:integer):string;

var
   Buffer:String[35];

begin
     str(Val:Digit:Decimal, Buffer);
     R2S:=Buffer;
end;
{**************************************************************************}
function NumCheck(Num : string):integer;
begin
end;
{**************************************************************************}
function RemLet(Name : string; N : integer):string;

var
   J : Integer;

begin
    J:=length(Name)-N+1;
    Delete(Name,J,N);
    RemLet:=Name;
end;
{**************************************************************************}
```

```
function shorten(Name : string):string;
var M : integer;
    T1,T2 : string;

begin

T1:=''; T2:='';

if length(Name)>34 then begin
   for M:=length(Name)-22 to length(Name) do T1:=T1+Name[M];
   T1:='...'+T1;
   for M:=1 to 34-length(T1) do T2:=T2+Name[M];
   Shorten:=T2+T1;
   end

else Shorten:=Name;

end;
{************************************************************************}
function RofG(number : integer):single;
var
   I : integer;
   Distance,Area,SumArea,MomInert,SumMomInert : single;

begin
     SumArea:=0; SumMomInert:=0;
     for I:=1 to number do begin
        if dimensions=2 then
        Distance:=sqrt(sqr(PosX[I]-CofGX)+sqr(PosY[I]-CofGY))
        else
        Distance:=sqrt(sqr(PosX[I]-CofGX)+sqr(PosY[I]-CofGY)+sqr(PosZ[I]-
CofGZ));
        Area:=Pi*sqr(Size[I]);
        SumArea:=SumArea+Area;
        MomInert:=Pi*PowerFn(Size[I],4)/4;
        SumMomInert:=SumMomInert+MomInert+sqr(Distance)*Area;
        end;
     RofG:=sqrt(SumMomInert/SumArea);
end;
{************************************************************************}
function Powerfn (number, exponent : real) : real;

begin

     if (exponent = 0.0) then
       powerfn := 1.0
     else if number = 0.0 then
       powerfn := 0.0
     else if abs(exponent*Ln(abs(number))) > 87.498 then
       begin writeln ('Overflow in POWERFN expression'); halt; end
     else if number > 0.0 then
       powerfn := Exp(exponent*Ln(number))
     else if (number < 0.0) and (Frac(exponent) = 0.0) then
       if Odd(Round(exponent)) then
         powerfn := -powerfn (-number, exponent)
       else
         powerfn :=  powerfn (-number, exponent)
     else
       begin writeln ('Invalid POWERFN expression'); halt; end;

end;
```

```
{**************************************************************************}
function t(row,column : integer):single;
var
   Digit : char;
   Joined : string;
   Data : text;
   Number : single;
   I,J,Code : integer;

begin
     Joined:='';
     Assign(Data,RData+'t.dat');
     Reset(Data);
     for I:=1 to row+4 do readln(Data);
     for J:=1 to (column*8) do read(Data,digit);
     repeat
           read(Data,digit);
           if digit<>' ' then Joined:=Joined+digit;
     until digit=' ';
     Close(Data);
     val(Joined,Number,Code);
     t:=Number;
end;
{**************************************************************************}
function GetPix:integer;

var
   Pix, Code : integer;
   Fig : Char;

begin
     if (SubType<>'S') or (save[length(save)]='r') then begin
        read(Storage,Fig);
        Val(Fig,Pix,code)
        end
     else begin
        read(Storage,Pix);
        Pix:=Pix+1;
        If Pix=255 then Pix:=0;
        end;

        GetPix:=Pix;
end;
{**************************************************************************}
function rfit(row,column : integer):single;

var
   Digit : char;
   Joined : string;
   Data : text;
   Number : single;
   I,J,Code : integer;

begin
     Joined:='';
     Assign(Data,RData+'rfit.dat');
     Reset(Data);
     for I:=1 to row+4 do readln(Data);
     for J:=1 to (column*8) do read(Data,digit);
     repeat
           read(Data,digit);
```

```
            if digit<>' ' then Joined:=Joined+digit;
        until digit=' ';
        Close(Data);
        val(Joined,Number,Code);
        rfit:=Number;
end;

{***************************************************************************}
procedure PDist2D(PosX,PosY:integer);
var
   X1,X2,X3,Y1,Y2,Y3 : single;

begin
     X1:=(Xin-Xout); X2:=Xin-PosX; X3:=Xout-PosX;
     Y1:=(Yin-Yout); Y2:=Yin-PosY; Y3:=Yout-PosY;
     a:=sqrt(sqr(X1)+sqr(Y1));
     b:=sqrt(sqr(X2)+Sqr(Y2));
     c:=sqrt(sqr(X3)+Sqr(Y3));
     d:=(a+b+c)/2;
     Perpdist:=(2/a)*sqrt(abs(d*(d-a)*(d-b)*(d-c)));
     end;
{***************************************************************************}
procedure PDist(PosX,PosY,PosZ:integer);
var
   X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3 : single;

begin
     X1:=(Xin-Xout); X2:=Xin-PosX; X3:=Xout-PosX;
     Y1:=(Yin-Yout); Y2:=Yin-PosY; Y3:=Yout-PosY;
     if Dimensions=2 then begin
     a:=sqrt(sqr(X1)+sqr(Y1));
     b:=sqrt(sqr(X2)+Sqr(Y2));
     c:=sqrt(sqr(X3)+Sqr(Y3));
     end
     else begin
     Z1:=(Zin-Zout); Z2:=Zin-PosZ; Z3:=Zout-PosZ;
     a:=sqrt(sqr(X1)+sqr(Y1)+sqr(Z1));
     b:=sqrt(sqr(X2)+sqr(Y2)+sqr(Z2));
     c:=sqrt(sqr(X3)+sqr(Y3)+sqr(Z3));
     end;
     d:=(a+b+c)/2;
     Perpdist:=(2/a)*sqrt(abs(d*(d-a)*(d-b)*(d-c)));
     end;
{***************************************************************************}
procedure IntSwitch(var A, B, X1, X2, Y1, Y2, S1, S2 : integer);

var
   C : Integer;
   X3, Y3, S3 : integer;

begin
     if A <> B then
        begin
             C := A;
             X3:=X1; Y3:=Y1; S3:=S1;
             A := B;
             X1:=X2; Y1:=Y2; S1:=S2;
             B := C;
             X2:=X3; Y2:=Y3; S2:=S3;
        end;
     end;
```

```
{***************************************************************************}
procedure RealSwitch(var A, B, X1, X2, Y1, Y2 : single;
                         S1, S2 : integer);

var
   C, X3, Y3 : Single;
   S3 : integer;

begin
     if A <> B then
        begin
             C := A;
             X3:=X1; Y3:=Y1; S3:=S1;
             A := B;
             X1:=X2; Y1:=Y2; S1:=S2;
             B := C;
             X2:=X3; Y2:=Y3; S2:=S3;
        end;
     end;
{***************************************************************************}
procedure IntSort(Left, Right : integer;
                      var A, B, C, Size : Array of integer);

var
   I, J, K : integer;
begin
     K := (A[Left] + A[Right]) div 2;
     I := Left;
     J := Right;

     repeat
           while A[I] < K do
                 Inc(I,1);

           while K < A[J] do
                 Dec(J,1);

           if I <= J then
              begin
                   IntSwitch(A[I],A[J],B[I],B[J],C[I],C[J],Size[I],Size[J]);
                   Inc(I,1);
                   Dec(J,1);
              end;
     until I > J;

     If left < J then
        IntSort(Left, J, A, B, C, Size);

     If I < Right then
        IntSort(I, Right, A, B, C, Size);

     end;
{***************************************************************************}
procedure RealSort(Left, Right : integer;
                      var A, B, C : Array of single;
                      size : Array of integer);
var
   I, J, K : integer;
begin
     K := round((A[Left] + A[Right]) / 2);
     I := Left;
     J := Right;

     repeat
           while A[I] < K do
                 Inc(I,1);

           while K < A[J] do
                 Dec(J,1);

           if I <= J then
              begin
                   RealSwitch(A[I],A[J],B[I],B[J],C[I],C[J],Size[I],Size[J]);
                   Inc(I,1);
                   Dec(J,1);
              end;
     until I > J;

     If left < J then
        RealSort(Left, J, A, B, C, Size);

     If I < Right then
        RealSort(I, Right, A, B, C, Size);

     end;
{**************************************************************************}
procedure Regression(n:integer; varX, varY : Array of Single;
                     W,H : single);

var
   SumX, SumY, SumXY, SumXSqu, SumYSqu : single;
   MeanX, MeanY, MeanXSqu, MeanYSqu, MeanXY : single;
   bStar1, bStar0, BetaStar1, BetaStar0 : single;
   VariY, Varib0, Varib1 : single;
   Freedom, Row,Column : integer;
   Check : integer;
   Error : single;
   Level : string;

begin
     SumX:=0; SumY:=0; SumXSqu:=0; SumYSqu:=0; SumXY:=0;

     for I:= 0 to n-1 do
         begin
              SumX:=SumX+varX[I];
              SumXSqu:=SumXSqu+sqr(varX[I]);
              SumY:=SumY+varY[I];
              SumYSqu:=SumYSqu+sqr(varY[I]);
              SumXY:=SumXY+(VarY[I]*varX[I]);
              if varX[I] > BigX then BigX:=varX[I];
              if varY[I] > BigY then BigY:=varY[I];
         end;

     linb:=(SumXY-(SumX*SumY)/n)/(SumXSqu-sqr(SumX)/n);
     lina:=(SumY-linb*SumX)/n;
     r:=(SumXY-(SumX*SumY)/n)/sqrt((SumXSqu-sqr(SumX)/n)*(SumYSqu-
sqr(SumY)/n));

     MeanX:=SumX/n; MeanY:=SumY/n; MeanXY:=SumXY/n;
     MeanXSqu:=SumXSqu/n; MeanYSqu:=SumYSqu/n;

     bStar1:=(MeanXY-MeanX*MeanY)/(MeanXSqu-sqr(MeanX));
```

```
    bStar0:=MeanY-bStar1*MeanX;

    VariY:=(n/(n-2))*((MeanYSqu-Sqr(MeanY))-sqr(bStar1)*(MeanXSqu-
Sqr(MeanX)));
    Varib0:=MeanXSqu*VariY/(n*(MeanXSqu-sqr(MeanX)));
    if Varib0<0 then Varib0:=abs(Varib0);
    Varib1:=VariY/(n*(MeanXSqu-sqr(MeanX)));
    if Varib1<0 then Varib1:=abs(Varib1);

    Freedom:=n-2;

    Case Convert of
         950 : Column:=1;
         980 : Column:=2;
         990 : Column:=3;
         995 : Column:=4;
         998 : Column:=5;
         999 : Column:=6;
    end;

    {Low and High values for Intercept}
    Beta0low:=bStar0-t(Freedom,Column)*sqrt(Varib0);
    Beta0high:=bStar0+t(Freedom,Column)*sqrt(Varib0);

    {Low and High values for Gradient}
    Beta1low:=bStar1-t(Freedom,Column)*sqrt(Varib1);
    Beta1high:=bStar1+t(Freedom,Column)*sqrt(Varib1);

    LHY:=Round(lina*YRatio);
    RHY:=Round((linb*W/XRatio+lina)*YRatio);

    SetColor(Red);
    SetLineStyle(DottedLn,0,NormWidth);
    line(Left,Bottom-LHY,Right,Bottom-RHY);
    SetColor(LightGray);
    LHY:=Round(Beta0low*YRatio);
    RHY:=Round((beta1low*W/XRatio+Beta0low)*YRatio);
    line(Left,Bottom-LHY,Right,Bottom-RHY);
    LHY:=Round(Beta0high*YRatio);
    RHY:=Round((beta1high*W/XRatio+Beta0high)*YRatio);
    line(Left,Bottom-LHY,Right,Bottom-RHY);
    SetColor(LightGreen);
    SetLineStyle(SolidLn,0,NormWidth);
    Rectangle(485,30,622,180);
    SetFillStyle(SolidFill,Black);
    FloodFill(490,35,LightGreen);
    SetTextJustify(LeftText,TopText);
    SetColor(Red);
    OutTextXY(490,75,'Gradient : '+R2S(linb,1,3));
    OutTextXY(490,95,'1-Rý : '+R2S((1-abs(r))*100,3,2)+'%');
    SetColor(LightGreen);
    Case n of
         3..22 : Row:=n-2;
         23..27 : Row:=21;
         28..32 : Row:=22;
         33..37 : Row:=23;
         end;
    SetTextJustify(CenterText,TopText);
    SetColor(LightGreen);
    Rectangle(515,272,585,378);
    SetFillStyle(SolidFill,Black);
```

```
     FloodFill(550,320,LightGreen);
     OutTextXY(550,272,'Conf. of fit');
     BestFit:='<90% ';
     for Column:=1 to 5 do begin
         Case column of
              1 : level:='90.0%';
              2 : level:='95.0%';
              3 : level:='98.0%';
              4 : level:='99.0%';
              5 : level:='99.9%';
              end;
         Error:=rfit(row,column);
         if (r>Error) or (r<-Error) then begin
         Setcolor(Green);
         BestFit:=level;
         end
         else SetColor(Red);
         OutTextXY(550,272+18*column,Level);
         end;
     SetTextJustify(leftText,TopText);

     SetColor(LightGray);
     OutTextXY(490,115,R2S(Convert/10,2,1)+'% limit :');
     OutTextXY(490,135,R2S(Betallow,1,3)+' < m < '+R2S(Betalhigh,1,3));
end;
{***************************************************************************}
procedure GraphIt(n,fit:integer; Title, Xtitle, Ytitle : string;
                  varX, varY : Array of Single);

var
   I,Row,Column : integer;
   ScrHeight, ScrWidth : integer;
   CentreX, CentreY : integer;
   SpotX, SpotY : integer;
   MaxX,MaxY : single;
   Points : string;

label
     Failed;

begin
     {Clear The Screen}
     ClearViewPort;
     SetColor(LightGreen);

     if n<4 then begin
        SetTextJustify(CenterText,TopText);
        OutTextXY(200,220,'Analysis Failed :');
        OutTextXY(200,240,'Insufficient data points');
        SetColor(Red);
        Rectangle(100,200,300,270);
        SetTextJustify(LeftText,TopText);
        SetColor(Green);
        ShowName;
        OuttextXY(450,460,'Press any key to continue');
        Delay(100);
        goto Failed;
        end;

     Rectangle(485,30,622,180);
```

```
      BigX:=0; BigY:=0;
      for I:= 0 to n-1 do begin
          if varX[I] > BigX then BigX:=varX[I];
          if varY[I] > BigY then BigY:=varY[I];
          end;

      CentreX:=GetMaxX div 2; CentreY:=GetMaxY div 2;
      Left:=round(GetMaxX*0.1); Right:=round(GetMaxX*0.9);
      Top:=round(GetMaxY*0.1);

      SetColor(Green);

      if (Method='S') and (Dimensions=3) and (FileType<>'M') then begin
         Bottom:=round(GetMaxY*0.80);
         SetTextJustify(leftText,TopText);
         Case View of
              'F' : OutTextXY(450,420,'Front view');
              'S' : OutTextXY(450,420,'Side view');
              'T' : OutTextXY(450,420,'Top view');
              end;
         end
      else if (Method='S') and (FileType='M') then begin
         Bottom:=round(GetMaxY*0.80);
         SetTextJustify(leftText,TopText);
         Case WalkDir of
              'R' : OutTextXY(450,420,'Left to Right');
              'L' : OutTextXY(450,420,'Right to Left');
              end;
         end
      else Bottom:=round(GetMaxY*0.85);

      ScrWidth:=Right-Left;
      ScrHeight:=Bottom-Top;

      line(Left,Top,Left,Bottom);
      line(Left,Bottom,Right,Bottom);

      XRatio:=ScrWidth/BigX/1.5;
      YRatio:=ScrHeight/BigY/1.6;

      If fit=1 then begin
      Regression(n,VarX,VarY,ScrWidth,ScrHeight);
      end;

      SetColor(White);
      For I:=0 to n-1 do begin
          SpotX:=round((varX[I])*XRatio)+Left;
          SpotY:=Bottom-round((varY[I])*YRatio);
          Cross(SpotX,SpotY);
      end;

      SetTextJustify(CenterText,CenterText);
      if Method='P' then begin
         OutTextXY(round(GetMaxX*0.4),Top div 2,Title);
         SetLineStyle(DottedLn, 0, NormWidth);
         SetColor(LightGreen);
         Case PoreMeth of 'A','W','P','I': Line(Left,Bottom-
round(AvPore*YRatio),Right,Bottom-round(AvPore*YRatio))
         else
Line(round(AvPore*XRatio)+Left,bottom,round(AvPore*XRatio)+Left,Top);
         end;
```

```
            SetLineStyle(SolidLn, 0, NormWidth);
            SetColor(White);
            end
        else OutTextXY(round(GetMaxX*0.4),Top div 2,'Fractal Dimension by
'+Title+' method');
        SetTextStyle(SmallFont,1,4);
        OutTextXY(5,CentreY,YTitle);
        SetTextStyle(SmallFont,0,4);
        OutTextXY(CentreX,Bottom+30,XTitle);
        SetColor(Green);
        MaxX:=(BigX*1.3); MaxY:=(BigY*1.5);
        if (Method='P') and (PoreMeth='W') then
        case Dimensions of
             2 : MaxX:=(RWall-LWall);
             3 : MaxX:=(RWall-LWall)/2;
             end;
        {Â X-axis ticks and labels Â}
        SetTextJustify(CenterText,TopText);
        if MaxX>1 then OutTextXY(round(MaxX*XRatio)+Left,Bottom+10,R2S(MaxX,4,1))
        else OutTextXY(round(MaxX*XRatio)+Left,Bottom+10,R2S(MaxX,4,2));
        Line(round(MaxX*XRatio)+Left,Bottom+5,round(MaxX*XRatio)+Left,Bottom);
        if MaxX>1 then
OutTextXY(round(MaxX*XRatio/2)+Left,Bottom+10,R2S(MaxX/2,4,1))
        else OutTextXY(round(MaxX*XRatio/2)+Left,Bottom+10,R2S(MaxX/2,4,2));

Line(round(MaxX*XRatio/2)+Left,Bottom+5,round(MaxX*XRatio/2)+Left,Bottom);
        {Â Y-axis ticks and labels Â}
        SetTextJustify(RightText,CenterText);
        if MaxY>1 then OutTextXY(Left-8,Bottom-
(round(MaxY*YRatio)),R2S(MaxY,4,1))
        else OutTextXY(Left-8,Bottom-(round(MaxY*YRatio)),R2S(MaxY,4,2));
        Line(Left-5,Bottom-(round(MaxY*YRatio)),Left,Bottom-
(round(MaxY*YRatio)));
        if MaxY>1 then OutTextXY(Left-8,Bottom-
(round(MaxY*YRatio/2)),R2S(MaxY/2,4,1))
        else OutTextXY(Left-8,Bottom-(round(MaxY*YRatio/2)),R2S(MaxY/2,4,2));
        Line(Left-5,Bottom-(round(MaxY*YRatio/2)),Left,Bottom-
(round(MaxY*YRatio/2)));
        SetTextJustify(LeftText,TopText);
        SetColor(White);
        OutTextXY(490,55,'Number of points : '+I2S(n));
        SetColor(LightGreen);
        if debug='On' then begin
           Points:=RemLet(Save,3)+'gra';
           Assign(GraphFile,Points);
           Rewrite(GraphFile);
           for K:=0 to n-1 do
               Writeln(GraphFile,R2S(VarX[K],6,3)+' '+R2S(VarY[K],6,3));
           Close(GraphFile);
           end;

Failed:

end;                                        {End 'GraphIt'}
{*****************************************************************************}
procedure ShowName;
var
   T1,T2,Shorten : string;

begin
     T1:='';T2:='';
     if length(Save)>38 then begin
        for M:=length(Save)-26 to length(Save) do T1:=T1+Save[M];
        T1:='...'+T1;
        for M:=1 to 38-length(T1) do T2:=T2+Save[M];
        Shorten:=T2+T1;
        end
     else Shorten:=Save;

     SetTextJustify(LeftText,TopText);
     SetColor(Green);
     if Length(Save)>30 then begin
        SetUserCharSize(5, 6, 5, 6);
        OuttextXY(450,440,Shorten);
        SetUserCharSize(1, 1, 1, 1);
        end
     else OuttextXY(450,440,Save);
end;
{***************************************************************************}
Procedure Cross(X,Y : integer);
var Orig : integer;

begin
     Orig:=GetColor;
     SetColor(white);
     line(X+2,Y+2,X-2,Y-2);      {X marks the spot!}
     line(X+2,Y-2,X-2,Y+2);
     SetColor(Orig);
end;
{***************************************************************************}
procedure CofG(number : integer);

var
   Area, SumArea, SumMomentX, SumMomentY, SumMomentZ : single;
   I : integer;
   SpotX,SpotY : integer;

begin
     SumArea:=0; SumMomentX:=0; SumMomentY:=0; SumMomentZ:=0;
     if Dimensions=2 then for I:=1 to Number do begin
        Area:=Pi*sqr(Size[I]);
        SumArea:=SumArea+Area;
        SumMomentX:=SumMomentX+(PosX[I]*Area);
        SumMomentY:=SumMomentY+(PosY[I]*Area);
        end
     else for I:=1 to Number do begin
          Area:=Pi*(4/3)*PowerFn(Size[I],3);
          SumArea:=SumArea+(Pi*(4/3)*PowerFn(Size[I],3));
          SumMomentX:=SumMomentX+(PosX[I]*Area);
          SumMomentY:=SumMomentY+(PosY[I]*Area);
          SumMomentZ:=SumMomentZ+(PosZ[I]*Area);
         end;
     CofGX:=SumMomentX/SumArea;
     CofGY:=SumMomentY/SumArea;
     if Dimensions=3 then CofGZ:=SumMomentZ/SumArea;

     SpotX:=round(CofGX);
     SpotY:=round(CofGY);
     Cross(SpotX,SpotY);
end;
{***************************************************************************}
procedure SEMCofG;
```

```
var
   SumArea, SumMomentX, SumMomentY, SumMomentZ,
   XDist, YDist, Dist, MaxDist : single;
   Pix,Code : integer;
   SpotX, SpotY, MaxX, MaxY : integer;
   Fig : char;
   level : integer;

begin
     SumMomentX:=0; SumMomentY:=0;
     Assign(Storage,save);
     Reset(Storage);
     PixNum:=0;

   for J:=1 to TB do Readln(Storage);          {Miss Top Border}
   for J:=TB to (HImage-BB) do begin           {Reads until BB}
       for K:=1 to LB do read(Storage,Fig);    {Miss Left Border}
       for K:=LB+1 to (WImage-RB) do begin     {Reads unitl RB}
             if GetPix=1 then begin
                SumMomentX:=SumMomentX+K;
                SumMomentY:=SumMomentY+J;
                PixNum:=PixNum+1;                {Add pixel to total}
                end;
             end;
       Readln(Storage);
       end;
   Close(Storage);

   if (PoreMeasure) and (not NewPore) then begin
   SetColor(Green);
   SetTextJustify(LeftText,TopText);
   PoreArea:=round(PixNum);
   OutTextXY(450,420,'Area of pore : '+R2S(PoreArea,5,0)+' pixels');
   ShowName;
   end;

   CofGX:=SumMomentX/PixNum;
   CofGY:=SumMomentY/PixNum;

   MaxDist:=0;
   Reset(Storage);
   for J:=1 to TB do Readln(Storage);          {Miss Top Border}
   for J:=TB to (HImage-BB) do begin           {Reads until BB}
       for K:=1 to LB do read(Storage,Fig);    {Miss Left Border}
       for K:=LB+1 to (WImage-RB) do begin     {Reads unitl RB}
             if GetPix=1 then begin
                XDist:=K-CofGX; XDist:=Sqr(XDist);
                YDist:=J-CofGY; YDist:=Sqr(YDist);
                Dist:=sqrt(XDist+YDist);
                if Dist>MaxDist then begin
                   FurthX:=K; FurthY:=J;
                   MaxDist:=Dist; end;
                end;
             end;
       Readln(Storage);
       end;
   Close(Storage);
   PutPixel(FurthX+ExtraX,FurthY+ExtraY,LightGreen);

   SpotX:=round(CofGX);
   SpotY:=round(CofGY);

   if (SpotX-LB)>((WImage-RB)-SpotX) then MaxX:=((WImage-RB)-SpotX)
      else MaxX:=(SpotX-LB);
   if ((HImage-BB)-SpotY)>(SpotY-TB) then MaxY:=(SpotY-TB)
      else MaxY:=((HImage-BB)-SpotY);
   if MaxX>=MaxY then MaxRad:=MaxY else MaxRad:=MaxX;

   if (Method='E') and (EncMeth='B') then begin
      SetColor(DarkGray); SetLineStyle(DottedLn,0,0);
      Rectangle(LB+ExtraX,TB+ExtraY,WImage-RB+ExtraX,Himage-BB+ExtraY);
      SetLineStyle(SolidLn,0,0);
      SetColor(Magenta);
      Circle(SpotX+ExtraX,SpotY+ExtraY,MaxRad);
      end;

   Cross(SpotX+ExtraX,SpotY+ExtraY);

end;
{************************************************************************}
function Cylpore(r1,d,x2,z2,r2 : single; j : integer) : single;

var
   y,size : integer;
   Root,PartArea,Partvol,Sect1,Sect2,A1,A2,Slice,
   ry,alpha,cenchord,base,height,x1,z1,dy : single;
   {**cenchord = distance from CellCen to chord intersecting particle**}
   con,a,b,c,rootx1,rootz1,rootx2,rootz2,dx,dz : double;

begin

x1:=CellCen; z1:=CellCen;

Size:=round(r2);
Slice:=0;

Partvol:=4/3*Pi*powerfn(r2,3);

for y:=-Size+1 to Size-1 do begin

    dx:=x1-x2; dz:=z1-z2;
    dy:=(r2-y);
    ry:=(sqr(r2)-sqr(y));
    PartArea:=Pi*ry;

    if (d-sqrt(ry)<r1) and (d+sqrt(ry)>r1) then begin
       con:=(sqr(x1)-sqr(x2))+(sqr(z1)-sqr(z2))+(ry-sqr(r1));
       a:=4*(sqr(dz)+sqr(dx));
       b:=4*(2*dz*dx*x1-2*sqr(dx)*z1-con*dz);
       c:=sqr(con)-4*con*x1*dx+4*sqr(dx)*(sqr(x1)+sqr(z1)-sqr(r1));

       Root:=sqr(b)-4*a*c; if abs(Root)<1e-3 then Root:=0;
       rootz1:=(-b+sqrt(Root))/(2*a);
       rootz2:=(-b-sqrt(Root))/(2*a);

       if dx<>0 then begin
          rootx1:=(con-2*rootz1*dz)/(2*dx);
          rootx2:=(con-2*rootz2*dz)/(2*dx);
          end
       else begin
          rootx1:=sqrt(sqr(r1)-sqr(rootz1-z1))+x1;
```

```
            rootx2:=x1+(rootx1-x1);
            end;

         dx:=rootx1-rootx2; dz:=rootz1-rootz2;
         base:=sqrt(sqr(dx)+sqr(dz))/2;          {Base of triangle = chord / 2}
         cenchord:=sqrt(sqr(r1)-sqr(base));
         height:=r1-cenchord;
         alpha:=arctan(base/cenchord);
         A1:=alpha*sqr(r1);
         A2:=base*cenchord;
         Sect1:=A1-A2;

         Root:=(ry-sqr(base));  if abs(Root)<5e-5 then Root:=0;
         height:=sqrt(root);
         if height<>0 then beta:=arctan(base/height) else beta:=Pi;
         {Distance from centre of particle to chord}
         A1:=beta*ry;
         A2:=base*height;
         if d>r1 then Sect2:=A1-A2
            else Sect2:=PartArea-A1+A2;
         Slice:=Slice+(Sect1+Sect2);
         end                                   {End circles OVERLAP}
      else Slice:=Slice+0;                     {else circles do not overlap}

      if d+sqrt(ry)<r1 then Slice:=PartArea;   {Particle INSIDE cylinder}

      end;                                     {end y loop}

  CylPore:=Slice;
  end;
  {**************************************************************************}
  end.
```

GYRATION.PAS - Radius of Gyration calculations. Note that the actual function for the calculation of the radius of gyration of a n object is contained within FUNCS.PAS

```
unit gyration;

{$N+,E+}

interface

uses Crt,Graph,funcs,mover,screeninfo,variables;

procedure  RadGyr;

implementation
{**************************************************************************}
procedure RadGyr;

var
   Xvar, Yvar : Array [1..50] of single;
   I,J : integer;
   Increments : single;

begin
     ShowName;
     If Ripple>32 then Ripple:=32;
     Increments:=number/ripple;
     for I:=1 to ripple do begin
         J:=round(I * increments);
         CofG(J);
         Xvar[I]:=ln(J);
         Yvar[I]:=ln(RofG(J));
         SetColor(LightBlue);
         Circle(round(CofGX),round(CofGY),round(RofG(J)));
         if debug='On' then for K:=1 to J do
Particle(PosX[K],PosY[K],Size[K],Red);
         delay(Wait);
     end;
     if (not multi) or (Debug='On') then Continue;
     GraphIt(ripple,1,'Radius of Gyration','ln # of Particles','ln Radius of
Gyr.',
     Xvar,Yvar);
     SetTextJustify(CenterText,TopText);
     SetColor(LightGreen);
     OutTextXY(554,35,Title);
     OutTextXY(554,155,chr($AF)+' D = '+R2S(1/linb,1,3)+' '+chr($AE));
     number:=number-1;           {Resets number for building}
     if (not multi) or (Debug='On') then Continue;
end;
{**************************************************************************}
end.
```

LOOK.PAS - Retrieves and displays saved agglomerate information for display or analysis

```
unit look;

{$N+,E+}

interface

uses Crt,Dos,Graph,
     boxcounting,disk,enclosing,funcs,gyration,menus,mover,porosity,
     revolve,screeninfo,swalk,variables;

procedure Display;
procedure Analyse;

implementation

var
   Average : single;
   Q : integer;

{****************************************************************************}
procedure Display;
var
   letter : char;
   FileID,J : integer;
   Ext : string;

label
     Scanned, NextFile;

begin
     Display_Menu;

     Model:='B';
     if Return then exit;

     If (Dimensions=3) and (Twist='On') then begin
        if multi then begin
           multi:=false;
           Save:=Root+Stem+'01.dat';
           end
        else Save:=Root+Name+'.dat';
        end;

     if multi then begin
        FileID:=1;
        Save:=Root+Stem+'01.dat';
        end;

     GraphDriver := Detect;        {Sets and checks graphmode}
     InitGraph(GraphDriver, GraphMode,Path);
     if GraphResult <> grOk then Halt(1);

     nextfile:

     Assign(Storage,Save); Reset(Storage);
     Readln(Storage,Title);
```

```
if Title='SEM Image (Scion)' then begin
   FileType:='S'; SubType:='S' end;
if Title='SEM Image (Fibre)' then begin
   FileType:='S'; Subtype:='I' end;
if Title='SEM Image' then begin
   FileType:='S'; Subtype:='V' end;

if FileType='S' then goto Scanned;

Read(Storage,Dimensions);
Readln(Storage);

if (Title<>'Fibre') and (Title<>'Basic') then begin
   FileType:='N';
   if Title[1]='M' then begin
      Model:='F'; FileType:='M';
      end;
   {Identify text & respond accordingly
   TimeTxt : Seed time
   TimeStr : 'Stopwatch' time}
   Read(Storage,letter);
   if Letter='T' then J:=12 else J:=6;
   For I:=1 to J do Read(Storage,letter);
   Readln(Storage,TimeTxt);
   Read(Storage,letter);
   if Letter='S' then J:=11 else J:=13;
   For I:=1 to J do Read(Storage,letter);
   Readln(Storage,TimeStr);
   end
else begin
if Title='Fibre' then FileType:='F';
if Title='Basic' then FileType:='B';
end;
Close(Storage);

Scanned:

DimBox;

Case FileType of
'N','M' : Retrieve;
'F' : Fibre;
'B' : Fryer;
'S' : SEM;
end;

Case FileType of
'N','F', 'B', 'M' : Review;
'S' : SEMView;
end;

if FileType<>'S' then

For I:=1 to Number do Particle(PosX[I],PosY[I],Size[I],Red);

if multi then begin
   if FileID < Numfiles then outtextXY(450,400,'Next file    : Pg Down');
   if FileID > 1 then outtextXY(450,420,'Previos file : Pg Up');
   end;

if (Dimensions=3) and (Twist='On') then
```

```
                  spin_menu(PosX, PosY, PosZ, Size,Number,CentreX,CentreY)

     else Option:=ReadKey;

     if (multi) and (option=#0) then begin
        option:=readkey;
        if ord(option)=81 then Inc(FileID);       {Pg Down - Next file}
        if ord(option)=73 then Dec(FileID);       {Pg Up   - Previous file}
        if FileID<1 then FileID:=1;
        if FIleID>NumFiles then FileID:= NumFiles;
        if FileID<10 then Ext:='0'+I2S(FileID)
        else Ext:=I2S(FileID);
        Ext:=Ext+'.dat';
        Save:=Root+Stem+Ext;
        ClearViewPort;
        Goto NextFile;
        end;

     CloseGraph;
end;
{***************************************************************************}
procedure NewSEM;

begin
     FailTxt:='';
     if Q<10 then Save:=Root+stem+'0'+I2S(Q)+'.dat'
     else Save:=Root+stem+I2S(Q)+'.dat';
     SEM; SEMDraw; ShowName;
end;
{***************************************************************************}
procedure PoreAnalyse;

var
   KeepExists : boolean;
   KeepName : string;
   XFlood,YFlood : integer;
   SaveCol : integer;

label
   WrongFlood;

begin

WrongFlood:

if InvPore then Save:=Root+ParentName+'.dat';

     if NewPore then begin
        multi:=false;
        PoreStore:=RemLet(Name,2);
        Assign(Keep,Root+PoreStore+'.sto');
        {$I-}
            Reset(Keep);
            Close(Keep);
            KeepExists := (IOResult = 0);
        {$I+}

        {*****Identify new pore in simulation*****}
        FurthX:=0; FurthY:=0;
        if FileType='M' then ReDraw
        else begin SEMDraw; SEMCofG; end;
```

```
DimBox;
SetTextJustify(CenterText,TopText);
OutTextXY(559,35,Title);
if FileType='M' then begin
   SetColor(blue);
   {Put SOLID line at top of cell (for floodfill)}
   Rectangle(LWall,Roof,RWall,Fl);
   Line(LWall,Roof,RWall,Roof);
   FurthX:=(RWall+LWall) div 2; FurthY:=(Roof+20);
   ExtraX:=0; ExtraY:=0;
   end
else begin
   SetColor(Blue);
   Rectangle(LB+ExtraX,TB+ExtraY,WImage-RB+ExtraX,Himage-BB+ExtraY);
   FurthX:=(LB+ExtraX+7);
   FurthY:=(TB+ExtraY+7);
   end;
Cross(FurthX,FurthY);
Save:=RemLet(PoreFile,3)+'por';
SetColor(Green);

{Check and react to previously saved pores}
if not(KeepExists) then begin
    ReWrite(Keep);
    Close(Keep);
    end
else begin
     Reset(Keep);
     repeat
           readln(Keep,KeepName);
           if KeepName=Name then begin
              Readln(Keep,XFlood);
              Readln(Keep,YFlood);
              SaveCol:=(GetColor);
              SetColor(Red);
              FloodFill(XFlood,YFlood,Blue);
              SetColor(Red);
              if FileType='S' then begin
                 OuttextXY(250,440,'Red areas indicate');
                 OuttextXY(250,460,'previously saved pores');
                 end
              else begin
                   OuttextXY(550,290,'Red areas indicate');
                   OuttextXY(550,310,'previously saved pores');
                   end;
              SetColor(SaveCol);
              end
           else begin Readln(Keep); ReadLn(Keep); end;
           until eof(Keep);
     Close(Keep);
     end;


Isolate;

if Aborted then begin
   Save:=Root+Name+'.dat';
   CloseGraph; exit; end;

InvPore:=False; Flood;
if InvPore=True then goto WrongFlood;
```

```
            PoreSave;
            Continue;
            NewPore:=False;
            SEM;
            SEMDraw;
            SEMCofG;
            SEMFerets;
            Save:=Root+Name+'.dat';
            SetTextJustify(CenterText,TopText);
            DelText(554,10,'Perimeter Analysis...');
            Continue;
            end                        {End NewPore}

        else begin
            if Multi then begin
              Average:=0;
              Assign(DataFile,Root+PoreStem+Stem+'P'+PropMeth+'.txt');
              Rewrite(DataFile);

              {Begin multiple pore analysis below}
              for Q:=1 to NumFiles do begin
                  FailTxt:='';
                  if Q<10 then Save:=Root+PoreStem+stem+'0'+I2S(Q)+'.por'
                  else Save:=Root+PoreStem+stem+I2S(Q)+'.por';
                  SEM;
                  SEMDraw;
                  SEMCofG;
                  SEMFerets;
                  case PropMeth of
                       'S' : begin
                             PoreWalk(FurthX,FurthY);
                             Write(DataFile,R2S(PoreArea,6,0)+' ');
                             Write(DataFile,R2S(1-linb,1,3)+' '+R2S((1-
abs(r))*100,5,2)+' ');
                             if FailTxt='' then Writeln(DataFile,BestFit)
                             else begin
                                  Write(DataFile,BestFit+' ');
                                  Writeln(DataFile,'  Fail @ '+FailTxt);
                                  end;
                             Average:=Average+1-linb;
                             end;
                       'E' : begin
                             PoreEnc;
                             Write(DataFile,R2S(PoreArea,6,0)+' ');
                             Write(DataFile,R2S(linb,1,3)+' '+R2S((1-
abs(r))*100,5,2)+' ');
                             Writeln(DataFile,BestFit);
                             Average:=Average+linb;
                             end;
                       end;
                  end;       {End for Q:=1 to NumFiles}
              end            {End Multi}
          else begin
                Save:=RemLet(PoreFile,3)+'por';
                SEM;                   {Reads file already on disk}
                SEMDraw;
                SEMCofG;
                SEMFerets;
                case PropMeth of
                     'S' : PoreWalk(FurthX,FurthY);
                     'E' : PoreEnc;
                  end;
              end;
          Method:='P';
          if Multi then begin
             Save:=Root+ParentName+'.dat';
             PoreMulti:=true;
             end
          else Save:=Root+Name+'.dat';
       end;                    {End not NewPore (anaylsis section)}

Name:=ParentName;

end;
{**************************************************************************}
procedure SEMAnalyse;

begin
     SEMDraw;

     if (PoreMeasure and (PoreMeth<>'O')) or (not PoreMeasure)
         then SEMCofG;

     Case Method of
          'S' : begin
                 if SubType='I' then FibreWalk
                 else begin
                      SEMFerets;
                      Cursor;
                      SEMWalk(FurthX, FurthY);
                      end
                 end;

          'E' : if EncMeth='B' then begin
                    if Multi then begin
                       Assign(DataFile,Root+Stem+Method+'.txt');
                       Rewrite(DataFile);
                       for Q:=1 to Numfiles do begin
                           NewSEM;
                           SEMCofG;
                           SEMEnc;
                           Average:=Average+(linb);
                           Writeln(DataFile,R2S(linb,1,3))
                           end;                              {End for Q...}
                       end                                  {End multi}
                    else SEMEnc;
                    end
                 else PoreEnc;

          'P' : begin
                   if Multi then begin
                      Assign(DataFile,Root+Stem+Method+'.txt');
                      Rewrite(DataFile);
                      for Q:=1 to Numfiles do begin
                         NewSEM;
                         SEMPore;
                         Average:=Average+(AvPore);
                         Writeln(DataFile,R2S(AvPore,1,4))
                         end;                              {End for Q...}
                      end                                  {End multi}
                   else SEMPore;
                   end;
```

```
        'B' : begin
                if Multi then begin
                   Assign(DataFile,Root+Stem+Method+'.txt');
                   Rewrite(DataFile);
                   for Q:=1 to Numfiles do begin
                      NewSEM;
                      SEMBox;
                      Average:=Average-linb;
                      Writeln(DataFile,R2S(-linb,1,3)+' '+R2S((1-
abs(r))*100,3,3))
                      end;                               {End for Q...}
                   CloseData(Average,Q);
                   end                                   {End multi}
                else SEMBox;
                end;
        end;          {End Case}

     end;             {End SEMAnalyse}
{***********************************************************************}
procedure AggAnalyse;

label
     SingleSkip, MultiSkip;

begin

     if not multi then begin
        if (Dimensions=3) and (FileType='M') and (Method='S') then goto
SingleSkip;
        if (FileType='M') and (Method='P') and (PoreMeth='W') and
(Dimensions=3) then PlanDraw
           else Redraw;
        SingleSkip:
        DimBox;
        end
     else begin
        Average:=0;
        {$I-}
        if (FileType='M') then begin
           if (Method='S') then
Assign(DataFile,Root+Stem+Method+WalkDir+'.txt')
           else if (Method='P') then
Assign(DataFile,Root+Stem+Method+PoreMeth+'.txt');
           end
        else begin
           {Include View in file name for 3-D Structured Walk}
           if (Dimensions=3) and (Method='S') then
Assign(DataFile,Root+Stem+Method+View+'.txt')
           else Assign(DataFile,Root+Stem+Method+'.txt')
           end;
        Rewrite(DataFile);
        if IOResult = 5 then begin CloseGraph; DiskError(5); exit; end
        {$I+}
        end;

     {Begin multiple agglomerate analysis below}
     if Multi then for Q:=1 to Numfiles do begin
     FailTxt:='';

        if Q<10 then begin
```

```
            Save:=Root+stem+'0'+I2S(Q)+'.dat';
            Gyr:=Root+stem+'0'+I2S(Q)+'.gyr';
            end
         else begin
            Save:=Root+stem+I2S(Q)+'.dat';
            Gyr:=Root+stem+I2S(Q)+'.gyr';
            end;

         Retrieve;
         if (Dimensions=3) and (FileType='M') and (Method='S') then goto
MultiSkip;
         if (Dimensions=3) and (FileType='M') and (Method='P') and
(PoreMeth='W') then PlanDraw
         else Redraw;
         MultiSkip:
         DimBox;

      Case Method of
      'S' : begin
            SetColor(Green);
            SetTextJustify(leftText,TopText);
            ShowName;
            if (dimensions=3) and (FileType<>'M') then Case View of
               'F' : OutTextXY(450,420,'Front view');
               'S' : OutTextXY(450,420,'Side view');
               'T' : OutTextXY(450,420,'Top view');
               end;
            if FileType='M' then begin
               FeretMax:=(RWall-LWall)/Pi;
               FiltSteps;
               end            {End FileType='M'}
            else begin
               AggFerets;
               AggWalk(PosX[Outer], PosY[Outer], Size[Outer]);
               if View <> 'F' then erase(Storage);
               end;
            if FailTxt='' then Writeln(DataFile,R2S(1-linb,1,3)+' '+R2S((1-
abs(r))*100,3,2))
            else begin
                Write(DataFile,R2S(1-linb,1,3)+' '+R2S((1-abs(r))*100,3,2));
                Writeln(DataFile,'  Fail @ '+FailTxt);
                end;
            Average:=Average+1-linb;
            end;   {End method 'S'}
      'E' : begin
            Enclose;
            Writeln(DataFile,R2S(linb,1,3)+' '+R2S((1-abs(r))*100,3,2));
            Average:=Average+linb;
            end;   {End Method 'E'}
      'P' : begin
            if (FileType<>'M') and (Dimensions=3) then RetrieveGyr; {Retrieve
'age' data}
            if FileType='M' then FiltPores else Pores;
            Writeln(DataFile,R2S(AvPore,1,3));
            Average:=Average+AvPore;
            end;   {End Method 'P'}
      'R' : begin           {Radius of gyration}
            if Dimensions=3 then RetrieveGyr;
            RadGyr;
            Writeln(DataFile,R2S(1/linb,1,3)+' '+R2S((1-abs(r))*100,3,2));
            Average:=Average+1/linb;
            end;    {End Method 'R'}
      end;
      end          {End Multi}

      else Case method of

      'S' : begin
            SetColor(Green);
            SetTextJustify(leftText,TopText);
            ShowName;
            if (dimensions=3) and (FileType<>'M') then Case View of
               'F' : OutTextXY(450,420,'Front view');
               'S' : OutTextXY(450,420,'Side view');
               'T' : OutTextXY(450,420,'Top view');
               end;
            if FileType='M' then begin
               FeretMax:=(RWall-LWall)/Pi;
               FiltSteps;
               end              {End FileType='M'}
            else begin
               AggFerets;
               AggWalk(PosX[Outer], PosY[Outer], Size[Outer]);
               if View <> 'F' then erase(Storage);
               end;             {End FileType<>'M'}
            end;        {End method 'S'}
      'E' : Enclose;
      'P' : begin
            if FileType<>'M' then begin
            if Dimensions=3 then RetrieveGyr; {Retrieve 'age' data}
            Pores;
            end
            else FiltPores
            end;    {End Method 'P'}
      'R' : begin
            if Dimensions=3 then RetrieveGyr;
            RadGyr;
            end;    {End Method 'R'}
      end;         {End NOT Multi}

end;
{*************************************************************************}
procedure Analyse;

var
   reps,sets : integer;
   ending : string;

label
    Skip, SingleSkip;

begin
      Ana_Menu;
      if return then exit;

      for sets:=0 to 0 do begin

{      ending:=I2S(0+5*sets);
      if sets<2 then ending:='00'+ending
      else if (sets>1) and (sets<20) then ending:='0' + ending;
      stem:='monte\8_8\050'+Ending;}
```

```
     Model:='B'; FileType:='N'; SubType:='N';          {Sets Defaults}
     if Multi then Save:=Root+stem+'01.dat';

     Assign(Storage,save);
     Reset(Storage);
     Readln(Storage,Title);

     case Title[1] of 'B','M','S','F' : FileType:=Title[1];
          end;

     if Title[1]='M' then Model:='F';
     if Title='SEM Image (Fibre)' then SubType:='I';
     if Title='SEM Image (Scion)' then SubType:='S';

     if FileType<>'S' then Read(Storage,Dimensions);
     Close(Storage);

     PoreMeasure:=((FileType='M') or (FileType='S')) and (Method='P') and
(PoreMeth='P');

     if PoreMeasure then Pore_Menu;
     if return then exit;

     InitGraph(GraphDriver, GraphMode,Path);

     for reps:=0 to 0 do begin

     case reps of
     1 : begin Method:='S'; WalkDir:='R'; end;
     2 : WalkDir:='L';
     3 : begin Method:='P'; PoreMeth:='V'; end;
     end;

     {Finds information for each simulation}

     Case FileType of
     'N','M' : retrieve;
     'F' : Fibre;
     'B' : Fryer;
     'S' : SEM;
     end;

     if PoreMeasure then begin
        PoreAnalyse;
        Goto Skip
        end;

     if FileType='S' then begin
        SEMAnalyse;
        Goto Skip
        end;

     AggAnalyse;

     Skip:

     if Multi then CloseData(Average,Q);

     end;                                    {End reps}

     CloseGraph;
     if PoreMeasure and Multi then Multi:=False;

     end;                                   {End sets}

end;
{****************************************************************************}
end.
```

MAKE.PAS – Controls creation of all simulation structures

```
unit make;

{$N+,E+}

interface

uses Crt,Dos,Graph,
     carlomove,disk,funcs,menus,revolve,rolling,screeninfo,variables,mover;

procedure Simulate;

implementation

var
   Elapse : single;
   hour, min, sec : integer;
   CurrSize : integer;

{************************************************************************}
procedure Agglomerate;

var
   OrigDir : single;
   MaxRad : single;
   DiffMoves, BalMoves : integer;

label
     redo;

begin
     if (((Auto='Auto') and (N=1)) or (Auto='Manual')) and (model='M')
        then Mixed_Menu;

     if Return=True then exit;

     if (Auto='Auto') and (Dimensions=3) then
        if N>10 then GYR:=Root+'auto_'+I2S(N)+'.GYR'
        else GYR:=Root+'auto_0'+I2S(N)+'.GYR';

     GraphDriver := Detect;                    {Sets and checks graphmode}
     InitGraph(GraphDriver, GraphMode,Path);
     if GraphResult <> grOk then Halt(1);

     PosX[1]:=CentreX; PosY[1]:=CentreY; PosZ[1]:=CentreZ; Size[1]:=seed;
     particle(PosX[1],PosY[1],Size[1],Blue);          {Seed particle details}

     if dimensions=3 then begin                       {write Rad of Gyr file}
        Assign(Storage,GYR);
        Rewrite(Storage);
        Writeln(Storage,number);
        Writeln(Storage,PosX[1]);
        Writeln(Storage,PosY[1]);
        Writeln(Storage,PosZ[1]);
        Writeln(Storage,Size[1]);
        end;

     BuildScr;                                        {SCREEN INFOrmation}

     if GraphDir='L' then begin Sound(600); Delay(75); NoSound; end;
     Inc(Number);                          {Changes to number TO ATTACH}

  if (Debug='Off') and (model='B') then
     begin
         SetColor(Green);
         OutTextXY(560,170,'xxx,yyy');
         end;

     SetTextJustify(CenterText,TopText);
     SetColor(LightGreen);

     case model of
         'B' : Title:='Ballistic Model';
         'D' : Title:='Diffusion Model';
         'M' : Title:=I2S(MaxDiffMoves*5)+'% Diffusion';
         end;

     OutTextXY(559,10,Title);

     If Auto='Auto' then
     OutTextXY(559,30,'Automatic Run # '+I2S(N)+'/'+I2S(Runs));
     If Auto='Manual' then OutTextXY(560,30,'Manual Operation');

     SetTextJustify(LeftText,TopText);

     SetColor(Black);                      {Resets circles from previous runs}
     Circle(CentreX,CentreY,Radius);
     Radius:=240;
     SetColor(LightMagenta);
     Circle(centreX,centreY,radius);

     for I:=2 to number do begin
     PosX[I]:=0; PosY[I]:=0; PosZ[I]:=0; Size[I]:=0; end;

     for I:=2 to number do begin           {repeat until all particles attached}
        SetColor(2);
        OutTextXY(575,130,R2S((I-1),4,0));
        flag:=false;
        size[I]:=minsize+random(maxsize-minsize+1);
        {+1 because random(1) can only = 0}
        currsize:=size[I];

        if (model<>'B') then begin
           SetColor(black);
           OutTextXY(30,450,I2S(radius));
           Circle(centrex,centrey,radius);
           Radius:=width(I-1);
           SetColor(lightmagenta);
           OutTextXY(30,450,I2S(radius));
           Circle(centreX,centreY,radius);
           end;

        if model='B' then repeat
           if keypressed then DebugControl;
           BallStep(Dimensions, radius);
           If dimensions=2 then for J:=1 to I-1 do test2D(I,J, 'B')
           else for J:=1 to I-1 do test3D(I,J, 'B');
           until flag;                     {END Ballistic step}

        if model='D' then begin
```

```
          Start;
          repeat
                if keypressed then DebugControl;
                DiffStep(Dimensions,I,CurrSize);
                If dimensions=2 then for J:=1 to I-1 do test2D(I,J,'D')
                else for J:=1 to I-1 do test3D(I,J, 'D');
                until flag;
          end;                                  {END diffusion steps}

      if model='M' then begin

         redo:
         BalMoves:=0;
         DiffMoves:=0;
         Start;
         OrigDir:=Random(360)/DegRad;
         repeat
               if keypressed then DebugControl;
               if BalMoves < MaxBalMoves then begin
               MixStep(Dimensions,I,CurrSize,OrigDir);
               Inc(BalMoves);
               if where>radius then goto redo;
               end;

               if (BalMoves = MaxBalMoves) and (DiffMoves < MaxDiffMoves)
                  and (flag=false) then begin
                  DiffStep(Dimensions,I,CurrSize);
                  Inc(DiffMoves);
                  end;

               if (BalMoves=MaxBalMoves) and (DiffMoves=MaxDiffMoves) then
   begin
                  BalMoves:=0;
                  DiffMoves:=0;
                  end;

               if dimensions=2 then for J:=1 to I-1 do test2D(I,J,'D')
               else for J:=1 to I-1 do test3D(I,J, 'D');

               until flag;

      end;                  {ENDS model = 'M'}

      if Dimensions=2 then attach2D(I)
      else attach3D(I);

      if (model='B') and (debug='On') then line(Xin,Yin,PosX[I],PosY[I]);

      if ((model='D') or (model='M')) and (debug='On') then begin
         {Replace circle}
         SetColor(LightMagenta);
         Circle(CentreX,CentreY,radius);
         if dimensions=3 then begin
            IntSort(0, I-1, PosZ, PosX, PosY, Size);
            for K:=1 to I do Particle(PosX[K],PosY[K],size[K],Red);
            end;
         end;

      if ((model='D') or (model='M')) or (Debug='On') then begin
         {Display particle coordinates}
         SetTextJustify(RightText,TopText);
         if dimensions=2 then DelText(599,170,I2S(PosX[I-1])+','+I2S(PosY[I-
1]))
         else DelText(599,170,I2S(PosX[I-1])+','+I2S(PosY[I-1])+','+I2S(PosZ[I-
1]));

         SetColor(Green);
         if dimensions=2 then OutTextXY(599,170,I2S(PosX[I])+','+I2S(PosY[I]))
         else
OutTextXY(599,170,I2S(PosX[I])+','+I2S(PosY[I])+','+I2S(PosZ[I]));

         SetTextJustify(LeftText,TopText);
         end;

      if Debug='On' then Delay(Wait);

      if I <= number-1 then DelText(575,130,R2S((I-1),4,0));
      {Replaces all but last number}

      if dimensions=3 then begin
         Writeln(Storage,PosX[I]);
         Writeln(Storage,PosY[I]);
         Writeln(Storage,PosZ[I]);
         Writeln(Storage,Size[I]);
         end;

      end;                              {ENDS agglomerate building (for..to.. do}

      if dimensions=3 then Close(Storage);

      GetTime(h,m,s,hund);
      EndTime:=(h*3600)+(m*60)+s+(hund/100);
      If EndTime<GoTime then EndTime:=EndTime+(24*3600);        {Past Midnight}
      Elapse:=(EndTime-GoTime);
      hour:=round(elapse) div 3600;
      min:=(round(elapse) mod 3600) div 60;
      sec:=round(elapse) - (3600 * hour) - (60 * min);
      TimeTxt := 'Stopwatch :
'+I2S(hour)+':'+I2S(min)+':'+I2S(sec)+'.'+I2S(hund)+'s';

      SetColor(Green);
      OutTextXY(450,420,TimeTxt);
      OutTextXY(450,440,Save);

      if GraphDir='L' then begin
         Sound(1000); Delay(150);
         Sound(500); Delay(150); NoSound;
         end;

      {Redraws agglomerate}
      if ((Debug='On') or (Dimensions=3)) and (Auto='Manual') then begin
         OuttextXY(450,460,'Press ENTER to redraw');
         Readln;
         SetColor(LightMagenta);
         SetFillStyle(SolidFill,Black);
         FillEllipse(CentreX,CentreY,Radius,Radius);

         if Dimensions=2 then Particle(PosX[1],PosY[1],size[1],Blue)
         else begin
                  IntSort(0, I-1, PosZ, PosX, PosY, Size);
                  Particle(PosX[1],PosY[1],size[1],Red);
                  end;
         for I:=2 to number do Particle(PosX[I],PosY[I],size[I],Red);
         DelText(450,460,'Press ENTER to redraw');
         end;

      if Auto='Auto' then begin
         if N<10 then Save:=Root+'auto_0'+I2S(N)+'.DAT'
         else Save:=Root+'auto_'+I2S(N)+'.DAT';
         AggStore;
         end

      else begin
          SetColor(Green);
          OuttextXY(450,380,'Press F2 to save & return');
          if Dimensions=3 then OuttextXY(450,400,'Press F3 to rotate
agglomerate');
          OuttextXY(450,460,'Press any key to return');
          option:=ReadKey;
          if option=#0 then begin
             option:=ReadKey;
             if option=#60 then AggStore;
             if option=#61 then spin_menu(PosX, PosY, PosZ,
Size,Number,CentreX,CentreY);
             end;
         end;
      number:=number-1;
      CloseGraph;
end;
{***************************************************************************}
procedure Carlo;

var
   highest : integer;
   xd,yd,zd,dist : single;
   p : integer;

label Falling;
label rollagain;
label atrest;
label retest;

begin
      if ((Auto='Auto') and (N=1)) or (Auto='Manual') then Carlo_Menu;

      if Return then Exit;

      GraphDriver := Detect;                    {Sets and checks graphmode}
      InitGraph(GraphDriver, GraphMode,Path);
      if GraphResult <> grOk then Halt(1);

      Title:='Monte Carlo';
      BuildScr;                                    {SCREEN INFOrmation}

      if (GraphDir='L') and ((Auto='Manual') or (N=1))  then begin
      Sound(600); Delay(75); NoSound; end;

      if (Debug='Off') then begin
         SetColor(Green);
         case Dimensions of
             2 : OutTextXY(560,170,'xxx,yyy');
             3 : OutTextXY(540,170,'xxx,yyy,zzz');
          end;
        end;
      SetTextJustify(CenterText,TopText);
      SetColor(LightGreen);

      OutTextXY(559,10,Title);

      if Auto='Auto' then OutTextXY(559,30,'Automatic Run #
'+I2S(N)+'/'+I2S(Runs))
        else OutTextXY(560,30,'Manual Operation');

      SetTextJustify(LeftText,TopText);

      Highest:=F1;

      for I:=1 to number do begin
          PosX[I]:=0; PosY[I]:=0; PosZ[I]:=0; Size[I]:=0; end;

      for I:=1 to number do begin         {repeat until all particles attached}

        SetColor(Green);
        OutTextXY(575,130,R2S(I,4,0));
        size[I]:=minsize+random(maxsize-minsize+1);
        {+1 because random(1) can only = 0}

        CarloStart;

        repeat
              Falling:
              CarloStep(Dimensions,Size[I]);
              {Allows turning on or off of debugging during simulation}
              if keypressed then DebugControl;

              Flag:=False;
              NotFloor:=False;
              Touch:=0;
              if ((Yout+Size[I])>(Highest-Maxsize)) or (I=1) then
              Case Dimensions of
                   2 : for J:=1 to I do CTest2D(I,J);
                   3 : for J:=1 to I do CTest3D(I,J);
                   end;
              until flag;

        Touch:=1;

        if (NotFloor) then floor:=false;

        if Dimensions=2 then CarloAttach2D(I)
        else CarloAttach3D(I);

        retest:

        if not floor then begin

        if Random<=(StickProb/100) then goto atrest;
        if Dimensions=2 then Roll2D(RollFrom)
        else Roll3D(RollFrom);

        case condition of
             'f' : begin
                   RollFrom:=0; Roll1:=0; Roll2:=0;
```

```
                    XStart:=PosX[I]; YStart:=PosY[I];
                    FreeFall(Dimensions,RollFrom,1);
                    if condition='f' then goto falling;
                    end;
                'b','w','n' : goto atrest;
                'p' : begin
                    for J:=1 to I-1 do Hit[J]:=False;
                    Hit[RollFrom]:=True;
                    goto retest;
                    end;
                end;
            end;                              {End 'not floor'}

            atrest:

            Particle(PosX[I],PosY[I],Size[I],Red);

            if PosY[I]-Size[I]<Highest then Highest:=PosY[I]-Size[I];

            Roll1:=0; Roll2:=0;

            if I < number then DelText(575,130,R2S(I,4,0));
            {Replaces all but last number}

            if Debug='On' then begin
              SetTextJustify(RightText,TopText);
              if dimensions=2 then DelText(599,170,I2S(PosX[I-1])+','+I2S(PosY[I-
1]))
                else DelText(599,170,I2S(PosX[I-1])+','+I2S(PosY[I-
1])+','+I2S(PosZ[I-1]));
              {Display particle coordinates}
              SetColor(Green);
              if dimensions=2 then
OutTextXY(599,170,I2S(PosX[I])+','+I2S(PosY[I]))
                else
OutTextXY(599,170,I2S(PosX[I])+','+I2S(PosY[I])+','+I2S(PosZ[I]));
              end;            {End debug='On'}

            SetTextJustify(LeftText,TopText);
            end;            {End for I:=1 to number loop}

        for I:=2 to number do Particle(PosX[I],PosY[I],size[I],Red);
        GetTime(h,m,s,hund);
        EndTime:=(h*3600)+(m*60)+s+(hund/100);
        If EndTime<GoTime then EndTime:=EndTime+(24*3600);       {Past Midnight}
        Elapse:=(EndTime-GoTime);
        hour:=round(elapse) div 3600;
        min:=(round(elapse) mod 3600) div 60;
        sec:=round(elapse) - (3600 * hour) - (60 * min);
        TimeTxt:='Stopwatch :
'+I2S(hour)+':'+I2S(min)+':'+I2S(sec)+'.'+I2S(hund)+'s';

        SetColor(Green);
        OutTextXY(450,420,TimeTxt);
        OutTextXY(450,440,Save);

        if (GraphDir='L') and (Auto='Manual') then begin
        Sound(1000); Delay(150);
        Sound(500); Delay(150); NoSound;
        end;
```

```
    if Auto='Auto' then begin
        if N<10 then Save:=Root+'auto_0'+I2S(N)+'.DAT'
        else Save:=Root+'auto_'+I2S(N)+'.DAT';
        FiltStore;
        if N=Runs then begin
           Sound(1000); Delay(150);
           Sound(500); Delay(150); NoSound;
           end;
        end

    else begin
    OuttextXY(450,380,'Press F2 to save & return');
    if Dimensions=3 then OuttextXY(450,400,'Press F3 to rotate agglomerate');
    OuttextXY(450,460,'Press any key to return');
    option:=ReadKey;
    if option=#0 then begin
       option:=ReadKey;
       if option=#60 then FiltStore;
       if option=#61 then spin_menu(PosX, PosY, PosZ,
Size,Number,CentreX,CentreY);
       end;
       end;
    CloseGraph;
end;
{*************************************************************************}
procedure Simulate;

var
   AutoDat, Autogyr : Text;
   DatExists, GyrExists : Boolean;

label
     Stop;

begin

     if ((Auto='Auto') and (N=1)) or (Auto='Manual') then begin
        GetInfo;
        if Return then exit;

        if OverWrite=True then begin
           K:=0;
           repeat
           {$I-}
           Inc(K);
           if K<10 then begin
              Assign(AutoDat,Root+'Auto_0'+I2S(K)+'.dat');
              Assign(AutoGyr,Root+'Auto_0'+I2S(K)+'.gyr');
              end
           else begin
              Assign(AutoDat,Root+'Auto_'+I2S(K)+'.dat');
              Assign(AutoGyr,Root+'Auto_'+I2S(K)+'.gyr');
              end;
           Reset(AutoDat); Close(AutoDat);
           DatExists := (IOResult = 0);
           if DatExists then Erase(AutoDat);
           Reset(AutoGyr); Close(AutoGyr);
           GyrExists := (IOResult = 0);
           if GyrExists then Erase(AutoGyr);
           until not DatExists;
           {$I-}
```

```
            end;
         end;

         OverWrite:=False;

         if Auto='Auto' then for N:=1 to Runs do begin
            if Model='F' then Carlo else Agglomerate;
            if Return then goto Stop;
            end

         else if Model='F' then Carlo else Agglomerate;

         Stop:

      end;
      {***************************************************************************}
      end.
```

MENUS.PAS - Contains all the text based menus for user input.

NB. Make Window is contained in SCREENINFO.PAS

```
unit menus;

{$N+,E+}

interface

uses Dos, Crt, disk, funcs, screeninfo, variables;

procedure Setup;
procedure GetInfo;
procedure Pore_Menu;
procedure Trace_Menu;
procedure Ana_menu;
procedure Mixed_Menu;
procedure Carlo_Menu;
procedure Display_Menu;

implementation

var
FileExists : boolean;

{**********************************************************************}
procedure GetFile;

var
   K,Last : integer;
   Temp, check : string;

begin
if not PoreMeasure then begin
     Window(6,17,70,22);
     Writeln('Default directory is '+Root);
     Writeln;
     Writeln('Please enter sub-directory and file name');
     Writeln;
     Writeln('Do not use file extension');
end;
     repeat
           if not PoreMeasure then begin
              Window(16,3,62,11);
              Temp:=Name;
              GotoXY(13,3);ClrEol;readln(Name);
              Check:='';
              for K:=length(Name)-3 to length(Name) do Check:=Check+Name[K];
              if check='.dat' then Name:=Remlet(Name,4);
              multi:=Name[Length(Name)]='*';
              {Increment file name if blank response}
              if Name='' then begin
                 Val(Temp[Length(Temp)-1]+Temp[Length(Temp)],Last,Code);
                 if Last+1>=10 then Name:=RemLet(Temp,2)+I2S(Last+1)
                 else Name:=RemLet(Temp,2)+'0'+I2S(Last+1);
                 end;
              end;
           if not Multi then begin
              Save:=Root+Name+'.dat';
              Gyr:=Root+Name+'.gyr';
```

```
            Assign(Storage,Save);
            {$I-}
            Reset(Storage);
            Close(Storage);
            FileExists := (IOResult = 0) and (Save <> '');
            {$I+}
            end                     {ends NOT multi}
         else begin            {begin multi}
            if PoreMeasure then begin
               Stem:=RemLet(PoreFile, 1);
               Save:=Root+PoreStem+Stem+'01'+'.por';
               end
            else begin
               Stem:=RemLet(Name, 1);
               Save:=Root+Stem+'01'+'.dat';
               end;
            Assign(Storage,Save);
            {$I-}
            Reset(Storage);
            Close(Storage);
            FileExists := (IOResult = 0) and (Save <> '');
            {$I+}
            if FileExists then begin
               J:=1;
               repeat
                     J:=J+1;
                     if J<10 then Name:=stem+'0'+I2S(J)
                     else Name:=stem+I2S(J);
                     if PoreMeasure then Save:=Root+PoreStem+Name+'.por'
                     else Save:=Root+Name+'.dat';
                     Assign(Storage,Save);
                     {$I-}
                     Reset(Storage);
                     Close(Storage);
                     FileExists := (IOResult = 0) and (Save <> '');
                     {$I+}
                     until not FileExists or (J=99);
               FileExists:=True;
               if J=99 then NumFiles:=J else NumFiles:=J-1;
               Stem:=RemLet(Name, 2);
               if Numfiles<10 then Name:=stem+'0'+I2S(Numfiles)
               else Name:=stem+I2S(Numfiles);
               if PoreMeasure then Save:=Root+PoreStem+Name+'.por'
               else Save:=Root+Name+'.dat';
               end;
            end;                    {ends multi}
            If not FileExists then begin
               Window(6,13,70,16);
               ClrScr;
               Sound(100);Delay(200);NoSound;
               TextColor(Green+Blink); GotoXY(25,1);
               CursorOff;
               Writeln('File does not exist!');
               TextColor(Green); GotoXY(19,3);
               Write('Please enter a valid file name');
               delay(2000);
               CursorOn;
               end;
            until FileExists;
      Assign(Storage,Save);
      Reset(Storage);
```

```
     Readln(Storage, Title);
     FileType:='N';              {Sets default FileType}
     case Title[1] of
          'S','P' : begin Dimensions:=2; FileType:='S'; end;
          'M' : begin
                FileType:='M';
                Model:='F';
                end;
          'B' : Model:='B';
          'D' : Model:='D';
          else Model:='M';
     end;

     if FileType<>'S' then Read(Storage,Dimensions);

     Close(Storage);
     end;
{**************************************************************************}
procedure Disabled(M : integer);

begin
     TextColor(DarkGray);
     GotoXY(1,M+2);
     ClrEol;
     Writeln(' '+I2S(M)+'.  Menu item disabled');
     TextColor(Green);
end;
{**************************************************************************}
procedure SetRand;

var
   J : integer;
   Z : integer;
   a,b,c,d,e,f : integer;
   SetTime : string;

begin
     GetTime(h,m,s,hund);
     Str(h,hr); Str(m,mn); Str(s,sc);
     Time:=(h*3600+m*60+s);
     MakeWindow(1);
     Writeln(' 1.  Random Seed : ',hr+':'+mn+':'+sc);
     Repeat
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then
     begin
          Writeln; Writeln;
          Write('Enter number of option to change : ');
          option:=Readkey;
                Case option of
                    '1' : begin
                          Writeln;
                          Writeln('Please enter Random Seed as 24-hour Clock
with leading zeros.');
                          Writeln;
                          Writeln('e.g.  12:06:59');
                          Window(16,3,62,11);
                          GotoXY(20,3);ClrEol;readln(SetTime);
                          numgen:='Set';
                                 val(SetTime[1],a,Z);
                                 val(SetTime[2],b,Z);
```

```
                                        hr:=SetTime[1]+SetTime[2];
                                        val(SetTime[4],c,Z);
                                        val(SetTime[5],d,Z);
                                        mn:=SetTime[4]+SetTime[5];
                                        val(SetTime[7],e,Z);
                                        val(SetTime[8],f,Z);
                                        sc:=SetTime[7]+SetTime[8];
                                        h:=a*10+b; m:=c*10+d; s:=e*10+f;
                                        Time:=(h*3600+m*60+s);
                               end;

                 end;
                                     Window(6,16,70,19);
                                     ClrScr;
      end;
         until (Upcase(yesno)<>'Y') ;
         SetUp;
end;
{***************************************************************************}
procedure SetSides;
begin
     MakeWindow(3);
     Writeln(' 1.  Number of sides : ',Sides);
     Write(' 2.  Shape (Regular/Irregular) : ',Quality);
     TextColor(LightGreen);
     GotoXY(13,4);Write('R');GotoXY(21,4);Writeln('I');
     TextColor(Green);
     Writeln(' 3.  Rotation (Degrees) : ',Rotate);
     Repeat
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then
     begin
          Writeln; Writeln;
          Write('Enter number of option to change : ');
          option:=Readkey;
          ClrScr;
          Window(16,3,62,12);
                Case option of
                    '1' : begin
                          GotoXY(24,3);ClrEol;
                          Readln(Sides);
                          end;
                    '2' : begin
                          GotoXY(34,4);ClrEol;
                          option:=UpCase(ReadKey);
                          Case Option of
                               'R' : Quality:='Reg';
                               'I' : Quality:='Irr';
                               end;
                          Writeln(Quality);
                          end;          {Ends choice '2'}
                    '3' : begin
                          GotoXY(27,5);ClrEol;
                          Readln(Rotate);
                          end;
                end       {END case}
          end;            {END choices}
        until (Upcase(yesno)<>'Y') ;
Setup;
end;
{***************************************************************************}
```

```
procedure Setup;                              {System Defaults}

var
   FileExists : boolean;
   Stem : String;
   Srec : SearchRec;
   Error : integer;

label
     man, OK;

begin
     MakeWindow(9);
     Writeln(' 1.  Number of dimensions (2/3) : ',Dimensions);
     Write(' 2.  Shape (Circles/Polygons) : ',Shape);
     TextColor(LightGreen);
     GotoXY(13,4);Write('C');GotoXY(21,4);Writeln('P');
     TextColor(Green);
     Write(' 3.  Random Number Generator (0/1/Set) : ',numgen);
     TextColor(LightGreen);
     GotoXY(35,5);Writeln('S');
     TextColor(Green);
     Writeln(' 4.  Visual Debugging (0/1) : ',debug);
     Writeln(' 5.  Visual Delay (ms) : ',wait);
     Write(' 6.  Manual/Automatic control : ',Auto);
     TextColor(LightGreen);
     GotoXY(6,8);Write('M');GotoXY(13,8);Writeln('A');
     TextColor(Green);
     if Auto='Manual' then Disabled(7)
     else Writeln(' 7.  Number of Automatic Runs : ',Runs);
     Writeln(' 8.  Data dir. : '+Shorten(Root));

     FileExists:=NetThere and LocalThere;
     If FileExists then
     begin
          Write(' 9.  Graph dir (Local/Network): '+GraphDir);
          TextColor(LightGreen);
          GotoXY(17,11);Write('L');GotoXY(23,11);Writeln('N');
          end
     else Disabled(9);
     CheckSubs;
     Repeat
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then
     begin
          Writeln; Writeln;
          Write('Enter number of option to change : ');
          option:=Readkey;
          ClrScr;
          Window(16,3,62,14);
                 Case option of
                      '1' : begin
                            GotoXY(35,3);ClrEol;
                            option:=(ReadKey);
                            Case Option of
                                 '2' : Dimensions:=2;
                                 '3' : Dimensions:=3;
                                 end;
                            Writeln(Dimensions);
                            end;
                      '2' : begin
```

```
                                        GotoXY(33,4);ClrEol;
                                        option:=UpCase(ReadKey);
                                        Case Option of
                                             'C' : Shape:='Circle';
                                             'P' : Shape:='Poly';
                                             end;
                                        if Shape='Poly' then SetSides;
                                        Writeln(Shape);
                                        end;         {Ends choice '2'}
                                  '3' : begin
                                        GotoXY(42,5);ClrEol;
                                        option:=UpCase(ReadKey);
                                        Case Option of
                                             '0' : numgen:='Off';
                                             '1' : numgen:='On';
                                             'S' : SetRand;
                                        end;
                                        Writeln(numgen);
                                        end;
                                  '4' : begin
                                        GotoXY(31,6);ClrEol;
                                        option:=ReadKey;
                                        Case Option of
                                             '0' : Debug:='Off';
                                             '1' : Debug:='On';
                                        end;
                                        Writeln(debug);
                                        end;
                                  '5' : begin
                                        GotoXY(26,7);ClrEol;readln(wait);
                                        end;
                                  '6' : begin
                                        GotoXY(33,8);ClrEol;
                                        option:=UpCase(ReadKey);
                                        man:
                                        Case Option of
                                             'A' : Auto:='Auto';
                                             'M' : Auto:='Manual';
                                        end;
                                        Writeln(Auto);
                                        if Auto='Manual' then Disabled(7)
                                        else begin
                                           Assign(Storage,Root+'\auto_01.dat');
                                           {$I-}
                                           Reset(Storage);
                                           Close(Storage);
                                           FileExists := (IOResult = 0);
                                           {$I+}
                                           If FileExists then
                                              begin
                                                   Window(6,16,70,21);
                                                   ClrScr;
                                                   Sound(100);Delay(200);NoSound;
                                                   TextColor(Green+Blink); GotoXY(19,1);
                                                   Writeln('Automatic file(s) already
               exist!');

                                                   TextColor(Green); GotoXY(24,3);
                                                   Write('Overwrite Files Y/[N] ? ');
                                                   Option:=UpCase(ReadKey);
                                                   ClrScr;
                                              case Option of
                                        'Y' : begin
                                              Window(16,3,62,14);
                                              GotoXY(1,9);
                                              Write(' 7.  Number of Automatic
Runs : ',Runs);

                                              OverWrite:=True;
                                              end;
                                        else begin
                                              Option:='M';
                                              Window(16,3,62,14);
                                              GotoXY(33,8);ClrEol;
                                              Goto man
                                              end;
                                        end          {end CASE Y/N}
                                   end          {end file exists}
                                   else begin
                                        Window(16,3,62,14);
                                        GotoXY(1,9);
                                        Write(' 7.  Number of Automatic Runs :
',Runs);

                                   end;
                                end;          {end Auto check}
                             end;          {end menu item 6}
                       '7' : begin
                             GotoXY(33,9);ClrEol;readln(Runs);
                             end;
                       '8' : begin
                             repeat
                             Window(6,16,70,21);
                             ClrScr;
                             Writeln;
                             Write('Please enter FULL path');
                             Window(16,3,62,14);
                             GotoXY(18,10);ClrEol;readln(Root);
                             {Add final '\'}
                             if Root[length(Root)]<>'\' then Root:=Root+'\';

                             FindFirst(Root+'*.*',AnyFile,Srec);

                             If DosError <> 0 then begin
                                Window(6,16,70,21);
                                ClrScr;
                                Sound(100);Delay(200);NoSound;
                                TextColor(Green+Blink); GotoXY(22,1);
                                Writeln('Directory does not exist!');
                                TextColor(Green); GotoXY(22,3);
                                Write('Create Directory Y/[N] ? ');
                                Option:=UpCase(ReadKey);
                                ClrScr;
                                case Option of
                                     'Y' : begin
                                           {Remove '\' to create directory}
                                           Root:=RemLet(Root,1);
                                           {Create Directory + subs}
                                           MkDir(Root);
                                           MkDir(Root+'\animator');
                                           MkDir(Root+'\2d');
                                           MkDir(Root+'\3d');
                                           MkDir(Root+'\3d\povray');
                                           Root:=Root+'\';
                                           Goto OK;
                                        end;
                                   else begin
                                        TextColor(Green); GotoXY(1,5);
                                        Write('Please enter a valid
directory');
                                        delay(2500);
                                        end;
                              end          {end CASE Y/N}
                              end;         {end doesn't exist}

                         CheckSubs;
                         If Created then goto OK;

                         until DosError=0;
                            OK:
                            Created:=False;
                            Window(16,3,62,14);
                            GotoXY(1,10); ClrEol;
                            Writeln(' 8.  Data dir. : '+Root);
                         end;                    {end Choice 8}
                    '9' : begin
                         GotoXY(33,11);ClrEol;
                         Option:=UpCase(ReadKey);
                         Case Option of
                              'L' : Path:='c:\tp\bgi';
                              'N' : Path:='z:\site\tp6\bgi';
                         end;
                         GraphDir:=Option;
                         Writeln(GraphDir);
                         end;
               end                               {end CASE number}
          end;
       until (Upcase(yesno)<>'Y') ;
end;
{*************************************************************************}
procedure GetInfo;                      {Input information about build}

var
   FileExists : boolean;
   K : integer;
   Check : string;

label
   man;

begin
      Save:=Root+Name+'.dat';
      If Auto='Auto' then save:='Automatic creation';

      repeat
      MakeWindow(7);
      Writeln(' 1.  File : ',Shorten(Save));
      Writeln(' 2.  Number of particles to attach : ',number);
      if (Model='F') then Disabled(3)
      else Writeln(' 3.  Seed particle radius : ',seed);
      Writeln(' 4.  Minimum particle radius : ',MinSize);
      Writeln(' 5.  Maximum particle radius : ',MaxSize);
      Write(' 6.  Model type : ');
      Case Model of
           'B' : Writeln('Ballistic');
           'D' : Writeln('Diffusion');
```

```
     'M' : Writeln('Mixed Mechanism');
     'F' : Writeln('Filter Simulation');
     end;
TextColor(Green+Blink);
Write(' B');
TextColor(Green);
Writeln('.  Back to main menu');

if (Auto='Auto') and (OverWrite=False) then begin
   Assign(Storage,Root+'\auto_01.dat');
   {$I-}
   Reset(Storage);
   Close(Storage);
   FileExists := (IOResult = 0);
   {$I+}
   If FileExists then begin
      Window(6,16,70,21);
      ClrScr;
      Sound(100);Delay(200);NoSound;
      TextColor(Green+Blink); GotoXY(19,1);
      Writeln('Automatic file(s) already exist!');
      TextColor(Green); GotoXY(24,3);
      Write('Overwrite Files Y/[N] ? ');
      Option:=UpCase(ReadKey);
      ClrScr;
      case Option of
           'Y' : begin
                 OverWrite:=True;
                 end;
           else begin
                 Auto:='Manual';
                 Save:=Root+'agglom.dat';
                 Window(16,3,62,11);
                 GotoXY(13,3);ClrEol;
                 Write(Save);
                 Goto man;
                 end;
           end;          {end CASE Y/N}
      end;         {end file exists}
      end;         {end Auto}

man:
Ask;
yesno:=ReadKey; if UpCase(yesno)='Y' then
begin
     Writeln; Writeln;
     Write('Enter number of option to change : ');
     option:=Readkey;
     ClrScr;
     Window(16,3,62,11);
           Case option of
               '1' : begin
                     repeat
                     Window(6,17,70,22);
                     Writeln('Default directory is '+Root);
                     Writeln;
                     Writeln('Please enter file name only');
                     Writeln;
                     Writeln('Do not use file extension');
                     Window(16,3,62,11);
                     GotoXY(13,3);ClrEol;readln(Name);
```

```
                    Check:='';
                    for K:=length(Name)-3 to length(Name) do
Check:=Check+Name[K];
                    if check='.dat' then Name:=Remlet(Name,4);
                    if Dimensions=3 then begin
                       POV:=Root+Name+'.pov';
                       GYR:=Root+Name+'.gyr';
                       end;
                    Case model of
                    'B','D','M' : Save:=Root+Name+'.dat'
                    else Save:=Root+Name+'.dat';
                    end;
                    Assign(Storage,save);
                    {$I-}
                    Reset(Storage);
                    Close(Storage);
                    FileExists := (IOResult = 0) and (Save <> '');
                    {$I+}
                    If FileExists then
                       begin
                            Window(6,14,70,18);
                            ClrScr;
                            Sound(100);Delay(200);NoSound;
                            TextColor(Green+Blink); GotoXY(25,1);
                            Writeln('File already exists!');
                            TextColor(Green); GotoXY(24,3);
                            Write('Overwrite File Y/[N] ? ');
                            Option:=UpCase(ReadKey);
                            ClrScr;
                       end;
                       Window(6,16,70,21);
                       ClrScr;
                       until (Option='Y') or (not FileExists);
                  end;
               '2' : begin
                    GotoXY(38,4);ClrEol;readln(number);
                    end;
               '3' : begin
                    GotoXY(29,5);ClrEol;readln(seed);
                    end;
               '4' : begin
                    GotoXY(32,6);ClrEol;readln(minsize);
                    if minsize>maxsize then maxsize:=minsize;
                    end;
               '5' : begin
                    GotoXY(32,7);ClrEol;readln(maxsize);
                    if maxsize<minsize then minsize:=maxsize;
                    end;
               '6' : begin
                    Window(6,16,70,21);
                    Writeln('Please choose from :');
                    Writeln;
                    Write('Ballistic, Diffusion, ');
                    Writeln('Mixed Mechanism & Filter Simulation');
                    TextColor(LightGreen);
                    GotoXY(1,3);Write('B');
                    GotoXY(12,3);write('D');
                    GotoXY(23,3);Writeln('M');
                    GotoXY(41,3);Writeln('F');
                    TextColor(Green);
                    Window(16,3,62,11);
                    GotoXY(19,8);ClrEol;
                    option:=ReadKey;
                    option:=Upcase(option);
                    Case Option of
                         'B' : model:='B';
                         'D' : model:='D';
                         'M' : model:='M';
                         'F' : model:='F';
                    end;                                {End case Option}
                    {Set defaults for different model types}
                    Case model of
                         'B','D','M' : begin
                         Number:=800; MinSize:=3; Maxsize:=3; seed:=3;
                         end;
                         'F' : begin
                         Number:=1999; MinSize:=8; Maxsize:=10;
                         end;
                    end;                                {End case Model}
                    end;
               end
          end;
         until ((Upcase(yesno)<>'Y') and (minsize <= maxsize));
   if Upcase(yesno)='B' then Return:=True;
   end;
{**********************************************************************}
procedure Trace_Menu;
begin
     MakeWindow(3);
     Writeln(' 1.  Height of image (pels) : ',RayHeight);
     Writeln(' 2.  Width of image (pels) : ',RayWidth);
     Writeln(' 3.  Number of animation frames : ',Frames);
     Repeat
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then
     begin
          Writeln; Writeln;
          Write('Enter number of option to change : ');
          option:=Readkey;
          ClrScr;
          if (option='1') or (option='2') then
          Writeln('Aspect ratio automatically maintained');
          Window(16,3,62,11);

               Case option of
                   '1' : begin
                         GotoXY(31,3);ClrEol;readln(RayHeight);
                         RayWidth:=round(RayHeight*4/3);
                         GotoXY(30,4);ClrEol;Writeln(RayWidth);
                         end;
                   '2' : begin
                         GotoXY(30,4);ClrEol;readln(RayWidth);
                         RayHeight:=round(RayWidth*3/4);
                         GotoXY(31,3);ClrEol;Writeln(RayHeight);
                         end;
                   '3' : begin
                         GotoXY(35,5);ClrEol;readln(Frames);
                         end;
               end
         end;
         until (Upcase(yesno)<>'Y');
end;
```

```
{****************************************************************************}
procedure SetRange(call : char);

var
Sim : char;

begin
    if Used=1 then begin
       LowFrac:=OldLow;
       UppFrac:=OldUpp;
       Used:=0;
       End;
    range:='Yes';
    Repeat
    MakeWindow(3);
    Writeln(' 1.  Lower value : ',R2S(LowFrac,1,3));
    Writeln(' 2.  Upper value : ',R2S(UppFrac,1,3));
    Writeln(' 3.  Increment : ',R2S(Gap,1,3));
    Ask;
    yesno:=ReadKey; if UpCase(yesno)='Y' then
    begin
         Writeln; Writeln;
         Write('Enter number of option to change : ');
         Option:=Readkey;
         ClrScr;
         Window(16,3,62,11);

               Case Option of
                   '1' : begin
                         Window(16,3,62,11);
                         GotoXY(20,3);ClrEol;readln(LowFrac);
                         end;
                   '2' : begin
                         Window(16,3,62,11);
                         GotoXY(20,4);ClrEol;readln(UppFrac);
                         end;
                   '3' : begin
                         Window(16,3,62,11);
                         GotoXY(18,5);ClrEol;readln(Gap);
                         end;
               end;
               Window(6,16,70,19);
               ClrScr;
    end;
       until (Upcase(yesno)<>'Y') ;
if Call='A' then Ana_Menu
else Pore_Menu
end;
{****************************************************************************}
procedure GetLatest(Name : string; Takeoff : integer; AddRoot : boolean);
{Gets last PoreFile saved if no other inputted / first time}

var
   NameStart, LenName : integer;

begin
    NameStart:=0;
    for J:=1 to Length(Name) do if Name[J]='\' then NameStart:=J+1;
    LenName:=Length(Name)-NameStart;
    if LenName>5 then Name:=RemLet(Name,Takeoff);
    if AddRoot then Name:=Root+Name+'01.por'
```

```
      else Name:=Name+'01.por';
      Assign(Storage,Name);
      {$I-}
      Reset(Storage);
      Close(Storage);
      FileExists := (IOResult = 0);
      {$I+}
      J:=0;
      if FileExists then repeat
         J:=J+1;
         Name:=RemLet(Name, 6);
         if J<10 then Name:=Name+'0'+I2S(J)+'.por'
         else Name:=Name+I2S(J)+'.por';
         Assign(Storage,Name);
      {$I-}
      Reset(Storage);
      Close(Storage);
      FileExists := (IOResult = 0);
      {$I+}
      if (J=99) and FileExists then begin
         Window(6,13,70,16);
         ClrScr;
         Sound(100);Delay(200);NoSound;
         TextColor(Green+Blink); GotoXY(25,1);
         CursorOff;
         Writeln('No more pores possible!');
         Delay(2000);
         CursorOn;
         PoreFile:=Name;
         NewPore:=False;
         exit;
         end;
      until Not FileExists;
      PoreFile:=Name;
end;
{*****************************************************************************}
procedure Pore_Menu;

var
   Response, IDNumber : string;

begin

Multi:=False;
if PoreMulti=True then Multi:=True;
PoreMulti:=False;
ParentName:=Name;
GetLatest(Name,2,True);
IDNumber:='';
for J:=Length(PoreFile)-5 to Length(PoreFile)-4 do
IDNumber:=IDNumber+PoreFile[J];
Val(IDNumber,Numfiles,code);

If Numfiles=1 then NewPore:=True else NewPore:=False;
NumFiles:=Numfiles-1;
Str(Numfiles:0,IDNumber);
If Numfiles<10 then IDNumber:='0'+IDNumber;
If (not NewPore) then PoreFile:=remlet(PoreFile,6)+IDNumber+'.por';

repeat
      MakeWindow(4);
```

```
    GotoXY(1,3);
    if not multi then Writeln(' 1.  File : ',Shorten(PoreFile))
    else begin Write(' 1.  File : ',Shorten(PoreName));
         GotoXY(12+Length(Shorten(PoreName)),3);
         ClrEol;
         Writeln('('+IDNumber+')');
         end;
    if NewPore then begin
       disabled(2);
       disabled(3);
       end
    else begin
    Write(' 2.  Analysis method : ');
    Case PropMeth of
         'S' : Writeln('Structured Walk');
         'E' : begin
               EncMeth:='O';
               Writeln('Enclosing Circle');
               end;
         end;
    if (PropMeth='E') and (Ripple>40) then Ripple:=40
    else if Ripple>32 then Ripple:=32;
    if (PropMeth='E') and (Ripple=32) then Ripple:=40;
    if PropMeth='S' then begin
         if Range='Yes' then Writeln(' 3.  Feret Fraction :
',R2S(LowFrac,1,3),'-',R2S(UppFrac,1,3),' @ ',R2S(Gap,1,3))
         else Writeln(' 3.  Feret Fraction : ',R2S(fraction,1,3));
         end
    else Writeln(' 3.  Number of points : ',ripple);
    end;                          {End not NewPore}
    case Method of
    'S' : begin
          case Dimensions of
          3 : begin
              Write(' 5.  Aspect (Front/Side/Top) : ',View);
              TextColor(LightGreen);
              GotoXY(14,7); Writeln('F');
              GotoXY(20,7); Writeln('S');
              GotoXY(25,7); Write('T');
              TextColor(Green);
              Writeln;
              end;      {End 3 Dimensions}
          end;          {End 'dimensions' case}
         end;          {End Structured walk}
         end;     {End 2-D Model 'F'}
    TextColor(Green+Blink);
    Write(' B');
    TextColor(Green);
    Writeln('.  Back to main menu');
    Ask;
    yesno:=ReadKey; if UpCase(yesno)='Y' then begin
      Writeln; Writeln;
      Write('Enter number of option to change : ');
      option:=Readkey;
      ClrScr;
      Case option of
      '1' : begin
            Window(6,17,70,22);
            Writeln('Default directory is '+Root);
            Writeln;
            Writeln('Please enter file name only');
```

```
                Writeln;
                Writeln('CR for new pore');
                Writeln;
                Writeln('Do not use file extension');
                Window(16,3,62,11);
                GotoXY(13,3);ClrEol;readln(PoreName);
                K:=Length(Name);
                for J:=K downto 1 do
                    if (Name[J]='\') and (PoreStem='') then
PoreStem:=RemLet(Name,K-J);
                If PoreName='' then begin
                   NewPore:=True;
                   Multi:=False
                   end
                   else PoreFile:=PoreName;
                If PoreFile[length(PoreFile)]='*' then begin
                   multi:=true;
                   Getfile;
                   NewPore:=False;
                   end
                else begin
                If NewPore then begin
                   PoreFile:=Root+Name;
                   PoreName:=PoreFile;
                   end
                else PoreFile:=Root+PoreStem+PoreFile+'.por';
                Assign(Storage,PoreFile);
                {$I-}
                Reset(Storage);
                Close(Storage);
                FileExists := (IOResult = 0);
                {$I+}
                NewPore:=not FileExists;
                if NewPore then GetLatest(PoreFile,2,False);
                end;
                Window(6,16,70,21);
                ClrScr;
                end;
         '2' : begin
               Window(6,16,70,21);
               Writeln('Please choose from : ');
               Writeln;
               Write('Structured Walk or Enclosing Circle, ');
               TextColor(LightGreen);
               GotoXY(1,3);Write('S');
               GotoXY(20,3);Write('E');
               TextColor(Green);
               Window(16,3,62,11);
               GotoXY(24,4);ClrEol;
               option:=UpCase(ReadKey);
               Case Option of
                    'S' : PropMeth:='S';
                    'E' : PropMeth:='E';
                    end;
               Writeln(Method);
               end;

         '3' : begin
               ClrScr;
               if PropMeth='S' then begin
               Writeln('Please enter fraction - ''R'' for range (''D'' for
default range)');
               Window(16,3,62,11);

{Read character input}
GotoXY(23,5);ClrEol;option:=readkey;

case UpCase(Option) of

'R' : SetRange('P');
'D' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.08; Gap:=0.02; end;
'!' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.1; Gap:=0.018; end;
'"' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.12; Gap:=0.016; end;
'*' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.08; Gap:=0.018; end;

else begin
     if (Option=',') or (Option='/') then Option:='.';
     Write(Option);
     Readln(Response);
     {Add keypress to start of 'response'}
     Response:=Option+Response;
     Range:='No';
     Val(Response,Fraction,Code);
     end;

     end;
     end              {Ends Method='S'}
     else begin             {Method <> 'S'}
               Window (16,3,62,11);
               GotoXY(25,5);ClrEol;readln(Ripple);
               end;
          end;        {Ends choice '3'}
          end
     end;
     until (Upcase(yesno)<>'Y') ;
     if Upcase(yesno)='B' then Return:=True;
     if Range='No' then begin
        OldLow:=LowFrac; OldUpp:=UppFrac;
        Used:=1;
        LowFrac:=Fraction-Gap;
        UppFrac:=Fraction;
        end;
end;
{***********************************************************************}
procedure Ana_menu;                    {Analysis options}

var
   Response : string;

begin
     Model:='B';                            {Sets Default}
     repeat
     MakeWindow(6);
     if Title[1]='M' then begin FileType:='M'; Model:='F'; end;
     if Title='SEM Image (Fibre)' then begin
        FileType:='S';
        SubType:='I'
        end;
     if (FileType='S') and (PoreMeth<>'P')
        and (PoreMeth<>'S') and (PoreMeth<>'O')
        then PoreMeth:='O';
```

```
    if not multi then Writeln(' 1.  File : ',Shorten(Save))
    else begin Write(' 1.  File : ',Shorten(Save));
         GotoXY(7+Length(Shorten(Save)),3);
         ClrEol;
         Writeln('('+I2S(Numfiles)+')');
         end;
    Write(' 2.  Analysis method : ');
    Case Method of
         'S' : Writeln('Structured Walk');
         'E' : Writeln('Enclosing Circle');
         'R' : Writeln('Radius of Gyration');
         'P' : Writeln('Porosity');
         'B' : Writeln('Box Counting');
         end;
    if (Method='E') and (Ripple>40) then Ripple:=40
    else if Ripple>32 then Ripple:=32;
    if (Method='E') and (Ripple=32) then Ripple:=40;
    if (Method='S') or (Method='B') then begin
         if Range='Yes' then Writeln(' 3.  Feret Fraction :
',R2S(LowFrac,1,3),'-',R2S(UppFrac,1,3),' @ ',R2S(Gap,1,3))
         else Writeln(' 3.  Feret Fraction : ',R2S(fraction,1,3));
         end
    else Writeln(' 3.  Number of points : ',ripple);
    If (FileType='S') and (Method='P') and (PoreMeth='O') then Disabled(3);
    if (FileType='M') and (Method='P') and (PoreMeth='P') then Disabled(3);
    Writeln(' 4.  Confidence Interval : ',R2S(Convert/10,2,1)+'%');
    If (FileType='S') and (Method='P') and (PoreMeth='O') then Disabled(4);
    case Method of
    'S' : begin
          case Dimensions of
          2 : begin
              if (FileType='M') or ((FileType='S') and (SubType='I'))
              then begin
                 Write(' 5.  Direction (Left/Right) : ',WalkDir);
                 TextColor(LightGreen);
                 GotoXY(17,7); Write('L');
                 GotoXY(22,7); Writeln('R');
                 TextColor(Green);
                 end     {End 2-D Model 'F'}
              else Disabled(5); {End 2-D NOT 'F'}
            end;       {End 2 Dimensions}
          3 : begin
              if (FileType<>'M') then begin
                 Write(' 5.  Aspect (Front/Side/Top) : ',View);
                 TextColor(LightGreen);
                 GotoXY(14,7); Writeln('F');
                 GotoXY(20,7); Writeln('S');
                 GotoXY(25,7); Write('T');
                 TextColor(Green);
                 Writeln;
                 end
              else begin
                 Write(' 5.  Direction (Left/Right) : ',Walkdir);
                 TextColor(LightGreen);
                 GotoXY(17,7); Writeln('L');
                 GotoXY(22,7); Write('R');
                 TextColor(Green);
                 Writeln;
                 end;
              end;      {End 3 Dimensions}
          end;          {End 'dimensions' case}
          end;           {End Structured walk}
     'P' : begin
           if (Model='F') or (FileType='S') then begin
           Write(' 5.  Porosity Method : ');
           if model='F' then begin
           Case PoreMeth of
                'C' : Writeln('Cummulative');
                'V' : Writeln('Vertical Profile');
                'W' : Writeln('Wall Effect');
                'P' : Writeln('Pore Space Prop.');
                end;
           end     {End 2-D Model 'F'}
           else if FileType='S' then begin
           Case PoreMeth of
                'S' : Writeln('Selection Porosity');
                'P' : Writeln('Pore Space Prop.');
                'O' : Writeln('Overall Porosity');
                end;
           end;
     end        {End 'F' or 'S'}
          else Disabled(5); {End 2-D NOT 'F'}
     end;       {End Method=P}
     'E' : begin
                if FileType='S' then begin
                Write(' 5.  Density Method : ');
                Case EncMeth of
                     'O' : Writeln('Overall');
                     'B' : Writeln('Bounded');
                     end;                    {End case}
                end                        {End FileType='S'}
                else Disabled(5); {End 2-D NOT 'F'}
                end;
     'B' : begin
                Disabled(5); {End 2-D NOT 'F'}
                end
                else Disabled(5); {End 2-D NOT 'F'}
     end;       {End case Method}
     TextColor(Green+Blink);
     Write(' B');
     TextColor(Green);
     Writeln('.  Back to main menu');
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then begin
       Writeln; Writeln;
       Write('Enter number of option to change : ');
       option:=Readkey;
       ClrScr;
       Case option of
            '1' : begin
                  Window(6,16,70,24);
                  GotoXY(1,8);
                  Writeln('To measure all files in set, use ''*'' in place of
ID code ');
                  GetFile;
                  If (FileType='S') and (Method='P') and (PoreMeth='O') then
Disabled(3);
                  end;
            '2' : begin
                  Window(6,16,70,24);
                  Writeln('Please choose from : ');
                  Writeln;
```

```
                    Writeln('Structured Walk');
                    Writeln('Enclosing Circle');
                    Writeln('Radius of Gyration');
                    Writeln('Porosity');
                    Writeln('Box Counting');
                    TextColor(LightGreen);
                    GotoXY(1,3);Write('S');
                    GotoXY(1,4);Write('E');
                    GotoXY(1,5);Write('R');
                    GotoXY(1,6);Write('P');
                    GotoXY(1,7);Writeln('B');
                    TextColor(Green);
                    Window(16,3,62,11);
                    GotoXY(24,4);ClrEol;
                    option:=UpCase(ReadKey);
                    Case Option of
                         'S' : Method:='S';
                         'E' : Method:='E';
                         'R' : Method:='R';
                         'P' : Method:='P';
                         'B' : Method:='B';
                         end;
                    Writeln(Method);
                    end;

              '3' : begin
                    ClrScr;
                    if (Method='S') or (Method='B') then begin
                    Writeln('Please enter fraction - ''R'' for range (''D'' for
default range)');
                    Window(16,3,62,11);
                    {Reads character input}
                    GotoXY(23,5);ClrEol;option:=readkey;
case UpCase(Option) of

'R' : SetRange('A');
'D' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.08; Gap:=0.02; end;
'!' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.1; Gap:=0.018; end;
'"' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.12; Gap:=0.016; end;
'*' : begin Range:='Yes'; Uppfrac:=0.32; LowFrac:=0.08; Gap:=0.018; end;
'L' : begin Range:='Yes'; Uppfrac:=0.08; LowFrac:=0.02; Gap:=0.005; end

else begin
     if (Option=',') or (Option='/') then Option:='.';
        Write(Option);
        Readln(Response);
        {Add keypress to start of 'response'}
        Response:=Option+Response;
        Range:='No';
        Val(Response,Fraction,Code);
        end;
     end;
     end                {Ends Method='S'}
else begin              {Method <> 'S'}
         Window (16,3,62,11);
         GotoXY(25,5);ClrEol;readln(Ripple);
         end;
     end;         {Ends choice '3'}

'4' : begin
           Writeln('Possible values: 95, 98, 99, 99.5, 99.8, 99.9');
```

```
          Window(16,3,62,11);
          GotoXY(28,6);ClrEol;
          Readln(Conf);
          Convert:=round(10*Conf);
          Writeln(R2S(Convert/10,2,1)+'%');
          end;        {Ends choice '4'}
'5' : begin
          Window(16,3,62,11);
          Case method of
          'P' : begin
          if model='F' then begin
             Window(6,16,70,21);
             Writeln('Please choose from : ');
             Writeln;
             Writeln('Cummulative, Vertical Profile, Wall Effect');
             Writeln;
             Writeln('Pore Space Properties');
             TextColor(LightGreen);
             GotoXY(1,3);Write('C');
             GotoXY(14,3);Write('V');
             GotoXY(32,3);Writeln('W');
             GotoXY(12,5);Writeln('P');
             TextColor(Green);
             Window(16,3,62,11);
             GotoXY(24,7);ClrEol;
             end                   {End model 'F'}
          else if FileType='S' then begin
             Window(6,16,70,21);
             Writeln('Please choose from : ');
             Writeln;
             Writeln('Selection Porosity, Pore Space Props., Overall
Porosity');
             TextColor(LightGreen);
             GotoXY(1,3);Write('S');
             GotoXY(21,3);Writeln('P');
             GotoXY(40,3);Writeln('O');
             TextColor(Green);
             Window(16,3,62,11);
             GotoXY(24,7);ClrEol;
             end                   {End FileType 'S'}
          else GotoXY(31,7);
          end;              {End Method 'P'}
          'S' : begin
          if (Model='F') or ((FileType='S') and (SubType='I'))
          then GotoXY(31,7) else GotoXY(32,7);
          end;              {End Method 'S'}
          'E' : begin
             Window(6,16,70,21);
             Writeln('Please choose from : ');
             Writeln;
             Writeln('Overall Image, Bounded Image');
             TextColor(LightGreen);
             GotoXY(1,3);Write('O');
             GotoXY(16,3);Writeln('B');
             TextColor(Green);
             Window(16,3,62,11);
             GotoXY(23,7);ClrEol;
             end;
end;                        {End case}
          ClrEol;
          Option:=UpCase(ReadKey);
          Case Option of
               'F' : View:='F';
               'S' : View:='S';
               'T' : View:='T';
               'L' : WalkDir:='L';
               'R' : WalkDir:='R';
               'C' : PoreMeth:='C';
               'V' : PoreMeth:='V';
               'W' : PoreMeth:='W';
               'P' : PoreMeth:='P';
               'I' : PoreMeth:='I';
               'O' : EncMeth:='O';
               'B' : EncMeth:='B';
               end;
          if (Model='F') or ((FileType='S') and (SubType='I'))
          then writeln(WalkDir) else Writeln(View);
end;      {Ends choice '5'}
end
     end;
     until (Upcase(yesno)<>'Y') ;
     if Upcase(yesno)='B' then Return:=True;
     if Range='No' then begin
                    OldLow:=LowFrac; OldUpp:=UppFrac;
                    Used:=1;
                    LowFrac:=Fraction-Gap;
                    UppFrac:=Fraction;
                    end;
end;
{*************************************************************************}
procedure Mixed_Menu;

begin
     Repeat
     MaxDiffMoves:=round(Diff_Frac/5);
     MaxBalMoves:=20-MaxDiffMoves;
     MakeWindow(2);
     Writeln(' 1.  Diffusive Fraction : ',MaxDiffMoves*5,'%');
     TextColor(Green+Blink);
     Write(' B');
     TextColor(Green);
     Writeln('.  Back to main menu');
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then
     begin
          Writeln; Writeln;
          Write('Enter number of option to change : ');
          option:=Readkey;
          ClrScr;
               Case option of
                    '1' : begin
                         Writeln('Please enter Fraction as a percentage');
                         Window(16,3,62,11);
                         GotoXY(27,3);ClrEol;readln(Diff_Frac);
                         end;
               end
     end;
     until (Upcase(yesno)<>'Y') ;
     if Upcase(yesno)='B' then Return:=true;
end;
{*************************************************************************}
procedure Carlo_Menu;
```

```
begin
     Repeat
     MakeWindow(3);
     Writeln(' 1.  Downward Probabilty : ',DownProb,'%');
     Writeln(' 2.  Sticking Probabilty : ',StickProb,'%');
     TextColor(Green+Blink);
     Write(' B');
     TextColor(Green);
     Writeln('.  Back to main menu');
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then
     begin
          Writeln; Writeln;
          Write('Enter number of option to change : ');
          option:=Readkey;
          ClrScr;
                Case option of
                    '1' : begin
                          Writeln('Please enter Probabilty as a percentage');
                          Window(16,3,62,11);
                          GotoXY(28,3);ClrEol;readln(DownProb);
                          end;
                    '2' : begin
                          Writeln('Please enter Probabilty as a percentage');
                          Window(16,3,62,11);
                          GotoXY(28,4);ClrEol;readln(StickProb);
                          end;
                end
     end;
     until (Upcase(yesno)<>'Y') ;
     if Upcase(yesno)='B' then Return:=true;
end;
{**************************************************************************}
procedure Display_Menu;

var
   FileExists : boolean;

begin
     repeat
     MakeWindow(3);
     Writeln(' 1.  File : ',Shorten(Save));
     if multi then begin
        GotoXY(7+Length(Shorten(Save)),3);
        ClrEol;
        Writeln('('+I2S(Numfiles)+')');
        end;
     Writeln(' 2.  3D Revolve (0/1) : ',twist);
     TextColor(Green+Blink);
     Write(' B');
     TextColor(Green);
     Writeln('.  Back to main menu');
     Ask;
     yesno:=ReadKey; if UpCase(yesno)='Y' then
     begin
          Writeln; Writeln;
          Write('Enter number of option to change : ');
          option:=Readkey;
          ClrScr;
          Case option of
```

```
                '1' : GetFile;
                '2' : begin
                      Window(16,3,62,11);
                      GotoXY(25,4);ClrEol;
                      option:=UpCase(ReadKey);
                      Case Option of
                           '0' : Twist:='Off';
                           '1' : Twist:='On';
                           end;
                      Writeln(Twist);
                      end;
                end             {Ends Choices}
     end;
     until  (Upcase(yesno)<>'Y');
     if Upcase(yesno)='B' then Return:=True;
end;
{****************************************************************************}
end.
```

MOVER.PAS - Routine for movement of particles under seed agglomeration simulation.

```
unit mover;

{$N+,E+}

interface

uses Crt,Graph,funcs,screeninfo,variables;

var
   Xend,Yend,Zend : integer;
   Xbest,Ybest,Zbest,mindist : single;
   escape : boolean;
   ZRad : single;

procedure BallStep(D,radius : integer);
procedure Start;
procedure DiffStep(D,I,step : integer);
procedure MixStep(D,I,step : integer; OrigDir : single);
procedure Test2D(I,J : integer; model : char);
procedure Test3D(I,J : integer; model : char);
procedure Attach2D(I : integer);
procedure Attach3D(I : integer);

implementation

{**************************************************************************}
procedure BallStep(D, radius : integer);

begin
     Repeat
            AngleIn:=Random(360)/DegRad;
            AngleOut:=Random(360)/DegRad;
            if D=3 then begin
               RealRad:=radius;
               Zin:=Random(2*Radius);
               Slice_In:=sqrt(sqr(RealRad)-sqr(RealRad-Zin));
               Zout:=Random(2*Radius);
               Slice_Out:=sqrt(sqr(RealRad)-sqr(RealRad-Zout));
               end
            else begin
                 Slice_In:=Radius;
                 Slice_Out:=Radius;
                 end;
     until (AngleIn <> AngleOut) {or (Slice_In<>Slice_Out)};

     Xin:=X(AngleIn,Slice_In); Xout:=X(AngleOut,Slice_Out);
     Yin:=Y(AngleIn,Slice_In); Yout:=Y(AngleOut,Slice_Out);
end;
{**************************************************************************}
procedure Start;
{var
   ZRad : single;}
begin

     AngleIn:=Random(360)/DegRad;

     if Dimensions=3 then begin
```

```
        RealRad:=radius;
        Zstart:=(CentreZ-Radius)+Random(2*Radius);
        Zrad:=(CentreZ-Zstart);
        ZRad:=sqr(ZRad);
        Slice_In:=round(sqrt(sqr(RealRad)-ZRad));
     end
     else Slice_In:=Radius;

     Xstart:=X(AngleIn,Slice_In);
     Ystart:=Y(AngleIn,Slice_In);
end;
{***************************************************************************}
procedure diffstep(D,I,step : integer);

var
   Direction, Elev, Depth : single;
   Slice : single;
   angle:single;

begin
     repeat
     Direction:=Random(360)/DegRad;
     If D=3 then begin
        Angle:=Random(180);
        Elev:=(Angle-90)/DegRad;              {Angle of 'elevation'}
        Depth:=(step*Sin(Elev));                      {Depth change}
        Zend:=round(Zstart+Depth);
        Zdist:=Zend-CentreZ;
        Slice:=sqrt(sqr(step)-sqr(Depth));
        Xend:=round(Xstart + Slice * Cos(Direction));
        Yend:=round(Ystart + Slice * Sin(Direction));
        end
     else begin
         Xend:=round(Xstart + step * Cos(Direction));
         Yend:=round(Ystart + step * Sin(Direction));
         end;

     Xdist:=Xend-CentreX;
     Ydist:=Yend-CentreY;

     if D=2 then where:=sqrt(sqr(Xdist)+sqr(Ydist))
     else where:=sqrt(sqr(Xdist)+sqr(Ydist)+sqr(Zdist));

          if Debug='On' then begin
                       SetColor(Green);
                       Circle(Xstart,Ystart,Size[I]);
                       delay(Wait);
                       SetColor(Black);
                       Circle(Xstart,Ystart,Size[I]);
                       end;

     until where <= radius;

     Xin:=Xstart; Yin:=Ystart;         {Sets 'Test' parameters}
     Xout:=Xend; Yout:=Yend;
     if D=3 then begin Zin:=Zstart; Zout:=Zend; end;

     Xstart:=Xend; Ystart:=Yend;
     if D=3 then Zstart:=Zend;
end;
{***************************************************************************}
```

```
procedure mixstep(D,I,step : integer; OrigDir : single);

var
   Elev, Depth, Slice : single;

begin

      If D=3 then begin
         Elev:=((Random(180))-90)/DegRad;              {Angle of 'elevation'}
         Depth:=(step*Sin(Elev));                      {Depth change}
         Zend:=round(Zstart+Depth);
         Zdist:=Zend-CentreZ;
         Slice:=sqrt(sqr(step)-sqr(Depth));
         Xend:=round(Xstart + Slice * Cos(OrigDir));
         Yend:=round(Ystart + Slice * Sin(OrigDir));
         end
      else begin
          Xend:=round(Xstart + step * Cos(OrigDir));
          Yend:=round(Ystart + step * Sin(OrigDir));
          end;

      Xdist:=Xend-CentreX;Ydist:=Yend-CentreY;
      if D=2 then where:=sqrt(sqr(Xdist)+sqr(Ydist))
      else where:=sqrt(sqr(Xdist)+sqr(Ydist)+sqr(Zdist));

      Xin:=Xstart; Yin:=Ystart;          {Sets 'Test' parameters}
      Xout:=Xend; Yout:=Yend;
      if D=3 then begin Zin:=Zstart; Zout:=Zend; end;

     if Debug='On' then begin
                        SetColor(Green);
                        Circle(Xin,Yin,Size[I]);
                        delay(Wait);
                        SetColor(Black);
                        Circle(Xin,Yin,Size[I]);
                        end;

      Xstart:=Xend; Ystart:=Yend;       {Resets start point}
      if D=3 then Zstart:=Zend;
end;
{*************************************************************************}
procedure test2D(I,J : integer; model : char);
var
   combined : integer;

begin
      PDist(PosX[J],PosY[J],0);
      combined:=size[I]+size[J];
      if model='B' then hit[J]:=perpdist<=combined
      else hit[J]:=(perpdist<=combined) and ((b<=combined) or (c<=combined));
      if (flag=false) and (hit[J]=true) then flag:=true;
end;
{*************************************************************************}
procedure test3D(I,J : integer; model : char);
var
   combined : integer;

begin
      PDist(PosX[J],PosY[J],PosZ[J]);
      combined:=size[I]+size[J];
      if model='B' then hit[J]:=perpdist<=combined
     else hit[J]:=(perpdist<=combined) and ((b<=combined) or (c<=combined));
     if flag=false then if hit[J]=true then flag:=true;
end;
{********************PROCEDURE   A T T A C H   BELOW**********************}
procedure attach2D(I : integer);


var
   Combined : integer;
   Xlen, Ylen : integer;
   Xdir, Ydir : integer;
   X, Y : double;
   Xmove, Ymove : boolean;
   dist, old_dist, best : single;
   XFix, YFix, DistFix : single;
   fraction : single;

label
     wayout;

begin
     Xlen:=abs(Xin-Xout); Ylen:=abs(Yin-Yout);
     Xmove:= Xlen >= Ylen;
     Ymove:= Ylen > Xlen;

     if Xin >= Xout then Xdir := -1 else Xdir:=1;
     if Yin >= Yout then Ydir := -1 else Ydir:=1;

     Best:=2*radius;
     DistFix:=4*radius;

for J:=1 to I-1 do begin

if hit[J] then begin

   Mindist:=4*radius;
   escape:=false;
   Dist:=2*radius;
   Combined:=Size[I]+Size[J];
   X:=Xin; Y:=Yin;

if Xmove then begin

   SetColor(Xdir+4);
   repeat
         old_dist:=dist;
         Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J]));
         if dist<mindist then begin
         mindist:=dist;
         Xbest:=X; Ybest:=Y; end;
         X:= X + Xdir;
         escape:=(abs(CentreX-X)>radius) and (Model='B');
   until ((old_dist > Combined) and (dist <= Combined)) or escape;
   if escape then goto wayout;
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   X := X - (Xdir*(1+Fraction));
   Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
   end;                    {End Xmove}

if Ymove then begin
```

```
   SetColor(Ydir+3);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J]));
         if dist<mindist then begin
         mindist:=dist;
         Xbest:=X; Ybest:=Y; end;
         Y:= Y + Ydir;
         escape:=(abs(CentreY-Y)>radius) and (Model='B');
   until ((old_dist > Combined) and (dist <= Combined)) or escape;
   if escape then goto wayout;
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   Y := Y - (Ydir*(1+Fraction));
   X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
   end;                   {End Ymove}

   wayout:
   if escape then begin X:=Xbest; Y:=Ybest; end;
   XFix:=X-Xin; YFix:=Y-Yin;
   DistFix:=sqrt(sqr(XFix)+sqr(YFix));

   end;                   {End 'if Hit[J]... loop}

   If DistFix < best then begin
      Best:=DistFix;
      PosX[I]:=round(X);
      PosY[I]:=round(Y);
   end;

   end;                   {End for J... loop}

   Particle(PosX[I],PosY[I],Size[I],Red);

end;                   {End Attach - Do not remove - Leave at bottom}
{******************PROCEDURE   A T T A C H  3 D  BELOW********************}
procedure attach3D(I : integer);


var
   Xlen, Ylen, Zlen : integer;
   Xdir, Ydir, Zdir : integer;
   X, Y, Z : double;
   Xmove, Ymove, Zmove : boolean;
   combined : integer;
   dist, old_dist, best : single;
   XFix, YFix, ZFix, DistFix : single;
   fraction : single;

label wayout;

begin
     Xlen:=abs(Xin-Xout); Ylen:=abs(Yin-Yout); Zlen:=abs(Zin-Zout);
     Xmove:= (Xlen >= Ylen) and (Xlen >= Zlen);
     Ymove:= (Ylen > Xlen) and (Ylen >= Zlen);
     Zmove:= (Zlen > Xlen) and (Zlen > Ylen);

     if Xin >= Xout then Xdir := -1 else Xdir:=1;
     if Yin >= Yout then Ydir := -1 else Ydir:=1;
     if Zin >= Zout then Zdir := -1 else Zdir:=1;
     best:=2*radius;
     DistFix:=4*radius;

for J:=1 to I-1 do begin

if hit[J]  then begin

     Mindist:=4*radius;
     escape:=false;
     Dist:=2*radius;
     X:=Xin; Y:=Yin; Z:=Zin;
     Combined:=Size[I]+Size[J];

if Xmove then begin
   SetColor(Xdir+4);
   repeat
         old_dist:=dist;
         Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
         Z := Zin + Zdir * abs((X - Xin) * (ZLen/XLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J])+sqr(Z-PosZ[J]));
         if dist<mindist then begin
         mindist:=dist;
         Xbest:=X; Ybest:=Y; Zbest:=Z; end;
         X:= X + Xdir;
         escape:=(abs(CentreX-X)>radius) and (Model='B');
   until ((old_dist > Combined) and (dist <= Combined)) or escape;
   if escape then goto wayout;
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   X := X - (Xdir*(1+Fraction));
   Y := Yin + Ydir * abs((X - Xin) * (YLen/XLen));
   Z := Zin + Zdir * abs((X - Xin) * (ZLen/XLen));
   end;                         {End Xmove}

if Ymove then begin
   SetColor(Ydir+3);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
         Z := Zin + Zdir * abs((Y - Yin) * (ZLen/YLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J])+sqr(Z-PosZ[J]));
         if dist<mindist then begin
         mindist:=dist;
         Xbest:=X; Ybest:=Y; Zbest:=Z; end;
         Y:= Y + Ydir;
         escape:=(abs(CentreY-Y)>radius) and (Model='B');
   until ((old_dist > Combined) and (dist <= Combined)) or escape;
   if escape then goto wayout;
   Fraction:=(Combined-Dist)/(Old_dist-Dist);
   Y := Y - (Ydir*(1+Fraction));
   X := Xin + Xdir * abs((Y - Yin) * (XLen/YLen));
   Z := Zin + Zdir * abs((Y - Yin) * (ZLen/YLen));
   end;                         {End Ymove}

if Zmove then begin
   SetColor(Zdir+5);
   repeat
         old_dist:=dist;
         X := Xin + Xdir * abs((Z - Zin) * (XLen/ZLen));
         Y := Yin + Ydir * abs((Z - Zin) * (YLen/ZLen));
         dist:=sqrt(sqr(X-PosX[J])+sqr(Y-PosY[J])+sqr(Z-PosZ[J]));
```

```
            if dist<mindist then begin
            mindist:=dist;
            Xbest:=X; Ybest:=Y; Zbest:=Z; end;
            Z:= Z + Zdir;
            escape:=(abs(CentreZ-Z)>radius) and (Model='B');
      until ((old_dist > Combined) and (dist <= Combined)) or escape;
      if escape then goto wayout;
      Fraction:=(Combined-Dist)/(Old_dist-Dist);
      Z := Z - (Zdir*(1+Fraction));
      X := Xin + Xdir * abs((Z - Zin) * (XLen/ZLen));
      Y := Yin + Ydir * abs((Z - Zin) * (YLen/ZLen));
      end;                      {End Zmove}

      wayout:
      if escape then begin X:=Xbest; Y:=Ybest; Z:=Zbest; end;
      XFix:=X-Xin; ZFix:=Z-Zin; YFix:=Y-Yin;
      DistFix:=sqrt(sqr(XFix)+sqr(ZFix)+sqr(YFix));

      end;                  {End 'if Hit[J]... loop}

             If DistFix < best then begin
                best:=DistFix;
                PosX[I]:=round(X);
                PosY[I]:=round(Y);
                PosZ[I]:=round(Z);

              end;

          end;                      {End 'for J... loop}

              Particle(PosX[I],PosY[I],Size[I],Red);
   end;       {End Attach - Do not remove - Leave at bottom}

   end.
```

POROSITY.PAS - Contains procedures for calculating porosity of systems

```
unit porosity;

interface
{$N+,E+}
uses Crt,Graph, funcs, mover, screeninfo,variables;

var
   Distance : single;
   CircSize,SphereSize : single;
   PartArea,EncMin, EncMax, Increment : single;
   Mass, RadEnc : Array [1..50] of Single;
   PoreMin,PoreMax : single;


procedure pores;
procedure FiltPores;
procedure SEMpore;

implementation
{***************************************************************************}
procedure pores;

var
   I,J : integer;


begin
      CofG(number);
      EncMin:=radius; EncMax:=0;
      for I:=1 to number do begin
          Distance:=sqrt(sqr(PosX[I]-CofGX)+sqr(PosY[I]-CofGY)+sqr(PosZ[I]-
CofGZ))+Size[I];
          if Distance < EncMin then EncMin:=Distance;
          if Distance > EncMax then EncMax:=Distance;
      end;

      Increment:=(EncMax-EncMin)/Ripple;
      SetColor(LightBlue);

      for I:=1 to Ripple do begin
          Mass[I]:=0;
          RadEnc[I]:=EncMin+I*(Increment)+1;
          CircSize:=Pi*sqr(RadEnc[I]);
          SphereSize:=(4/3)*Pi*PowerFn(RadEnc[I],3);
          Circle(round(CofGX),round(CofGY),round(RadEnc[I]));
          if dimensions=2 then for J:=1 to number do begin
                 Distance:=sqrt(sqr(PosX[J]-CofGX)+sqr(PosY[J]-CofGY))+Size[J];
                 if Distance < RadEnc[I] then begin
                    PartArea:=Pi*sqr(Size[J]);
                    Mass[I]:=Mass[I]+PartArea;
                    if debug='On' then Particle(PosX[J],PosY[J],Size[J],Red);
                 end;
          end                                {End 2 dimensions}
          else for J:=1 to number do begin
                 Distance:=sqrt(sqr(PosX[J]-CofGX)+sqr(PosY[J]-
CofGY)+sqr(PosZ[J]-CofGZ))+Size[J];
                 if Distance < RadEnc[I] then begin
                    PartArea:=(4/3)*Pi*PowerFn(Size[J],3);
                    Mass[I]:=Mass[I]+PartArea;
```

```
                  if Debug='On' then Particle(PosX[J],PosY[J],Size[J],Red);
               end;
            end;                              {End 3 dimensions}
          if Dimensions=2 then Mass[I]:=(CircSize-Mass[I])/CircSize
          else Mass[I]:=(SphereSize-Mass[I])/SphereSize;
      if Debug='On' then delay(Wait);
      end;                                    {End 'Ripple' repeat}

      PoreMax:=0;PoreMin:=1;AvPore:=0;
      for I:=1 to Ripple do begin
      if Mass[I]>PoreMax then PoreMax:=Mass[I];
      if Mass[I]<PoreMin then PoreMin:=Mass[I];
      AvPore:=AvPore+Mass[I];
      end;
      AvPore:=AvPore/Ripple;

      if (not multi) or (Debug='On') then Continue;

      PoreMeth:='A';

      GraphIt(Ripple,0,'Porosity Measurement','Radius','Porosity',RadEnc,Mass);

      SetTextJustify(CenterText,TopText);
      OutTextXY(554,140,'Range = '+R2S(PoreMin,3,3)+' - '+R2S(PoreMax,3,3));
      SetColor(LightGreen);
      OutTextXY(554,35,Title);
      OutTextXY(554,155,chr($AF)+' P '+chr($F7)+' '+R2S(AvPore,3,3)+'
'+chr($AE));
      number:=number-1;             {Resets number for building}
      if (not multi) or (Debug='On') then Continue;
end;
{************************************************************************}
procedure FiltPores;

var
   I,J,K,X,Y,Weight,Height,Found : integer;
   Depth : array [1..50] of single;
   A1,A2,T, Area, Beta, Elev, Temp, Base,
   PrevDepth, CumMass, SqSize : single;
   Distance, deltaX, deltaZ : single;
   Vol,LastVol,LastSize : single;

begin
      SetTextJustify(leftText,TopText);
      ShowName;
      Height:=F1+1;
      for I:=1 to number do if (PosY[I]-Size[I]) < Height then
      Height:=(PosY[I]-Size[I])-1;

      SetColor(LightBlue);
      CumMass:=0; LastVol:=0; LastSize:=0;

      for I:=1 to Ripple do begin
      Mass[I]:=0;  Vol:=0;

      if PoreMeth='W' then begin
         if Dimensions=2 then begin
         Increment:=(RWall-LWall)/Ripple;
         Depth[I]:=LWall+I*(Increment);
         if I=1 then PrevDepth:=LWall else PrevDepth:=Depth[I-1];
         {Find Highest particle in strip}
```

```
        Y:=Height-5; found:=0;
        repeat
           Inc(Y);
           for X:=round(PrevDepth) to round(Depth[I]) do begin
               col:=GetPixel(X,Y); if (col=blue) or (col=green) then found:=1;
               end;
           until found=1;
        end                         {End 2 Dimensions}
        else begin
        Increment:=CellRad/Ripple;
        Depth[I]:=I*Increment;
        {Determine highest particle}
        Height:=Fl;
        for J:=1 to number do begin
            deltaX:=PosX[J]-CellCen; deltaZ:=PosZ[J]-CellCen;
            Distance:=sqrt(sqr(deltaX)+sqr(deltaZ))-Size[J];
            if Distance<=Depth[I] then
               if PosY[J]-Size[J]<Height then Height:=PosY[J]-Size[J];
            end;                     {End for J:=1 to number loop}
            SqSize:=(Fl-Height)*Pi*sqr(Depth[I]);
        end;                        {End 3 Dimensions}
        end                         {End PoreMeth='W'}
     else begin
        Increment:=(Fl-Height)/Ripple;
        Depth[I]:=Fl-I*(Increment);
        if I=1 then PrevDepth:=Fl else PrevDepth:=Depth[I-1];
        end;

     for J:=1 to number do begin
        if dimensions=2 then PartArea:=Pi*sqr(Size[J])
           else PartArea:=4/3*Pi*Powerfn(Size[J],3);

     case PoreMeth of

     'V','C' : begin
               if PosY[J]+Size[J]>=Depth[I]-1 then begin
               case Dimensions of
               2 : begin
                     if PoreMeth='C' then SqSize:=(Rwall-LWall)*(Fl-Depth[I])
                     else SqSize:=(Rwall-LWall)*(PrevDepth-Depth[I]);
                     end;
               3 : begin
                     if PoreMeth='C' then SqSize:=Pi*sqr((Rwall-LWall)/2)*(Fl-
Depth[I])
                     else SqSize:=Pi*sqr((Rwall-LWall)/2)*(PrevDepth-
Depth[I]);
                     end;
               end;       {End case dimensions}
               {If ANY PART of particle is within lines}
               if (PosY[J]+Size[J] >= Depth[I]) and (PosY[J]-Size[J] <=
PrevDepth) then begin

               A1:=0; A2:=0;
               Temp:=Size[J];

               {Case 1 - Centre of particle in area}
               if (PosY[J]<PrevDepth) and (PosY[J]>Depth[I]) then begin

               {Case 1a - Bottom Edge of particle outside area}
               if (PosY[J]+Size[J])>PrevDepth then begin
               {Measure area OUTSIDE boundary}
               Elev:=PrevDepth-PosY[J];
               Base:=sqrt(sqr(Temp)-sqr(Elev));
               T:=Elev*Base;
               Beta:=arctan(Base/Elev);
               if Dimensions=2 then A1:=Beta*sqr(Temp)-T
                  else A1:=(Beta*Powerfn(Temp,3)*2-Pi*sqr(Temp)*Elev)/3
               end;

               {Case 1b - Top Edge of particle outside area}
               if PosY[J]-Size[J]<Depth[I] then begin
               {Measure area OUTSIDE boundary}
               Elev:=PosY[J]-Depth[I];
               Base:=sqrt(sqr(Temp)-sqr(Elev));
               T:=Elev*Base;
               Beta:=arctan(Base/Elev);
               if Dimensions=2 then A2:=Beta*sqr(Temp)-T
                  else A2:=(Beta*Powerfn(Temp,3)*2-Pi*sqr(Base)*Elev)/3
               end;
               Area:=PartArea-A1-A2;
               end;                     {End centre in area}

               {Case 1c - Exactly half of particle in area}
               if (PosY[J]=PrevDepth) or (PosY[J]=Depth[I]) then
Area:=PartArea/2;

               {Case 2 - Top Edge of particle in area}
               if (PosY[J]-Size[J]<PrevDepth) and (PosY[J]>PrevDepth) then
begin
               {Measure area INSIDE boundary}
               Elev:=PosY[J]-PrevDepth;
               Base:=sqrt(sqr(Temp)-sqr(Elev));
               T:=Elev*Base;
               Beta:=arctan(Base/Elev);
               if Dimensions=2 then Area:=Beta*sqr(Temp)-T
                  else Area:=(Beta*Powerfn(Temp,3)*2-Pi*sqr(Base)*Elev)/3
               end;

               {Case 3 - Bottom Edge of particle in area}
               if (PosY[J]+Size[J]>Depth[I]) and (PosY[J]<Depth[I]) then begin
               {Measure area INSIDE boundary}
               Elev:=Depth[I]-PosY[J];
               Base:=sqrt(sqr(Temp)-sqr(Elev));
               T:=Elev*Base;
               Beta:=arctan(Base/Elev);
               if Dimensions=2 then Area:=Beta*sqr(Temp)-T
                  else Area:=(Beta*Powerfn(Temp,3)*2-Pi*sqr(Base)*Elev)/3
               end;

               if PoreMeth='C' then begin
                  CumMass:=CumMass+Area;
                  Vol:=CumMass;
                  end
               else Vol:=Vol+Area;

               if debug='On' then Particle(PosX[J],PosY[J],Size[J],Red);
               Rectangle(LWall,round(PrevDepth),Rwall,round(Depth[I]));
               end;
               end;          {End particle below boundary}
               end;

          'W' : begin
```

```
                    if Dimensions=2 then begin
                    SqSize:=(Depth[I]-PrevDepth)*(F1-Y);

                    {If ANY PART of particle is within lines}
                    if (PosX[J]-Size[J] <= Depth[I]) and (PosX[J]+Size[J] >=
PrevDepth) then begin

                    A1:=0; A2:=0;

                    {Case 1 - Centre of particle in area}
                    if (PosX[J]>PrevDepth) and (PosX[J]<Depth[I]) then begin

                    {Case 1a - LH Edge of particle outside area}
                    if (PosX[J]-Size[J])<PrevDepth then begin
                    Elev:=PosX[J]-PrevDepth;
                    Temp:=Size[J]; Base:=sqrt(sqr(Temp)-sqr(Elev));
                    T:=Elev*Base;
                    Beta:=arctan(Base/Elev);
                    A1:=(Beta*sqr(Temp)-T); {Area outside}
                    end;                     {End Case 1a}

                    {Case 1b - RH Edge of particle outside area}
                    if PosX[J]+Size[J]>Depth[I] then begin
                    Elev:=Depth[I]-PosX[J];
                    Temp:=Size[J]; Base:=sqrt(sqr(Temp)-sqr(Elev));
                    T:=Elev*Base;
                    Beta:=arctan(Base/Elev);
                    A2:=(Beta*sqr(Temp)-T);
                    end;                     {End Case 1b}
                    Area:=PartArea-A1-A2;
                    end;                     {End case 1}

                    {Case 1c - Exactly half of particle in area}
                    if (PosX[J]=PrevDepth) or (PosX[J]=Depth[I]) then
Area:=PartArea/2;

                    {Case 2 - RH Edge of particle in area}
                    if (PosX[J]+Size[J]>PrevDepth) and (PosX[J]<PrevDepth) then
begin
                    Elev:=PrevDepth-PosX[J];
                    Temp:=Size[J]; Base:=sqrt(sqr(Temp)-sqr(Elev));
                    T:=Elev*Base;
                    Beta:=arctan(Base/Elev);
                    Area:=(Beta*sqr(Temp)-T);
                    end;                     {End Case 2}

                    {Case 3 - LH Edge of particle in area}
                    if (PosX[J]-Size[J]<Depth[I]) and (PosX[J]>Depth[I]) then begin
                    Elev:=PosX[J]-Depth[I];
                    Temp:=Size[J]; Base:=sqrt(sqr(Temp)-sqr(Elev));
                    T:=Elev*Base;
                    Beta:=arctan(Base/Elev);
                    Area:=(Beta*sqr(Temp)-T);
                    end;                     {End Case 3}
                    Vol:=Vol+Area;

                    if Debug='On' then Particle(PosX[J],PosY[J],Size[J],Red);

                    end;

                    end                      {End 2 Dimensions}
```

```
              else begin
                 deltaX:=PosX[J]-CellCen; deltaZ:=PosZ[J]-CellCen;
                 Distance:=sqrt(sqr(deltaX)+sqr(deltaZ));
                 if (Distance-Size[J])<=Depth[I] then begin
                 {Some or all of particle inside cylinder}
                    if Distance+Size[J]>Depth[I] then
Area:=CylPore(Depth[I],Distance,PosX[J],PosZ[J],Size[J],J)
                    {Only part of particle is within cylinder}
                    else Area:=PartArea;
                    {All of particle is within cylinder}
                    Vol:=Vol+Area;
                    if Debug='On' then Particle(PosX[J],PosZ[J],Size[J],Red);
                 end;                          {End Distance <=Depth[I]}
              end;                            {End 3 Dimensions}

              end;

          end; {End Case}

          end;        {End for J:=1 to number}

          if PoreMeth='W' then
          if Dimensions=3 then begin
             Vol:=Vol+Area-LastVol; SqSize:=SqSize-LastSize;
             LastVol:=Vol; LastSize:=SqSize;
             Circle(CellCen,CellCen,round(Depth[I]));
             end
          else Rectangle(round(PrevDepth),Fl,round(Depth[I]),Y);

          Mass[I]:=(SqSize-Vol)/SqSize;

      if Debug='On' then delay(Wait);
      end;                                   {End 'Ripple' repeat}
      PoreMax:=0; PoreMin:=1; AvPore:=0; Weight:=0;
      for I:=1 to Ripple do begin
          if PoreMeth='W' then begin
          if Dimensions=2 then Depth[I]:=Depth[I]-LWall-Increment/2 end
          else Depth[I]:=Fl-Depth[I]-Increment/2;
          {'/2' gives midpoint of sample}
          if Mass[I]>PoreMax then PoreMax:=Mass[I];
          if Mass[I]<PoreMin then PoreMin:=Mass[I];
          {Weights porosity average to centre of structure (4:1 weighting)}
          if ((I<Ripple*0.2) or (I>Ripple*0.8)) or (PoreMeth='W') then begin
             AvPore:=AvPore+Mass[I];
             Inc(Weight);
             end
      else begin
          AvPore:=AvPore+(Mass[I]*4);
          Weight:=Weight+4;
          end;
      end;        {End weighting}

      AvPore:=AvPore / Weight;

      if (not multi) or (Debug='On') then Continue;

      Case PoreMeth of
      'W' : GraphIt(Ripple,0,'Horizontal Porosity Profile','Position across
cell','Porosity'
                                                                  ,Depth,Mass);
```

```
     'V' : GraphIt(Ripple,0,'Vertical Porosity Profile','Porosity','Height
above base'
                                                                  ,Mass,Depth);
     'C' : GraphIt(Ripple,0,'Cummulative Porosity
Measurement','Porosity','Height above base'
                                                                  ,Mass,Depth);
     end;
     SetTextJustify(CenterText,TopText);
     OutTextXY(554,140,'Range = '+R2S(PoreMin,3,3)+' - '+R2S(PoreMax,3,3));
     SetColor(LightGreen);
     OutTextXY(554,35,Title);
     OutTextXY(554,155,chr($AF)+' P '+chr($F7)+' '+R2S(AvPore,3,3)+'
'+chr($AE));
     if (not multi) or (Debug='On') then Continue;
end;
{**************************************************************************}
procedure SEMpore;

var
   I, Pix, Code : integer;
   Pixels, Stuff : longint;
   VGo,VStop,HGo,HStop :integer;
   Fig : char;

begin

If PoreMeth='S' then begin               {'Selection Porosity' Method}
     EncMin:=5; EncMax:=MaxRad;

     Increment:=(EncMax-EncMin)/Ripple;
     SetColor(LightBlue);

     for I:=1 to Ripple do begin
         Mass[I]:=0;
         RadEnc[I]:=EncMin+I*(Increment);
         CircSize:=Pi*sqr(RadEnc[I]);
         Circle(round(CofGX)+ExtraX,round(CofGY)+ExtraY,round(RadEnc[I]));
         VGo:=round(CofGY-RadEnc[I]);
         VStop:=round(CofGY+RadEnc[I]);
         HGo:=round(CofGX-RadEnc[I]-1);
         HStop:=round(CofGX+RadEnc[I]+1);

         Assign(Storage,save);
         Reset(Storage);

         for J:=1 to VGo-1 do Readln(Storage);       {Skips outside RadEnc}

         for J:=VGo to VStop do begin

         for K:=1 to HGo-1 do Read(Storage,Fig);     {Skips outside RadEnc}
         for K:=HGo to HStop do begin
                 if (K>LB) and (GetPix=1) then begin
                    Distance:=sqrt(sqr(K-CofGX)+sqr(J-CofGY));
                    if (distance < RadEnc[I]) then begin
                    Mass[I]:=Mass[I]+1;
                    if debug='On' then PutPixel(K+ExtraX,J+ExtraY,Red);
                    end;                             {Ends Inside Circle}
                end;                                 {Ends found pixel}
         end;                                        {Ends left to right}
     Readln(Storage);
     end;                                {End top to bottom}
     Close(Storage);

     Mass[I]:=(CircSize-Mass[I])/(CircSize);
     if Debug='On' then delay(Wait);
     end;                                 {End 'Ripple' repeat}
     PoreMax:=0;PoreMin:=1;AvPore:=0;
     for I:=1 to Ripple do begin
     RadEnc[I]:=(RadEnc[I]); Mass[I]:=(Mass[I]);
     AvPore:=AvPore+Mass[I];
     if Mass[I]>PoreMax then PoreMax:=Mass[I];
     if Mass[I]<PoreMin then PoreMin:=Mass[I];
     end;
     AvPore:=AvPore / Ripple;

     if (not multi) or (Debug='On') then Continue;

     GraphIt(Ripple,0,'Porosity Measurement','Radius','Porosity',RadEnc,Mass);

     SetTextJustify(CenterText,TopText);
     OutTextXY(554,140,'Range = '+R2S(PoreMin,3,3)+' - '+R2S(PoreMax,3,3));
     SetColor(LightGreen);
     OutTextXY(554,35,Title);
     OutTextXY(554,155,chr($AF)+' P '+chr($F7)+' '+R2S(AvPore,3,3)+'
'+chr($AE));
     if multi then Writeln(DataFile,R2S(AvPore,1,3)+' '+R2S((1-
abs(r))*100,3,2));
     end                          {End 'Selection Porosity' method}

else begin                        {Overall Porosity}

     Pixels:=0; Stuff:=0;
     Assign(Storage,save);
     Reset(Storage);
     for J:=1 to TB do Readln(Storage);          {Miss Top Border}
     for J:=TB to (HImage-BB) do begin           {Reads until BB}
         for K:=1 to LB do read(Storage,Fig);    {Miss Left Border}
         for K:=LB+1 to (WImage-RB) do begin     {Reads unitl RB}
             Pix:=GetPix;
             if Pix=1 then PutPixel(K+ExtraX,J+ExtraY,LightBlue);
             if Pix=0 then Stuff:=Stuff+1;
             Pixels:=Pixels+1;
         end;
         Readln(Storage);
         end;
         Close(Storage);
         AvPore:=Stuff/Pixels;
         SetColor(LightGreen);
         SetTextJustify(LeftText,TopText);
         OutTextXY(50,15,'Using -ve Space:  Porosity = '
         +R2S(AvPore,6,4)+' ('+R2S(AvPore*100,0,2)+' %)');
         OutTextXY(50,28,'Using +ve Space:  Porosity = '
         +R2S(1-AvPore,6,4)+' ('+R2S(100-AvPore*100,0,2)+' %)');
     end;
     if (not multi) or (Debug='On') then Continue;
end;
{*************************************************************************}
end.
```

REVOLVE.PAS - Unit containing mathematics for revolving three dimensional agglomerates

```
unit revolve;

{$N+,E+}

interface

uses Crt,Dos,Graph,disk,funcs,mover,screeninfo,variables;

procedure spin_menu(X,Y,Z,Size : info; Number,MidX,MidY : integer);

implementation

var
   angle, newangle : single;
   Xnew, Ynew, Znew : info;
   DeltaX, DeltaY, DeltaZ : integer;
   Length : single;

{***************************************************************************}
procedure Horiz(X,Y,Z : array of integer;
          Dir, Scale, Number : integer);

          {Moves in X & Z planes, sorts w.r.t. Z}
          {NB. f[I]¯f[I-1] to realign arrays (0.. cf. 1..)}
var
   I : integer;

begin
     for I:=1 to number do begin
         DeltaX:=X[I-1]-240;
         DeltaZ:=Z[I-1]-240;
         if DeltaX>0 then angle:=arctan(DeltaZ/DeltaX);
         if DeltaX<0 then angle:=arctan(DeltaZ/DeltaX)+Pi;
         if DeltaX=0 then begin
            if DeltaZ >=0 then angle:=-Pi/2
            else angle:=Pi/2;
         end;

         newangle:=angle+(Scale*Dir/DegRad);
         if newangle<0 then newangle:=newangle+2*Pi;
         if newangle>(2*Pi) then newangle:=newangle-2*Pi;

         length:=sqrt(sqr(DeltaX)+sqr(DeltaZ));
         Xnew[I]:=round(length*Cos(newangle)+240);
         Ynew[I]:=Y[I-1];
         Znew[I]:=round(length*Sin(newangle)+240);
     end;                                     {For 0 = 1 to ...}
end;
{***************************************************************************}
procedure Vert(X,Y,Z : array of integer;
          Dir, Scale, Number : integer);

          {Moves in Y & Z planes, sorts w.r.t. Z}
          {NB. f[I]¯f[I-1] to realign arrays (0.. cf. 1..)}
var
   I : integer;

begin
```

```
     for I:=1 to number do begin
         DeltaY:=Y[I-1]-240;
         DeltaZ:=Z[I-1]-240;
         if DeltaY>0 then angle:=arctan(DeltaZ/DeltaY);
         if DeltaY<0 then angle:=arctan(DeltaZ/DeltaY)+Pi;
         if DeltaY=0 then begin
            if DeltaZ >=0 then angle:=-Pi/2
            else angle:=Pi/2;
         end;

         newangle:=angle+(Scale*Dir/DegRad);
         if newangle<0 then newangle:=newangle+2*Pi;
         if newangle>(2*Pi) then newangle:=newangle-2*Pi;

         length:=sqrt(sqr(DeltaY)+sqr(DeltaZ));
         Ynew[I]:=round(length*Cos(newangle)+240);
         Xnew[I]:=X[I-1];
         Znew[I]:=round(length*Sin(newangle)+240);
     end;                                  {For 0 = 1 to ...}
end;

{**************************************************************************}
procedure spin_menu(X,Y,Z,Size : info; Number, MidX, MidY : integer);

var
   Xorig, Yorig, Zorig, Sorig : info;
   Xdist, Ydist, Maxdist : longint;
   I,J,Col : integer;
   Distance : single;
   Option : char;
   Flag1, Flag2, Flag3, Flag4 : boolean;
   Xcam, Ycam, Xbulb, Ybulb : integer;
   NumLock : boolean;
   Regs : registers;

begin
     Flag1:=False; Flag2:=False; Flag3:=False; Flag4:=False;
     Xorig:=X; Yorig:=Y;Zorig:=Z;Sorig:=Size;    {Remember original positions}
     Xnew:=X; Ynew:=Y; Znew:=Z;
     Col:=Blue;

     RevolveScr;                             {SCREEN INFOrmation}

     repeat
           Regs.AH:=2;
           Intr($16, Regs);
           NumLock:=(Regs.AL and $20)>0;     {Check NumLock state}
           SetColor(LightGreen);
           SetTextJustify(LeftText,TopText);
           SetColor(Red);
           OutTextXY(540,460,'with NumLock OFF');
           while KeyPressed do Option:=Readkey;
     until Numlock=False;

     DelText(540,460,'with NumLock OFF');
     SetColor(Green);

     repeat
     Option:=ReadKey;

     if Option=#0 then begin
   Option:=ReadKey;
   J:=ord(Option);
   case J of
        60 : Flag1:=True;                      {F2}
        61 : Flag3:=True;                      {F3}
        62 : Flag4:=True;                      {F4}
        71 : Flag2:=True;                      {HOME}
        75 : Horiz(X,Y,Z,1,1,Number);          {Left}
        77 : Horiz(X,Y,Z,-1,1,Number);         {Right}
        72 : Vert(X,Y,Z,1,1,Number);           {Up}
        80 : Vert(X,Y,Z,-1,1,Number);          {Down}
   end;
   end             {Ends special key catch}
else
    begin
         J:=ord(Option);
         case J of
              52 : Horiz(X,Y,Z,1,45,Number);        {Shift + Left}
              54 : Horiz(X,Y,Z,-1,45,Number);       {Shift + Right}
              56 : Vert(X,Y,Z,1,45,Number);         {Shift + Up}
              50 : Vert(X,Y,Z,-1,45,Number);        {Shift + Down}
             104 : Halt;                            {'h'}
         end;
    end;           {ENDS other key catch}

if FileType='N' then begin
maxdist:=0;
for I:= 1 to number do
    begin
         Xdist:=(X[I]-MidX); YDist:=(Y[I]-MidY);
         Distance:=Sqrt(sqr(Xdist)+sqr(Ydist)) + Size[I];
         If distance > maxdist then begin
            maxdist:=round(distance)+2;
         end;
    end;

SetColor(Black);
SetFillStyle(SolidFill,Black);
FillEllipse(MidX,MidY,Maxdist,Maxdist);
Crosshairs(MidX,MidY);
end
else begin
SetColor(Black);
SetFillStyle(SolidFill,Black);
ClearSpace(26,40,424,470);
end;

if Flag1=True then begin                      {Store .dat file}
   Store3D(X,Y,Z,Size);
   Flag1:=False;
end

else if Flag2=True then begin                 {HOME position}
   Sound(1000);Delay(10);NoSound;
   X:=Xorig; Y:=Yorig; Z:=Zorig; Size:=Sorig;
   Flag2:=False;
   IntSort(0, Number-1, Z, X, Y, Size);
   Col:=Red;
end

else begin X:=Xnew; Y:=Ynew; Z:=Znew;
```

```
               IntSort(0, Number-1, Z, X, Y, Size);
               Col:=Blue;
          end;

     For I:=1 to Number do Particle(X[I],Y[I],Size[I],Col);

     if Flag3=True then begin              {Store .pov file}
        POVRay;
        Flag3:=False;
     end;


     if flag4=True then begin
        POVAnim;
        Flag4:=False;
     end;

     until J=27;
end;
{***************************************************************************}
end.
```

ROLLING.PAS – Routine for rolling three-dimension particle in filtration simulations

```
unit rolling;

{$N+,E+}

{*** Angles used for particle rolling motion :
Alpha   - angle of elevation of moving particle from horizontal
Gamma   - angle of Rolling particle relative to primary target
Delta   - angle of secondary target relative to primary target
Epsilon - angle of rolling particle relative to vertical (x-z plane)
          after rolling particle has contacted secondary target
Thi     - angle between line connecting rolling particle and primary
          target and primary and secondary targets

Note use of boolean operators :   ord(Boolean) gives True=1, False=0
                                                                   ***}

interface

uses Crt,Graph,funcs,screeninfo,variables;

var
   P : Array [2..4] of Integer;
   XRoll,ZRoll : integer;

procedure Roll3D(Q : integer);

implementation

const M=0.005;

var
   DeltaX,DeltaY,DeltaZ : integer;
   Alpha,Gamma,Delta,Epsilon,Thi : single;
   Dist,R,X,Y,Z,CenX,CenY,CenZ,IQ,QC : single;
   XMove,ZMove : boolean;
   Move : integer;

procedure Direction(Move,Q:integer);
begin
     Case Move of

          0 : condition:='n';
          1 : begin
              {Movement in z-plane only}
              if PosZ[I]>PosZ[Q] then ZRoll:=1                {Away}
              else ZRoll:=-1;                                 {Towards}
              XRoll:=0;
              end; {Move=1}
          2 : begin
              {Movement in x-plane only}
              if PosX[I]>PosX[Q] then XRoll:=1                {Clockwise}
              else XRoll:=-1;                                 {Anti-Clockwise}
              ZRoll:=0;
              end; {Move=2}
          3 : begin
              {Movement in x- and z-planes}
              if PosZ[I]>PosZ[Q] then ZRoll:=1                {Away}
              else ZRoll:=-1;                                 {Towards}
              if PosX[I]>PosX[Q] then XRoll:=1                {Clockwise}
              else XRoll:=-1;                                 {Anti-Clockwise}
```

```
            end {Move=3}
        end; {Case}
end;
{***************************************************************************}
procedure RollTest3D(X,Y,Z : single; Q : integer);

var
   K,Best,d : integer;
   Distance,BestFix : single;

begin
     Touch:=1;
     BestFix:=2*CellRad;
     Rest:=False;

     Distance:=sqrt(sqr(X-CellCen)+sqr(Z-CellCen));
     {Calculates distance of particle from CENtre of CELL}

     if Distance>=CellRad then begin
        Rest:=True;
        Condition:='w';          {'b' for bottom}
        end;                     {end rest}

     if (Y+Size[I])>=Fl then begin
        Rest:=True;
        Condition:='b';          {'b' for bottom}
        end;                      {end rest}

     if Rest=True then exit;

     for K:=I-1 downto 1 do begin
          d:=Size[I]+Size[K];
          if (K<>Q) then begin
           XDist:=X-PosX[K]; YDist:=Y-PosY[K]; ZDist:=Z-PosZ[K];
           Distance:=sqrt(sqr(XDist)+sqr(YDist)+sqr(ZDist));
           if (Distance<=d) then begin
              Condition:='p';
              Touch:=Touch+1;
              If Touch>2 then begin
                 Rest:=True;
                 condition:='n';
                 Exit
                 end;
              If Distance<BestFix then begin
                 BestFix:=Distance;
                 P[2]:=K;
                 end;
              if Debug='On' then delay(wait);
              end;                               {End distance<d}
           end;                                  {End K<>Q & K<>Roll1}
     end;                                        {End for K:=1 to I-1 (not
rest)}
end;                                             {End procedure}
{***************************************************************************}
procedure Roll1(Q:integer);

begin
{Define Angles for Falling / Primary Particle (Gamma)}
     XDist:=PosX[Q]-PosX[I];
     XMove:=XDist<>0;
     YDist:=PosY[Q]-PosY[I];
```

```
      ZDist:=PosZ[Q]-PosZ[I];
      ZMove:=ZDist<>0;
      R:=Size[I]+Size[Q];
      Move:=ord(ZMove)+2*ord(XMove);

      if Move=0 then begin
            condition:='n';
            Rest:=true;
            exit;
            end;

      Direction(Move,Q);

      Gamma:=AngFind(XDist,ZDist);

      Dist:=sqrt(sqr(XDist)+sqr(ZDist));
      if Dist=0 then Alpha:=0 else Alpha:=arctan(YDist/Dist);

      if Debug='On' then begin
         Particle(PosX[Q],PosY[Q],Size[Q],Red);
         SetColor(Red);
         OutTextXY(461,226,'I   :
   '+I2S(PosX[I])+','+I2S(PosY[I])+','+I2S(PosZ[I]));
         SetColor(Green);
         Outtextxy(460,202,'Q   :
   '+I2S(PosX[Q])+','+I2S(PosY[Q])+','+I2S(PosZ[Q]));
         OutTextXY(460,262,chr($E0));                {Alpha}
         SetTextJustify(RightText,TopText);
         end;

      Rest:=False;
      WasTouching:=1;

      repeat
            Alpha:=Alpha-M;
            X:=PosX[Q]-R*Cos(Alpha)*Sin(Gamma);
            Z:=PosZ[Q]-R*Cos(Alpha)*Cos(Gamma);
            Y:=PosY[Q]-R*Sin(Alpha);

            if Debug='On' then begin
               OutTextXY(599,170,R2S(X,3,0)+','+R2S(Y,3,0)+','+R2S(Z,3,0));
               OutTextXY(540,262,R2S(Alpha*DegRad,9,4)+chr($F8));
               PutPixel(round(X),round(Y),LightGreen+ord(XRoll));
               Cross(PosX[Q],PosY[Q]);
               Delay(wait);
               DelText(599,170,R2S(X,3,0)+','+R2S(Y,3,0)+','+R2S(Z,3,0));
               DelText(540,262,R2S(Alpha*DegRad,9,4)+chr($F8));
               end;

            RollTest3D(X,Y,Z,Q);
            until Rest or (Alpha <= 0) or (Touch>1);

      if (Alpha <= 0) then condition:='f';

      if Debug='On' then begin
         SetTextJustify(LeftText,TopText);
         DelText(461,226,'I   :
   '+I2S(PosX[I])+','+I2S(PosY[I])+','+I2S(PosZ[I]));
         DelText(460,202,'Q   :
   '+I2S(PosX[Q])+','+I2S(PosY[Q])+','+I2S(PosZ[Q]));
         DelText(460,262,chr($E0));                {Alpha}
      end;

   if Condition='w' then begin
      {Backs off particle in order to remain within cell boundaries}
      Dist:=sqrt(sqr(X-CellCen)+sqr(Z-CellCen));
      if Dist>CellRad then begin
         Alpha:=Alpha+M/10;
         X:=PosX[Q]-R*Cos(Alpha)*Sin(Gamma);
         Z:=PosZ[Q]-R*Cos(Alpha)*Cos(Gamma);
         Y:=PosY[Q]-R*Sin(Alpha);
         end;
      end;
end;
{***************************************************************************}
procedure Roll2(Q:integer);

var
absx,absz:integer;
RollAng,Ratio : single;

label
   landed;

begin
{Define Angles for Falling / Primary Particle (Epsilon)}
   XDist:=PosX[Q]-X;
   ZDist:=PosZ[Q]-Z;
   Epsilon:=AngFind(XDist,ZDist);

   XMove:=XDist<>0;
   ZMove:=ZDist<>0;
   Move:=ord(ZMove)+2*ord(XMove);
   Direction(Move,Q);

{Define Angles for Secondary / Primary Particle (Delta)}
   DeltaX:=PosX[Q]-PosX[P[2]];
   DeltaZ:=PosZ[Q]-PosZ[P[2]];

   Delta:=AngFind(DeltaX,DeltaZ);
   if DeltaZ=0 then begin
      if PosX[P[2]]>PosX[Q] then Delta:=3*Pi/2 else Delta:=Pi/2;
      end;

   IQ:=sqrt(sqr(XDist)+sqr(ZDist));
   Thi:=Epsilon-Delta;

   If IQ=0 then begin
      rest:=true;
      condition:='n';
      goto landed;
      end;

   ZMove:=DeltaZ<>0;
   If ZMove then QC:=Cos(Thi)*IQ else QC:=IQ;

   Ratio:=QC/IQ;

   CenX:=PosX[Q]-QC*Sin(Delta);                      {Centre of rotation, x}
   CenY:=PosY[Q]-(Ratio*(PosY[Q]-PosY[P[2]]));       {Centre of rotation, y}
   CenZ:=PosZ[Q]-QC*Cos(Delta);                      {Centre of rotation, z}
```

```
    XDist:=X-CenX; YDist:=Y-CenY; ZDist:=Z-CenZ;
    R:=Sqrt(sqr(XDist)+sqr(YDist)+sqr(ZDist));      {Radius of rotation}
    RollAng:=AngFind(XDist,ZDist);

    Dist:=sqrt(sqr(XDist)+sqr(ZDist));
    if Dist=0 then Alpha:=0 else Alpha:=arctan(YDist/Dist);

    if Alpha=0 then begin
       Rest:=True;
       Condition:='n';
       Exit;
       end;

    if Delta>=Pi then Delta:=Delta-Pi;
    if (Epsilon<=Delta) or (Epsilon-Delta>Pi) then RollAng:=delta-Pi/2
    else RollAng:=delta+Pi/2;
    If RollAng>2*Pi then RollAng:=RollAng-2*Pi;
    If RollAng<0 then RollAng:=RollAng+2*Pi;

    If Debug='On' then begin
       if PosX[Q]>PosX[P[2]] then
          absx:=(PosX[P[2]]-Size[P[2]]-500) else absx:=(PosX[Q]-Size[Q]-500);
       if PosZ[Q]>PosZ[P[2]] then
          absz:=(PosZ[P[2]]-Size[P[2]]-30) else absz:=(Posz[Q]-Size[Q]-30);
       SetColor(Red);
       OutTextXY(461,226,'I  : '+R2S(X,3,1)+','+R2S(Y,3,1)+','+R2S(Z,3,1));
       Circle(round(x)-absx,480-(round(z)-absz),size[i]);
       SetColor(yellow);
       OutTextXY(462,214,'P'+Chr($FD)+' ('+I2S(P[2])+') : '
       +I2S(PosX[P[2]])+','+I2S(PosY[P[2]])+','+I2S(PosZ[P[2]]));
       Particle(posx[p[2]],posy[p[2]],size[p[2]],yellow);
       Circle(posx[P[2]]-absx,480-(posz[P[2]]-absz),size[P[2]]);
       SetColor(Green);
       Outtextxy(460,202,'Q ('+I2S(Q)+') :
'+I2S(PosX[Q])+','+I2S(PosY[Q])+','+I2S(PosZ[Q]));
       Particle(posx[q],posy[q],size[q],green);
       Circle(posx[q]-absx,480-(posz[q]-absz),size[q]);
       OutTextXY(460,238,chr($E5));                     {Delta}
       OutTextXY(460,250,chr($EE));                     {Epsilon}
       OutTextXY(460,262,chr($E0));                     {Alpha}
       OutTextXY(460,274,'IQ : '+R2S(IQ,3,2));
       OutTextXY(460,286,'QC : '+R2S(QC,3,2));
       OutTextXY(460,298,'Centre '+R2S(CenX,3,2)+','+R2S(CenY,3,2)+'
'+R2S(CenZ,3,2));
       OutTextXY(460,310,'Angle  '+R2S(RollAng*DegRad,9,4));
       OutTextXY(460,322,'Radius '+R2S(R,9,4));
       SetTextJustify(RightText,TopText);
       OutTextXY(540,238,R2S(Delta*DegRad,9,4)+chr($F8));
       OutTextXY(540,250,R2S(Epsilon*DegRad,9,4)+chr($F8));
       OutTextXY(540,262,R2S(-Alpha*DegRad,9,4)+chr($F8));
       SetTextJustify(LeftText,TopText);

       setcolor(lightgray);
       line(round(x)-absx,480-(round(z)-absz),posx[q]-absx,480-(posz[q]-
absz));
       line(posx[P[2]]-absx,480-(posz[P[2]]-absz),posx[q]-absx,480-(posz[q]-
absz));
       line(posx[P[2]]-absx,480-(posz[P[2]]-absz),round(x)-absx,480-
(round(z)-absz));
       cross(round(cenx-absx),480-round(cenz-absz));
       line(round(cenx-absx),round(480-(cenz-absz)),
            round(cenx-30*sin(rollang)-absx),
            480-round(cenz-30*cos(rollang)-absz));

        SetColor(Green);
        end;

    Rest:=False;
    WasTouching:=2;

    repeat
          X:=CenX-R*Cos(Alpha)*Sin(RollAng);
          Z:=CenZ-R*Cos(Alpha)*Cos(RollAng);
          Y:=CenY+R*Sin(Alpha);

          Alpha:=Alpha+M;

          If Debug='On' then begin
             PutPixel(round(x)-absx,480-(round(z)-absz),lightblue);
             SetColor(green);
             PutPixel(round(X),round(Y),LightGreen+ord(XRoll));
             Cross(PosX[Q],PosY[Q]);
             SetTextJustify(RightText,TopText);
             OutTextXY(599,170,R2S(X,3,0)+','+R2S(Y,3,0)+','+R2S(Z,3,0));
             OutTextXY(540,262,R2S(-Alpha*DegRad,9,4)+chr($F8));
             Delay(wait);
             DelText(599,170,R2S(X,3,0)+','+R2S(Y,3,0)+','+R2S(Z,3,0));
             Deltext(540,262,R2S(-Alpha*DegRad,9,4)+chr($F8));
             SetTextJustify(LeftText,TopText);
             end;

          RollTest3D(X,Y,Z,Q);

          until Rest or (Alpha >= 0);

    if (Alpha >= 0) then condition:='f';

    if Condition='w' then begin
       {Backs off particle in order to remain within cell boundaries}
       Dist:=sqrt(sqr(X-CellCen)+sqr(Z-CellCen));
       if Dist>CellRad then begin
          Alpha:=Alpha-M;
          X:=CenX-R*Cos(Alpha)*Sin(RollAng);
          Z:=CenZ-R*Cos(Alpha)*Cos(RollAng);
          Y:=CenY+R*Sin(Alpha);
          end;
       end;


    if Debug='On' then begin
       DelText(460,202,'Q ('+I2S(Q)+') :
'+I2S(PosX[Q])+','+I2S(PosY[Q])+','+I2S(PosZ[Q]));
       DelText(462,214,'P'+Chr($FD)+' ('+I2S(P[2])+') : '
       +R2S(PosX[P[2]],3,1)+','+R2S(PosY[P[2]],3,1)+','+R2S(PosZ[P[2]],3,1));
       DelText(461,226,'I  : '+R2S(X,3,1)+','+R2S(Y,3,1)+','+R2S(Z,3,1));
       DelText(460,238,chr($E5));                        {Delta}
       DelText(460,250,chr($EE));                        {Epsilon}
       DelText(460,262,chr($E0));                        {Alpha}
       DelText(460,274,'IQ : '+R2S(IQ,3,2));
       DelText(460,286,'QC : '+R2S(QC,3,2));
       DelText(460,298,'Centre '+R2S(CenX,3,2)+','+R2S(CenY,3,2)+'
'+R2S(CenZ,3,2));
```

```
           DelText(460,310,'Angle  '+R2S(RollAng*DegRad,9,4));
           DelText(460,322,'Radius '+R2S(R,9,4));
           SetTextJustify(RightText,TopText);
           DelText(460,238,R2S(delta*DegRad,9,4)+chr($F8));
           DelText(460,250,R2S(epsilon*DegRad,9,4)+chr($F8));
           SetTextJustify(LeftText,TopText);
           Particle(posx[q],posy[q],size[q],red);
           for K:=2 to 4 do Particle(posx[P[K]],posy[P[K]],size[P[K]],red);
           end;

        landed:

end;
{**************************************************************************}
procedure Roll3D(Q : integer);

var
   absx,absz : integer;

begin
      if Debug='On' then begin
           SetTextJustify(RightText,TopText);
           DelText(599,170,I2S(PosX[I-1])+','+I2S(PosY[I-1])+','+I2S(PosZ[I-1]));
           SetTextJustify(LeftText,TopText);
           DelText(540,170,'xxx,yyy,zzz');
           end;

      repeat

      Case Touch of
           1 : Roll1(Q);
           2 : Roll2(Q);
           end; {Case}

      until rest or (condition='f');

      if Debug='On' then begin
           Particle(posx[q],posy[q],size[q],red);
           end;

      ClearSpace(441,191,639,479);

      PosX[I]:=round(X); PosY[I]:=round(Y); PosZ[I]:=round(Z)

end;
{**************************************************************************}
end.
```

SCREENIN.PAS - Contains information and routines for various types of screen output

```
unit screeninfo;

{$N+,E+}

interface

uses Crt,Dos,Graph,funcs,variables;

var
   Changed : boolean;

procedure DelText(X,Y:integer; Phrase:string);
procedure ClearSpace(X1,Y1,X2,Y2:integer);
procedure Particle(X,Y,size:integer; Col : word);
procedure DimBox;
procedure DebugControl;
procedure AggCirc;
procedure FiltCell;
procedure Camera(X,Y : integer);
procedure Sun(X,Y : integer);
procedure BuildScr;
procedure redraw;
procedure PlanDraw;
procedure RevolveScr;
procedure Cursor;
procedure Isolate;
procedure Flood;
procedure Review;
procedure SEMView;
procedure SEMDraw;
procedure Crosshairs(MidX, MidY : integer);
procedure MakeWindow(options : integer);
procedure CursorOff;
procedure CursorOn;
procedure Ask;
procedure StartPoint;
procedure Continue;
procedure GoBack;
procedure Crash(Message : String);
procedure DiskError(Error : integer);

implementation
var
   Orig : array [0..6,0..6] of word;
   X,Y : integer;
   dX,dY : integer;

{*************************************************************************}
procedure DelText(X,Y:integer; Phrase:string);

var
   Orig : integer;

begin
     Orig:=GetColor;
     SetColor(Black);
     OutTextXY(X,Y,Phrase);
     SetColor(Orig);
end;
```

```
{***************************************************************************}
procedure ClearSpace(X1,Y1,X2,Y2:integer);
{CLEARs rectangular SPACE defined by X1,Y1 - X2,Y2}

var
Rectangle : array [1..4,1..2] of integer;
Orig : integer;

begin
     Rectangle[1,1]:=X1;  Rectangle[1,2]:=Y1;
     Rectangle[2,1]:=X2;  Rectangle[2,2]:=Y1;
     Rectangle[3,1]:=X2;  Rectangle[3,2]:=Y2;
     Rectangle[4,1]:=X1;  Rectangle[4,2]:=Y2;

     Orig:=GetColor;
     SetColor(Black);
     SetFillStyle(SolidFill,Black);
     FillPoly(4, Rectangle);
     SetColor(Orig);
end;
{***************************************************************************}
procedure particle(X,Y,size:integer; Col : word);               {Draws & Stores
Particle}

var
   Orig : integer;

begin
     Orig:=GetColor;
     SetColor(Green);
     SetFillStyle(SolidFill,Col);
     FillEllipse(X,Y,Size,Size);          {Fills it!}
     SetColor(Orig);
end;

{***************************************************************************}
Procedure DimBox;
begin
     SetTextStyle(SmallFont,0,4);
     SetColor(LightGreen);
     Rectangle(10,10,30,26);
     SetFillStyle(SolidFill,Red);
     FloodFill(20,17,LightGreen);
     SetTextJustify(CenterText,CenterText);

     if FileType='S' then begin
     OutTextXY(20,17,'SEM');
     end
     else begin
     OutTextXY(20,17,I2S(Dimensions)+'-D');
     end;
end;
{***************************************************************************}
procedure DebugControl;

var
   OldStyle: TextSettingsType;

begin
     Option:=Readkey;
     GetTextSettings(OldStyle);
     if Model<>'F' then begin
     DelText(560,170,'xxx,yyy');
     SetTextJustify(CenterText,TopText);
     if Debug='Off' then begin
        SetColor(Red);
        Rectangle(495,202,622,264);
        Debug:='On';
        SetColor(LightGreen);
        OutTextXY(558,210,'Visual Debugging');
        SetColor(Green);
        OutTextXY(558,228,'Current Delay : '+I2S(Wait)+'ms');
        OutTextXY(558,246,'Use +/- to change');
        end
     else begin
        DelText(558,228,'Current Delay : '+I2S(Wait)+'ms');
        case Option of
             '+' : Wait:=Wait+5;
             '-' : if Wait>=5 then Wait:=Wait-5;
             else begin
                  SetColor(Black);
                  Rectangle(495,202,622,264);
                  DelText(558,210,'Visual Debugging');
                  DelText(558,246,'Use +/- to change');
                  Debug:='Off';
                  end;
             end;                                {End Case Option}
        if (Option='+') or (Option='-') then begin
           SetColor(Green);
           OutTextXY(558,228,'Current Delay : '+I2S(Wait)+'ms');
           end;
        end;                                {End Debug='On'}
        end                                 {End Model<>'F'}
        else if debug='On' then Debug:='Off' else Debug:='On';

        with OldStyle do begin                   { Restore old text style }
             SetTextJustify(Horiz, Vert);
             SetTextStyle(Font, Direction, CharSize);
             end;
end;
{************************************************************************}
Procedure AggCirc;
begin
     SetColor(LightMagenta);
     Circle(CentreX,CentreY,radius);
end;
{************************************************************************}
Procedure FiltCell;
begin
        SetColor(LightMagenta);
        Rectangle(LWall,Roof,RWall,Fl);
        SetColor(Black);
        Line(LWall,Roof,RWall,Roof);
        SetColor(LightMagenta);
        SetLineStyle(DottedLn, 0, NormWidth);
        Line(10,40,440,40);
        SetLineStyle(SolidLn, 0, NormWidth);
end;
{************************************************************************}
procedure camera(X,Y : integer);

begin
```

```
     SetColor(LightGray);                                   {Body}
     SetFillStyle(InterLeaveFill,DarkGray);
     Bar(X-10,Y-7,X+10,Y+7);
     Rectangle(X-10,Y-7,X+10,Y+7);
     Rectangle(X+4,Y-10,X+8,Y-7);                           {Knob}
     SetColor(LightGray);                                   {Lens}
     SetFillStyle(SolidFill,LightGray);
     FillEllipse(X,Y,4,4);
     SetColor(Black);
     SetFillStyle(SolidFill,DarkGray);
     FillEllipse(X,Y,3,3);

end;
{***************************************************************************}
procedure sun(X,Y : integer);

var
   I : single;
   J : integer;
   Xstart, Xend, Ystart, Yend : single;

begin
     SetColor(Yellow);
     for J:=1 to 12 do begin
         I:=J/(180/Pi)*30;
         Xstart:=cos(I)*6+X;
         Xend:=cos(I)*8+X;
         Ystart:=sin(I)*6+Y;
         Yend:=sin(I)*8+Y;
         line(round(Xstart),round(Ystart),round(Xend),round(Yend));
     end;
     SetFillStyle(Solidfill,Yellow);
     FillEllipse(X,Y,4,4);
end;
{***************************************************************************}
procedure BuildScr;

var
   h,m,s, hund : word;

begin
     if Model='F' then FiltCell else Aggcirc;
     GetTime(h,m,s,hund);
     GoTime:=(h*3600)+(m*60)+s+(hund/100);
     if numgen='On' then
        begin
             Str(h,hr); Str(m,mn); Str(s,sc);
             Time:=(h*3600+m*60+s);
        end;
     if (Numgen='On') or (Numgen='Set') then TimeStr:=hr+':'+mn+':'+sc
     else TimeStr:='00:00:00';
     if numgen='Off' then randseed:= 0 else randseed:=round(Time);

     DimBox;
     Rectangle(495,5,622,190);
     SetColor(Green);
     SetTextJustify(RightText,TopText);
     if NumGen='Off' then OutTextXY(599,50,'Held at '+I2S(RandSeed))
     else OutTextXY(599,50,TimeStr);
     SetTextJustify(LeftText,TopText);
     OutTextXY(500,50,'Seed: ');
```

```
      if Model<>'F' then begin
         OutTextXY(500,70,'Seed Size: ');
         OutTextXY(575,70,R2S(Seed,4,0));
         end
      else begin
         SimInfo:='D:'+R2S(DownProb,3,0)+' S:'+R2S(StickProb,3,0);
         OutTextXY(500,70,SimInfo);
         end;
      OutTextXY(500,90,'Min. Size: ');
      OutTextXY(575,90,R2S(MinSize,4,0));
      OutTextXY(500,110,'Max. Size: ');
      OutTextXY(575,110,R2S(MaxSize,4,0));
      OutTextXY(500,130,'Particle #');
      Line(500,145,615,145);
      OutTextXY(500,150,'Agg. Size: ');
      OutTextXY(575,150,R2S(number,4,0));
      OutTextXY(500,170,'Pos.: ');
end;
{**************************************************************************}
procedure redraw;
begin

      ClearViewPort;
      SetColor(Blue);
      SetFillStyle(SolidFill,Blue);

      for K:=1 to Number do
          begin
               FillEllipse(PosX[K],PosY[K],Size[K],Size[K]);
          end;

      if Debug='On' then if FileType='M' then FiltCell else AggCirc;
end;
{**************************************************************************}
procedure PlanDraw;
begin

      ClearViewPort;
      SetColor(Blue);
      SetFillStyle(SolidFill,Blue);
      CellRad:=(RWall-LWall) div 2;
      CellCen:=LWall+CellRad;              {Define CellCentre}

      for K:=1 to Number do FillEllipse(PosX[K],PosZ[K],Size[K],Size[K]);

      if Debug='On' then begin
         SetColor(LightMagenta);
         Circle(CellCen,CellCen,CellRad);
         end;
end;

{**************************************************************************}
Procedure RevolveScr;

var
   I : integer;

begin
      {Clears old text entry}
      Deltext(450,380,'Press F2 to save & return');
      Deltext(450,400,'Press F3 to rotate agglomerate');
```

```
     DelText(450,420,TimeTxt);
     Deltext(450,460,'Press any key to continue');
     SetFillStyle(SolidFill,Black);
     {Empties T.L box}
     FloodFill(550,100,LightGreen);
     SetColor(LightGreen);
     SetTextJustify(LeftText,TopText);
     OutTextXY(450,460,'Use Number Pad');
     SetColor(Green);
     OuttextXY(450,380,'F2 : save .dat file');
     OuttextXY(450,400,'F3 : save .pov file');
     OutTextXY(450,420,'F4 : save animation file');
     ShowName;
     OutTextXY(500,50,'Rotate Up');
     OutTextXY(500,70,'Rotate Down');
     OutTextXY(500,90,'Rotate Right');
     OutTextXY(500,110,'Rotate Left');
     OutTextXY(500,130,'+       to rotate 45ø');
     OutTextXY(500,150,'Original');
     OutTextXY(500,170,'Quit');
     SetTextStyle(DefaultFont,0,1);
     SetTextJustify(CenterText,CenterText);
     for I:=0 to 3 do begin                {Draws Cursor Keys}
         SetColor(DarkGray);
         Rectangle(580,50+I*20,590,60+I*20);
         SetFillStyle(SolidFill,LightGray);
         FloodFill(585,55+I*20,DarkGray);
         DelText(585,56+I*20,chr(I+24));
     end;
     SetColor(DarkGray);
     Rectangle(570,150,600,162);         {Home}
     FloodFill(585,155,DarkGray);
     Rectangle(510,130,542,142);         {SHIFT}
     FloodFill(530,135,DarkGray);
     Rectangle(574,170,596,182);         {Esc}
     FloodFill(585,175,DarkGray);
     SetTextStyle(SmallFont,0,4);
     DelText(585,155,'Home');
     DelText(526,135,'SHIFT');
     DelText(585,175,'Esc');
     SetColor(Green);
end;
{****************************************************************************}
Procedure CursDraw(col : integer);

begin
     for I:=0 to 6 do begin                     {Put orig. colours back}
         for J:=0 to 6 do begin
             PutPixel(I+X-3,J+Y-3,Orig[I,J]);
             end;
         end;
     DelText(554,180,R2S(X,3,0)+','+R2S(Y,3,0));
     X:=X+dX; Y:=Y+dY;
     SetColor(Col);
     OutTextXY(554,180,R2S(X,3,0)+','+R2S(Y,3,0));
     for I:=0 to 6 do begin                            {Gets orig. colours}
      for J:=0 to 6 do begin
       Orig[I,J]:=GetPixel(X+(I-3),Y+(J-3));
       end;
      end;
     SetColor(LightGray);
     line(X+3,Y,X-3,Y);
     line(X,Y-3,X,Y+3);
end;
{**************************************************************************}
procedure Instruct(which:char);
var
 I,J,K : Integer;

label
 restart;

begin
     Aborted:=False;

     Case Which of
          'S' : Col:=LightGreen;
          'E' : Col:=Red;
          end;
     SetTextJustify(CenterText,TopText);
     DelText(554,10,'Analyzing...');
     SetTextJustify(LeftText,TopText);
     Deltext(450,460,'Press Enter to continue');
     {Empties T.L box}
     SetColor(LightGreen);
     Rectangle(485,30,622,160);
     SetFillStyle(SolidFill,Black);
     FloodFill(550,95,LightGreen);
     if ((WImage-RB+ExtraX)>485) and ((TB+ExtraY)<160) and (FileType='S') then
begin
        {Draw pat of instruction box in blue}
        SetColor(Blue);
        Line(485,160,(WImage-RB+ExtraX),160);
        Line(485,(TB+ExtraY),485,160);
        end;
     SetColor(Green);
     OutTextXY(500,50,'Move Up');
     OutTextXY(500,68,'Move Down');
     OutTextXY(500,86,'Move Right');
     OutTextXY(500,104,'Move Left');
     OutTextXY(500,122,'Save');
     OutTextXY(500,140,'Abort');
     SetTextStyle(DefaultFont,0,1);
     SetTextJustify(CenterText,CenterText);
     SetColor(DarkGray);
     for I:=0 to 3 do begin              {Draws Cursor Keys}
         Rectangle(580,50+I*18,590,60+I*18);
         SetFillStyle(SolidFill,LightGray);
         FloodFill(585,55+I*18,DarkGray);
         DelText(585,56+I*18,chr(I+24));
     end;
     Rectangle(574,122,596,134);         {'Esc' Box}
     FloodFill(585,127,DarkGray);
     SetTextStyle(SmallFont,0,4);
     DelText(585,127,'Esc');
     Rectangle(574,140,596,152);         {'Q' Box}
     FloodFill(585,145,DarkGray);
     SetTextStyle(SmallFont,0,4);
     DelText(585,145,'Q');

     SetColor(Green);
     dX:=0; dY:=0;
```

```
case FileType of
'M' : begin
X:=FurthX+ExtraX; Y:=FurthY+ExtraY;
end;
'S' : begin
X:=FurthX; Y:=FurthY;
end;
end;

PutPixel(X,Y,Blue);
for I:=0 to 6 do begin                         {Gets orig. colours}
 for J:=0 to 6 do begin
  Orig[I,J]:=GetPixel(X+(I-3),Y+(J-3));
  end;
 end;
repeat

restart:

Option:=ReadKey;

if Option=#0 then begin
   Option:=ReadKey;
   K:=ord(Option);
   dX:=0; dY:=0;
   case K of
        75 : dX:=-1;           {Left}
        71 : dX:=-15;          {Home - long left}
        77 : dX:=+1;           {Right}
        79 : dX:=+15;          {End - long right}
        72 : dY:=-1;           {Up}
        73 : dY:=-15;          {PgUp - long Up}
        80 : dY:=+1;           {Down}
        81 : dY:=+15           {PgDn - long down}
   end;             {End case}
   Changed:=True;
   end              {Ends special key catch}
   else begin
        J:=ord(upcase(option));
        dX:=0; dY:=0;
        end;
 CursDraw(Col);
 until (J=27) or (J=81);

Aborted:=J=81;

if not(aborted) and (Orig[3,3]<>0) then begin
   StartPoint;
   J:=0;
   GoTo restart;
   end;

for I:=0 to 6 do begin
    for J:=0 to 6 do begin
        PutPixel(I+X-3,J+Y-3,Orig[I,J]);
        end;
    end;

if not Aborted then begin
if PoreMeasure then PutPixel(X,Y,Col);
FurthX:=X-ExtraX; FurthY:=Y-ExtraY;
```

```
     end
     else PutPixel(FurthX+ExtraX,FurthY+ExtraY,Col);
end;
{*************************************************************************}
Procedure Cursor;

var
   I : integer;

begin
     Repeat;
     SetTextJustify(LeftText,TopText);
     SetColor(LightGreen);
     OuttextXY(450,380,'F1 : move start point');
     SetColor(Red);
     OuttextXY(450,400,'F2 : move end point');
     SetColor(Green);
     OuttextXY(450,460,'Press Enter to continue');
     ShowName;
     Option:=ReadKey;
     if Option=#0 then begin
        Option:=ReadKey;
        J:=ord(Option);
        Deltext(450,380,'F1 : move start point');
        Deltext(450,400,'F2 : move end point');
        case J of
             59 : begin                              {F2}
                  instruct('S');
                  end;
             60 : begin                              {F3}
                  instruct('E');
                  end;
            end;
        end              {Ends special key catch}
        else begin
             J:=ord(option);
             end;
     DelText(554,180,R2S(X,3,0)+','+R2S(Y,3,0));
     until J=13;

     SetTextJustify(LeftText,TopText);
     DelText(450,460,'Use Number Pad');
     Deltext(450,380,'F1 : move start point');
     Deltext(450,400,'F2 : move end point');
     Deltext(450,460,'Press Enter to continue');
     SetTextJustify(CenterText,CenterText);
end;
{*************************************************************************}
procedure Isolate;
var
   I : integer;

begin
     SetTextJustify(CenterText,TopText);
     if FileType='S' then begin
        OuttextXY(100,440,'Please move crosshair');
        OuttextXY(100,460,'to pore space');
        end
     else begin
          OuttextXY(550,250,'Please move crosshair');
          OuttextXY(550,270,'to pore space');
          end;
      SetTextJustify(LeftText,TopText);
      ShowName;
      Instruct('S');
      DelText(554,180,R2S(X,3,0)+','+R2S(Y,3,0));
      SetTextJustify(CenterText,TopText);
      if FileType='S' then begin
          Deltext(250,440,'Red areas indicate');
          Deltext(250,460,'previously saved pores');
         end
      else begin
          Deltext(550,290,'Red areas indicate');
          Deltext(550,310,'previously saved pores');
          end;
      SetTextJustify(CenterText,CenterText);
end;
{***************************************************************************}
procedure ScanLine(Dir : Char;X,Y,D,T : integer);

begin
      delay(wait);
      SetColor(Black);
      if Dir='H' then Line(X+2*D,Y+1*T,X+8*D,Y+1*T)
      else Line(X+1*T,Y+2*D,X+1*T,Y+8*D);
      SetColor(LightMagenta);
      if Dir='H' then Line(X+2*D,Y,X+8*D,Y)
      else Line(X,Y+2*D,X,Y+8*D);
end;
{***************************************************************************}
procedure PoreScan;
var
   X,Y : integer;
   X1,X2,Y1,Y2 : integer;

begin
      SetTextJustify(CenterText,TopText);
      Case FileType of
            'M' : begin
                  Deltext(550,250,'Please move crosshair');
                  Deltext(550,270,'to pore space');
                  X1:=LWall; X2:=RWall;
                  Y1:=Roof; Y2:=Fl;
                  end;
            'S' : begin
                  Deltext(100,440,'Please move crosshair');
                  Deltext(100,460,'to pore space');
                  X1:=LB+ExtraX; X2:=WImage-RB+ExtraX;
                  Y1:=TB+ExtraY; Y2:=HImage-BB+ExtraY;
                  end;
            end;

      {Left to right}
      X:=X1;
      repeat
            X:=X+1; Y:=Y1;
            repeat
                  Y:=Y+1;
                  Flag:=GetPixel(X,Y)=Blue;
                  until (flag) or (Y=Y2-1);
                  if debug='On' then begin
                     ScanLine('V',X,Y1,-1,-1);
```

```
                    ScanLine('V',X,Y2,1,-1);
                    end;
            until flag;
     LB:=X;

     {Right to left}
     X:=X2;
     repeat
            X:=X-1; Y:=Y1;
            repeat
                   Y:=Y+1;
                   Flag:=GetPixel(X,Y)=Blue;
                   until (flag) or (Y=Y2-1);
                   if debug='On' then begin
                      ScanLine('V',X,Y1,-1,1);
                      ScanLine('V',X,Y2,1,1);
                      end;
            until flag;
     RB:=X;

     WImage:=RB-LB;

     {Top to bottom}
     Y:=Y1;
     repeat
            Y:=Y+1; X:=X1;
            repeat
                   X:=X+1;
                   Flag:=GetPixel(X,Y)=Blue;
                   until (flag) or (X=X2-1);
                   if debug='On' then begin
                      ScanLine('H',X1,Y,-1,-1);
                      ScanLine('H',X2,Y,1,-1);
                      end;
            until flag;
     TB:=Y;

     {Bottom to top}
     Y:=Y2;
     repeat
            Y:=Y-1; X:=X1;
            repeat
                   X:=X+1;
                   Flag:=GetPixel(X,Y)=Blue;
                   until (flag) or (X=X2-1);
                   if debug='On' then begin
                      ScanLine('H',X1,Y,-1,1);
                      ScanLine('H',X2,Y,1,1);
                      end;
            until flag;
     BB:=Y;

     HImage:=BB-TB;
     DelText(554,10,'Analyzing...');
end;
{****************************************************************************}
procedure Flood;

var
   X, Y, X1, X2, Y1, Y2 : integer;
```

```
begin
     PoreArea:=0;
     SetColor(LightGreen);
     SetTextJustify(CenterText,TopText);
     if FileType='S' then begin
        Deltext(100,440,'Please move crosshair');
        Deltext(100,460,'to pore space');
        end
     else begin
          Deltext(550,250,'Please move crosshair');
          Deltext(550,270,'to pore space');
          end;

     OutTextXY(554,10,'Analyzing...');
     SetFillStyle(SolidFill,LightMagenta);
     FloodFill(FurthX+ExtraX,FurthY+ExtraY,Blue);
     SetFillStyle(SolidFill,Black);

     SetTextJustify(LeftText,TopText);
     SetColor(Green);
     OutTextXY(450,420,'Area of pore : ');

     Case FileType of
          'M' : begin
                X1:=LWall; X2:=RWall;
                Y1:=Roof; Y2:=Fl;
                end;
          'S' : begin
                X1:=LB+ExtraX; X2:=WImage-RB+ExtraX;
                Y1:=TB+ExtraY; Y2:=HImage-BB+ExtraY;
                end;
          end;

        for Y:=Y1+1 to Y2-1 do begin
           for X:=X1+1 to X2-1 do
               if GetPixel(X,Y)=LightMagenta then begin
                  if (Y=Y1+1) or (Y=Y2-1) or (X=X1+1) or (X=X2-1) then begin
                     SetTextJustify(CenterText,CenterText);
                     SetColor(LightMagenta);
                     if FileType='S' then
                        OutTextXY(100,420,'Invalid Pore Space')
                        else OutTextXY(550,320,'Invalid Pore Space');
                     StartPoint;
                     InvPore:=True;
                     exit;
                     end;
               PutPixel(X,Y,Blue);
               if debug='On' then DelText(544,420,R2S(PoreArea,5,0)+'
pixels');
               PoreArea:=PoreArea+1;
               if debug='On' then begin
                  SetColor(Green);
                  OutTextXY(544,420,R2S(PoreArea,5,0)+' pixels');
                  end;
               end
               else PutPixel(X,Y,Black);
               if debug='On' then begin
                  ScanLine('H',X1,Y,-1,-1);
                  ScanLine('H',X2,Y,1,-1);
                  end;
           end;
        SetColor(Green);
        OutTextXY(544,420,R2S(PoreArea,5,0)+' pixels');
        ShowName;
        SetColor(Black);
        Line(X2+2,Y,X2+10,Y);
        Line(X1-2,Y,X1-10,Y);
        PoreScan;
end;
{*************************************************************************}
Procedure Review;
var
   I : integer;

begin
     Radius:=240;
     if Model='F' then FiltCell else Aggcirc;
     SetTextStyle(SmallFont,0,4);
     SetColor(LightGreen);
     Rectangle(495,5,622,190);
     SetTextJustify(CenterText,TopText);
     OutTextXY(559,10,Title);
     OutTextXY(559,30,'Manual Viewing');
     if (Dimensions=2) or (Twist='Off') then begin
     SetColor(Green);
     OutTextXY(559,50,'Seed');
     OutTextXY(559,85,'Stopwatch');
     OutTextXY(559,120,'Particles');
     if Model='F' then OutTextXY(559,155,'Carlo Info.');
     SetColor(Red);
     OutTextXY(559,62,TimeTxt);
     OutTextXY(559,97,TimeStr);
     OutTextXY(559,132,I2S(Number));
     if Model='F' then OutTextXY(559,167,SimInfo);
     end;
     GoBack;
end;
{*************************************************************************}
procedure SEMView;

begin
     SEMDraw;
     SetColor(LightGreen);
     Rectangle(495,5,622,40);
     SetTextJustify(CenterText,TopText);
     OutTextXY(559,7,Title);
     OutTextXY(559,23,'Manual Viewing');
     GoBack;
end;
{*************************************************************************}
procedure SEMDraw;

var
   K : Integer;
   Fig : char;

begin
     ClearViewPort;
     SetColor(Blue);
     SetFillStyle(SolidFill,Blue);

     DimBox;
     Assign(Storage,save);
     Reset(Storage);

     for J:=1 to TB do Readln(Storage);          {Miss Top Border}
     for J:=TB to (HImage-BB) do begin           {Reads until BB}
         for K:=1 to LB do read(Storage,Fig);    {Miss Left Border}
         for K:=LB+1 to (WImage-RB) do begin     {Reads unitl RB}
             if GetPix=1 then PutPixel(K+ExtraX,J+ExtraY,Blue);
         end;
         Readln(Storage);
         end;
     Close(Storage);
end;
{******************************************************************************}
Procedure Crosshairs(MidX,MidY : integer);      {Centre Crosshairs}
begin
     SetColor(DarkGray);
     SetLineStyle(DottedLn,0,NormWidth);
     Line(MidX,0,MidX,2*MidY);
     Line(0,MidY,2*MidX,MidY);
     SetLineStyle(SolidLn,0,NormWidth);
end;
{******************************************************************************}
Procedure MakeWindow(options : integer);

var
K : integer;

begin
     Window(1,1,80,25);
     ClrScr;
     TextColor(Green);
     Window(15,2,64,options+7);
     Writeln('ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»');
     For K:=1 to options+3 do
     begin
          Writeln('°'); GotoXY(49,(K+1)); Writeln('°');
     end;
     Writeln('ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼');
     Window(16,3,62,options+5);
     TextBackground(Black);
     ClrScr;
     TextColor(White);
     Writeln('       Parameters are set as follows:');
     Writeln;
     TextColor(Green);
end;
{*****************************************************************************}
procedure CursorOff;                                {Turn Cursor off}
var
   Regs : Registers;

begin
   Regs.AH:=$01; Regs.CH:=$20; Regs.CL:=$20; Intr($10,Regs);
end;
{*****************************************************************************}
procedure CursorOn;                                 {Turn Cursor on}
var
   Regs : Registers;
```

```
begin
     Regs.AH:=$01; Regs.CH:=6; Regs.CL:=7; Intr($10,Regs);
end;
{**********************************************************************}
procedure ask;
begin
     Window(6,17,70,20);
     ClrScr;
     TextColor(Green);
     Write('Do you wish to change any of these values Y/[N] ? ');
end;
{***********************************************************************}
procedure StartPoint;
var
   x,y : integer;

begin
     SetTextJustify(CenterText,CenterText);
     SetColor(LightGray);

     if FileType='S' then begin
        x:=100; y:=440; end
     else begin
        x:=550; y:=340; end;

     OuttextXY(x,y,'Please find another');
     OuttextXY(x,y+20,'startpoint or Abort (''Q'')');
     SetTextJustify(LeftText,TopText);
     SetColor(Green);
     OuttextXY(450,460,'Press any key to continue');
     Option:=Readkey;
     Deltext(450,460,'Press any key to continue');
     SetTextJustify(CenterText,CenterText);
     Deltext(x,y,'Please find another');
     Deltext(x,y+20,'startpoint or Abort (''Q'')');
end;
{***********************************************************************}
procedure Continue;
begin
     SetTextJustify(LeftText,TopText);
     SetColor(Green);
     ShowName;
     OuttextXY(450,460,'Press any key to continue');
     Option:=Readkey;
     Deltext(450,460,'Press any key to continue');
end;
{***********************************************************************}
procedure GoBack;
begin
     SetTextJustify(LeftText,TopText);
     SetColor(Green);
     ShowName;
     OuttextXY(450,460,'Press any key to continue');
end;
{**********************************************************************}
procedure Crash(Message : String);
begin
MakeWindow(4);
ClrScr;
TextColor(White+Blink);
Writeln('                Program has failed :');
```

```
TextColor(White);
GotoXY(23-(Length(Message) div 2),4);
Writeln(Message);
Window(6,17,70,20);
Writeln;
halt;
end;
{*************************************************************************}
procedure DiskError(Error : Integer);
var
   Message : string;

begin
case Error of
     5 : Message:='Please close open files & try again';
     end;

MakeWindow(4);
CursorOff;
ClrScr;
Writeln('                Disk Error :');
TextColor(White);
GotoXY(23-(Length(Message) div 2),3);
Writeln(Message);
GotoXY(12,5);
Writeln('Press any key to continue');
Option:=Readkey;
CursorOn;
end;
{*************************************************************************}
end.
```

SWALK.PAS - Structured Walk analysis algorithms for different simulation types

```
unit swalk;

{$N+,E+}

interface

uses Crt,Dos,Graph,
     disk,enclosing,funcs,gyration,menus,mover,porosity,
     revolve,screeninfo,variables;

var
   FailTxt : string;

procedure FiltSteps;
procedure AggFerets;
procedure AggWalk(OutX, OutY, Size : integer);
procedure SEMFerets;
procedure SEMWalk(OutX, OutY : integer);
procedure PoreFerets;
procedure PoreWalk(OutX, OutY : integer);
procedure FibreWalk;

implementation

var
     Exists, Right, Top : boolean;
     Edge : single;
     Look : File;
{***************************************************************************}
procedure InfoBox;

begin
   SetColor(LightGreen);
   SetTextJustify(CenterText,TopText);
   OutTextXY(554,10,'Perimeter Analysis...');
   SetTextJustify(LeftText,TopText);
   SetLineStyle(0,0,1);
   Rectangle(485,30,622,160);
   {Redraws information after staring posn. has been changed -
                                          deletes cursor instructions}
   if Changed then begin
      SetFillStyle(SolidFill,Black);
      FloodFill(559,95,LightGreen);
      SetColor(Green);
      OutTextXY(560,55,R2S(FeretMin,7,2));
      OutTextXY(560,75,R2S(FeretMax,7,2));
      SetColor(LightGreen);
      Changed:=False;
      end;
   SetTextJustify(CenterText,TopText);
   OutTextXY(559,35,Title);
   SetTextJustify(LeftText,TopText);
   SetColor(Green);
   OutTextXY(490,55,'Min. Feret :');
   OutTextXY(490,75,'Max. Feret :');
   OutTextXY(490,95,'Fraction :');
   OutTextXY(490,115,'Steplength :');
   OutTextXY(490,135,'Perimeter :');
end;
```

```
{***************************************************************************}
procedure ResultBox;
begin
     if Increments>3 then begin
        SetTextJustify(CenterText,TopText);
        OutTextXY(554,35,Title);
        OutTextXY(554,155,chr($AF)+' D = '+R2S(1-linb,1,3)+' '+chr($AE));
        SetColor(Green);
        SetTextJustify(CenterText,TopText);
        Rectangle(485,190,622,204);
        SetFillStyle(SolidFill,Black);
        FloodFill(553,200,Green);
        if (R2S(LowFrac,1,3)='0.080') and (R2S(UppFrac,1,3)='0.320')
           and (R2S(Gap,1,3)='0.020') then
        OutTextXY(553,190,'Default Range') else
        OutTextXY(553,190,R2S(lowfrac,1,3)+' '+chr($AF)+' '+R2S(uppfrac,1,3)
               +' @ '+R2S(Gap,1,3));
        if (not multi) or (Debug='On') then Continue;
        end
end;
{***************************************************************************}
procedure steps(OutX, OutY, Size : integer);

var
     EndX, EndY : longint;
     StartX, StartY, FirstX, FirstY : integer;
     StartAng, Angle : integer;
     Attempts, NumSteps, Fail, N, Calc : integer;
     Posn, Checkdist : single;
     Test : Word;
     Failed : array [1..35] of boolean;

begin
     Calc:=2*ord(Top)+ord(Right);

     {Define step length as fraction of Feret Diameter}
     if Range='Yes' then LowFrac:=LowFrac-Gap;

     Increments:=round((UppFrac-LowFrac)/Gap);
     Fail:=0;

For N:=increments downto 1 do begin
     NumSteps:=0;
     Perim[N]:=0;
     Failed[N]:=False;
     Fraction:=LowFrac+(N*Gap);
     Steplength[N]:=Fraction * FeretMax;

     SetColor(Green);
     OutTextXY(560,95,R2S(Fraction,8,3));
     OutTextXY(560,115,R2S(Steplength[N],7,2));

     SetColor(LightBlue);

     Case Calc of
          3 : StartAng:=315;
          2 : StartAng:=225;
          0 : StartAng:=135;
          1 : StartAng:=45;
     end;
```

```
if Size>0 then begin
StartX:=OutX+round((Size*((ord(Right)*2)-1)*Edge));
StartY:=OutY-round((Size*((ord(Top)*2)-1)*Edge));
FirstX:=StartX; FirstY:=StartY;
end
else begin
StartX:=FurthX+ExtraX;
StartY:=FurthY+ExtraY;
FirstX:=StartX; FirstY:=StartY;
end;

repeat
      Inc(StartAng);
      Attempts:=0;
      repeat
            Inc(Attempts);
            StartAng:=StartAng-1;
            EndX:=round(StartX+Steplength[N]*Cos(StartAng/DegRad));
            EndY:=round(StartY+Steplength[N]*Sin(StartAng/DegRad));
            Test:=GetPixel(EndX,EndY);
            Failed[N]:=Attempts>360;
      until (test=black) or Failed[N];
      {Ensures starting from Black}

      Angle:=StartAng;
      if angle<0 then angle:=angle+360;

      Attempts:=0;
      if not failed[N] then repeat
            Inc(Attempts);
            Inc(Angle);
            posn:=Angle/DegRad;
            EndX:=round(StartX+Steplength[N]*Cos(Posn));
            EndY:=round(StartY+Steplength[N]*Sin(Posn));
            Test:=GetPixel(EndX,EndY);
            Failed[N]:=Attempts>360;
      until ((test=blue) and (EndX>0) and (EndX<GetMaxX))
            or Failed[N];

      If Debug='On' then Delay(wait);
      Checkdist:=sqrt(sqr(EndX-FirstX)+sqr(EndY-FirstY));
      Inc(NumSteps);

      if (Range='No') then line(StartX,StartY,EndX,EndY);

      StartX:=EndX; StartY:=EndY;
      StartAng:=Angle-90;
      If StartAng<0 then StartAng:=StartAng+360;
      Perim[N]:=Perim[N]+Steplength[N];
      if Range='Yes' then Failed[N]:=Perim[N]>5*FeretMax;

until (CheckDist<=(steplength[N]-1)) or (Failed[N]);

if (NumSteps<4) then Failed[N]:=True;

{**** '-1' because 1st CheckDist=Steplength ****}

If Failed[N] then begin
   Inc(Fail);
   OutTextXY(510,160+12*Fail,'Failed @ '+R2S(Fraction,5,3));
   FailTxt:=FailTxt+R2S(Fraction,5,3)+' ';
```

```
        end;

      Perim[N]:=Perim[N]+Checkdist;

      if Range='No' then line(EndX,EndY,FirstX,FirstY);

      SetColor(Green);
      DelText(560,95,R2S(Fraction,8,3));
      DelText(560,115,R2S(Steplength[N],7,2));
      DelText(560,135,R2S(Perim[N+1],7,2));
      OutTextXY(560,135,R2S(Perim[N],7,2));
  end;                                      {END N:=1 to increments}

      OutTextXY(560,95,R2S(Fraction,8,3));
      OutTextXY(560,115,R2S(Steplength[N],7,2));
      OutTextXY(560,135,R2S(Perim[N],7,2));
      if Range='Yes' then LowFrac:=LowFrac+Gap;

      SetTextJustify(CenterText,TopText);
      DelText(554,10,'Perimeter Analysis...');

      if (not multi) or (Debug='On') then Continue;

      if Range='Yes' then begin
      J:=1;
      for N:=1 to Increments do begin
          if not failed[N] then begin
             Steplength[J]:=ln(Steplength[N]);
             Perim[J]:=ln(Perim[N]);
             Inc(J);
             end;
          end;
      Increments:=Increments-Fail;
      GraphIt(Increments,1,'Structured Walk','ln Steplength','ln Perimeter',
                          Steplength,Perim);
      ResultBox;
      end;
end;                              {End 'Steps'}
{***************************************************************************}
function FindTop(L:single;Dir:char):integer;
var
   X,Xcheck,YCheck : integer;
   Test : Word;

begin
for X:=round(L) downto 1 do begin
    for YCheck:=Roof to Fl do begin
        if Dir='R' then XCheck:=LWall+X else XCheck:=RWall-X;
        Test:=GetPixel(XCheck,YCheck);
        if Test=Blue then begin
           FindTop:=YCheck;
           exit;
           end; {End Test=Blue}
        end;    {End YCheck}
    end;        {End X}
FindTop:=0;
end;
{***************************************************************************}
procedure FiltSteps;

var
```

```
     Fail, N, High, StartX, StartY, FirstX, FirstY : integer;
     EndX, EndY : longint;
     Turn,Centre,Attempts,StartAng, Angle, Swing : integer;
     grad_av,r_av,dx,dz,X,Z,Orig,NewAng,Posn, Checkdist : single;
     Test : Word;
     Next : Boolean;
     Failed : array [1..35] of boolean;

begin
     Centre:=(RWall+LWall) div 2;
     Turn:=-1;

     grad_av:=0; r_av:=0;

     Repeat

     if Dimensions=3 then ClearViewPort;
     DimBox;
     FiltCell;

     Inc(Turn);

     SetColor(Green);
     SetTextJustify(LeftText,TopText);
     ShowName;

     if Dimensions=2 then Turn:=2
     else begin
          OutTextXY(450,400,'Rotation : '+I2S(Turn*120)+CHR(248));
          SetColor(Blue);
          SetFillStyle(SolidFill,Blue);
          for K:=1 to Number do begin
              if Turn=0 then FillEllipse(PosX[K],PosY[K],Size[K],Size[K])
              else begin
                   dx:=PosX[K]-Centre;  dz:=PosZ[K]-Centre;
                   if dz=0 then if dx>=0 then Orig:=Pi/2
                                else Orig:=3*Pi/2
                   else Orig:=arctan(dx/dz);
                   if dz<0 then Orig:=Orig+Pi;
                   dx:=sqr(dx); dz:=sqr(dz);
                   Distance:=sqrt(dx+dz);
                   NewAng:=Orig+Turn*(2*Pi/3);
                   X:=Centre+Distance*Sin(NewAng);
                   FillEllipse(Round(X),PosY[K],Size[K],Size[K])
                   end;                         {End Turn<>0}
              end;                              {End K=1 to number do}
          end;                                  {End Dimensions=3}

     InfoBox;

     High:=Fl;
     for I:=1 to number do if (PosY[I]-Size[I]) < High then High:=(PosY[I]-
Size[I])-1;

     if debug='On' then begin
        {Draws Width across filter 'cell' just above highest particle}
        SetLineStyle(1,0,1);
        SetColor(LightRed);
        Line (Lwall,high-5,Rwall,high-5);
        end;
     OutTextXY(560,75,R2S(FeretMax,7,2));

     {Define step length as fraction of Feret Diameter}
     if Range='Yes' then LowFrac:=LowFrac-Gap;
     Increments:=round((UppFrac-LowFrac)/Gap);
     Fail:=0;

For N:=increments downto 1 do begin
     Perim[N]:=0;
     Failed[N]:=False;
     Fraction:=LowFrac+(N*Gap);
     Steplength[N]:=Fraction * FeretMax;

     {Determine Starting posn. and rotation direction}
     FirstY:=FindTop(Steplength[N],WalkDir);
     Failed[N]:=FirstY=Fl;
     Case WalkDir of
     'R' : begin
           FirstX:=LWall; Swing:=1;
           OutTextXY(450,420,'Left to Right');
           end;
     'L' : begin
           FirstX:=RWall; Swing:=-1;
           OutTextXY(450,420,'Right to Left');
           end;
     end;

     SetColor(Green);
     OutTextXY(560,95,R2S(Fraction,8,3));
     OutTextXY(560,115,R2S(Steplength[N],7,2));
     SetColor(LightBlue);
     StartX:=FirstX; StartY:=FirstY;
     StartAng:=-90;
     Next:=False;

     repeat
       StartAng:=StartAng-Swing;
       Attempts:=0;
       repeat
         Inc(Attempts);
         StartAng:=StartAng+Swing;
         posn:=StartAng/DegRad;
         EndX:=round(StartX+Steplength[N]*Cos(Posn));
         EndY:=round(StartY+Steplength[N]*Sin(Posn));
         Test:=GetPixel(EndX,EndY);
         if StartAng<0 then StartAng:=StartAng+360;
         if WalkDir='R' then Next:=(test=black) and (EndX>=LWall)
            else Next:=(test=black) and (EndX<=RWall);
         Failed[N]:=Attempts>360
         until Next or Failed[N];              {Ensures starting from Black}

       Next:=False; Angle:=StartAng;

       Attempts:=0;
       repeat
         Inc(Attempts);
         Angle:=Angle+Swing;
         posn:=Angle/DegRad;
         EndX:=round(StartX+Steplength[N]*Cos(Posn));
         EndY:=round(StartY+Steplength[N]*Sin(Posn));
         Test:=GetPixel(EndX,EndY);
```

```
          if WalkDir='R' then Next:=(test=blue) or (EndX>=RWall)
             else Next:=(test=blue) or (EndX<=LWall);
          Failed[N]:=Attempts>360
          until Next or Failed[N];

        If Debug='On' then Delay(wait);
        Case Walkdir of
             'R' : Checkdist:=RWall-EndX;
             'L' : Checkdist:=EndX-LWall;
             end;
        if (Range='No') then line(StartX,StartY,EndX,EndY);
        StartX:=EndX; StartY:=EndY;
        StartAng:=Angle-(Swing*90);
        If StartAng<0 then StartAng:=StartAng+360;
        Perim[N]:=Perim[N]+Steplength[N];
        if (Range='Yes') and (Failed[N]=False) then
Failed[N]:=Perim[N]>12*FeretMax;

        until (CheckDist<=steplength[N])or (Failed[N]);

      If Failed[N] then begin
         Inc(Fail);
         OutTextXY(510,160+12*Fail,'Failed @ '+R2S(Fraction,5,3));
         FailTxt:=FailTxt+R2S(Fraction,5,3)+' ';
         end;

      Perim[N]:=Perim[N]+Checkdist;

      if Range='No' then Case WalkDir of
      'R' : line(EndX,EndY,RWall,EndY);
      'L' : line(EndX,EndY,LWall,EndY);
      end;

      SetColor(Green);
      DelText(560,95,R2S(Fraction,8,3));
      DelText(560,115,R2S(Steplength[N],7,2));
      DelText(560,135,R2S(Perim[N+1],7,2));
      OutTextXY(560,135,R2S(Perim[N],7,2));
end;                                          {END N:=1 to increments}

      OutTextXY(560,95,R2S(Fraction,8,3));
      OutTextXY(560,115,R2S(Steplength[N],7,2));
      OutTextXY(560,135,R2S(Perim[N],7,2));
      if Range='Yes' then LowFrac:=LowFrac+Gap;
      SetTextJustify(CenterText,TopText);
      DelText(554,10,'Analyzing...');

      if (not multi) or (Debug='On') then Continue;

      if Range='Yes' then begin
      J:=1;
      for N:=1 to Increments do begin
          if not failed[N] then begin
             Steplength[J]:=ln(Steplength[N]);
             Perim[J]:=ln(Perim[N]);
             Inc(J);
             end;
          end;
      Increments:=Increments-Fail;
      GraphIt(Increments,1,'Structured Walk','ln Steplength','ln Perimeter',
                           Steplength,Perim);
```

```
     ResultBox;

     end;

     if (Dimensions=3) then begin
        grad_av:=grad_av+linb;
        r_av:=r_av+r;
        end;

     if FailTxt<>'' then Failtxt:=Failtxt+' : ';

     until Turn=2;

     if (Dimensions=3) then begin
        linb:=grad_av/3;
        r:=r_av/3;
        end;

end;                                {End 'Steps'}
{***************************************************************************}
procedure FibreWalk;

var
     Fail, N, TopLeft, TopRight, StartX, StartY, FirstX, FirstY : integer;
     EndX, EndY : longint;
     Steps,Attempts,StartAng, Angle, Swing : integer;
     Posn, Checkdist : single;
     Test : Word;
     Next : Boolean;
     Failed : array [1..35] of boolean;
     I, Pix, Code : integer;
     Fig : char;

begin
     InfoBox;

     {Find top left point}
     Assign(Storage,save);
     Reset(Storage);
     for J:=1 to TB do Readln(Storage);           {Miss Top Border}
     J:=0;
     repeat
           for K:=1 to LB do read(Storage,Fig);   {Miss Left Border}
           read(Storage,Fig);
           Val(Fig,Pix,code);
           Inc(J);
           Readln(Storage);
           until Pix=1;
     Close(Storage);
     TopLeft:=J+ExtraY+TB;

     {Find top right point}
     Assign(Storage,save);
     Reset(Storage);
     for J:=1 to TB do Readln(Storage);           {Miss Top Border}
     J:=0;
     repeat
           for K:=LB+1 to (WImage-RB-1) do begin    {Reads unitl RB}
               read(Storage,Fig);
               end;
           read(Storage,Fig);
```

```
            Val(Fig,Pix,code);
            Inc(J);
            Readln(Storage);
            until Pix=1;
      Close(Storage);
      TopRight:=J+ExtraY+TB;

      if WalkDir='R' then FirstY:=TopLeft else FirstY:=TopRight;
          if debug='On' then begin
         {Draws Width across filter 'cell' just above highest particle}
         SetLineStyle(1,0,1);
         SetColor(LightRed);
         Line(LB+ExtraX,TB+ExtraY,(WImage-RB)+ExtraX,TB+ExtraY);
         Line(LB+ExtraX,TB+ExtraY,LB+ExtraX,(HImage-BB)+ExtraY);
         Line(LB+ExtraX,(HImage-BB)+ExtraY,(WImage-RB)+ExtraX,(HImage-
BB)+ExtraY);
         Line((WImage-RB)+ExtraX,(HImage-BB)+ExtraY,(WImage-
RB)+ExtraX,TB+ExtraY);
         SetLineStyle(0,0,1);
         end;

      FeretMax:=WImage;
      OutTextXY(560,75,R2S(FeretMax,7,2));

      {Define step length as fraction of Feret Diameter}
      if Range='Yes' then LowFrac:=LowFrac-Gap;

      Increments:=round((UppFrac-LowFrac)/Gap);

      {Determine Starting posn. and rotation direction}
      Case WalkDir of
      'R' : begin
            FirstX:=LB+ExtraX; Swing:=1;
            end;
      'L' : begin
            FirstX:=(WImage-RB)+ExtraX; Swing:=-1;
            end;
      end;

      Fail:=0;

For N:=increments downto 1 do begin
      Perim[N]:=0;
      Failed[N]:=False;
      Fraction:=LowFrac+(N*Gap);
      Steplength[N]:=Fraction * FeretMax;

      SetColor(Green);
      OutTextXY(560,95,R2S(Fraction,8,3));
      OutTextXY(560,115,R2S(Steplength[N],7,2));

      SetColor(LightBlue);

      StartX:=FirstX; StartY:=FirstY;
      StartAng:=-90;

      Next:=False;
      Steps:=0;

      repeat
        if Steps=1 then StartAng:=StartAng-Swing
        else StartAng:=StartAng+Swing;
        Attempts:=0;
        repeat
          Inc(Attempts);
          if Steps=1 then StartAng:=StartAng+Swing
             else StartAng:=StartAng-Swing;
          posn:=StartAng/DegRad;
          EndX:=round(StartX+Steplength[N]*Cos(Posn));
          EndY:=round(StartY+Steplength[N]*Sin(Posn));
          Test:=GetPixel(EndX,EndY);
          if StartAng<0 then StartAng:=StartAng+360;
          if WalkDir='R' then Next:=(test=black) and (EndX>=LB)
             else Next:=(test=black) and (EndX<=(WImage-RB));
          Failed[N]:=Attempts>360
          until Next or Failed[N];              {Ensures starting from Black}

        Next:=False; Angle:=StartAng;

        Attempts:=0;
        repeat
          Inc(Attempts);
          Angle:=Angle+Swing;
          posn:=Angle/DegRad;
          EndX:=round(StartX+Steplength[N]*Cos(Posn));
          EndY:=round(StartY+Steplength[N]*Sin(Posn));
          Test:=GetPixel(EndX,EndY);
          if WalkDir='R' then Next:=(test=blue) or (EndX>=(WImage-RB))
             else Next:=(test=blue) or (EndX<=LB);
          Failed[N]:=Attempts>360
          until Next or Failed[N];

        Inc(Steps);

        If Debug='On' then Delay(wait);

        Case Walkdir of
             'R' : Checkdist:=(WImage-RB+ExtraX)-EndX;
             'L' : Checkdist:=EndX-(LB+ExtraX);
             end;

        if (Range='No') then line(StartX,StartY,EndX,EndY);
        StartX:=EndX; StartY:=EndY;
        StartAng:=Angle-(Swing*90);
        If StartAng<0 then StartAng:=StartAng+360;
        Perim[N]:=Perim[N]+Steplength[N];
        if (Range='Yes') and (Failed[N]=False) then
Failed[N]:=Perim[N]>5*FeretMax;

        until (CheckDist<=steplength[N])or (Failed[N]);

      If Failed[N] then begin
         Inc(Fail);
         OutTextXY(510,160+12*Fail,'Failed @ '+R2S(Fraction,5,3));
         FailTxt:=FailTxt+R2S(Fraction,5,3)+' ';
         end;

      Perim[N]:=Perim[N]+Checkdist;

      if Range='No' then Case WalkDir of
      'R' : line(EndX,EndY,(WImage-RB)+ExtraX,EndY);
      'L' : line(EndX,EndY,LB+ExtraX,EndY);
      end;

      SetColor(Green);
      DelText(560,95,R2S(Fraction,8,3));
      DelText(560,115,R2S(Steplength[N],7,2));
      DelText(560,135,R2S(Perim[N+1],7,2));
      OutTextXY(560,135,R2S(Perim[N],7,2));
end;                                        {END N:=1 to increments}

      OutTextXY(560,95,R2S(Fraction,8,3));
      OutTextXY(560,115,R2S(Steplength[N],7,2));
      OutTextXY(560,135,R2S(Perim[N],7,2));
      if Range='Yes' then LowFrac:=LowFrac+Gap;
      SetTextJustify(CenterText,TopText);
      DelText(554,10,'Analyzing...');

      if (not multi) or (Debug='On') then Continue;

      if Range='Yes' then begin
      J:=1;
      for N:=1 to Increments do begin
          if not failed[N] then begin
             Steplength[J]:=ln(Steplength[N]);
             Perim[J]:=ln(Perim[N]);
             Inc(J);
             end;
          end;
      Increments:=Increments-Fail;
      GraphIt(Increments,1,'Structured Walk','ln Steplength','ln Perimeter',
                          Steplength,Perim);
      ResultBox;
      end;
end;                                {End 'Steps'}
{**************************************************************************}
procedure AggWalk(OutX, OutY, Size : integer);
begin

      Edge:=Sin(45/DegRad);         {Allows step to start from EDGE of particle}

      Particle(OutX,OutY,2,Red);
      Cross(CentreX,CentreY);

      {Choose start position}
      Right:=(OutX>=CentreX);
      Top:=(OutY<=CentreY);

      Steps(OutX, OutY, Size);
      number:=number-1;             {Resets number for building}
end;
{**************************************************************************}
procedure AggFerets;

var
   Distance : extended;
   Xrad, Yrad : single;
   angle, side, posn : integer;
   X1, X2, X3, Y1, Y2, Y3 : comp;
   a : integer;
   b,c,d : extended;
   good : array [0..1] of integer;
   Xmin, Ymin, Xmax, Ymax : array [0..1] of integer;
   perpdist : extended;
   perpmin : array [0..1] of extended;
   Xf, Yf : array [0..1,0..1] of integer;
   MaxAngle, MinAngle : Integer;
   Space : array [0..1,0..1] of extended;

const
   Tangent=200;

begin
   a:=2*Tangent;
   maxdist:=0;
   FeretMax:=0;
   FeretMin:=2*radius;
   InfoBox;

   for I:= 2 to Number do begin        {Calculate outermost particle}
       Xdist:=(PosX[I]-PosX[1]); YDist:=(PosY[I]-PosY[1]);
       Distance:=Sqrt(sqr(Xdist)+sqr(Ydist)) + Size[I];
       If distance > maxdist then begin
          maxdist:=round(distance);
          outer:=I;
          end;
       end;
   maxdist:=maxdist+10;

   for angle:=0 to 179 do begin        {Calculate Feret diameters}
       for side:=0 to 1 do begin
           perpmin[side]:=2*radius;
           posn:=(angle-180*side);
           arc(CentreX,CentreY,-(posn+2),-posn,maxdist);
           If Debug='On' then delay(wait);
           Xrad:=CentreX+maxdist*Cos(posn/DegRad);
           Yrad:=CentreY+maxdist*Sin(posn/DegRad);
           Xf[side,0]:=round(Xrad+Tangent*Cos(Pi/2-Posn/DegRad));
           Yf[side,0]:=round(Yrad-Tangent*Sin(Pi/2-Posn/DegRad));
           Xf[side,1]:=round(Xrad-Tangent*Cos(Pi/2-Posn/DegRad));
           Yf[side,1]:=round(Yrad+Tangent*Sin(Pi/2-Posn/DegRad));
           for J:=1 to Number do begin
               X2:=PosX[J]-Xf[side,0]; Y2:=PosY[J]-Yf[side,0];
               b:=sqrt(sqr(X2)+sqr(Y2));

               X3:=PosX[J]-Xf[side,1]; Y3:=PosY[J]-Yf[side,1];
               c:=sqrt(sqr(X3)+sqr(Y3));

               d:=0.5*(a+b+c);
               perpdist:=(2/a)*sqrt(abs(d*(d-a)*(d-b)*(d-c)));
               perpdist:=perpdist-size[J];

               if (perpdist < perpmin[side]) then begin
                  perpmin[side]:=perpdist;
                  good[side]:=J;
                  end;
               end;        {ends 'J' loop}

       end;            {ends 'side' loop}

       Feret:=2 * MaxDist - perpmin[0] - perpmin[1];

       If Feret < FeretMin then begin
          FeretMin:=Feret;
```

```
            Xmin[0]:=PosX[good[0]]; Xmin[1]:=PosX[good[1]];
            Ymin[0]:=PosY[good[0]]; Ymin[1]:=PosY[good[1]];
            MinAngle:=Angle;
            Space[0,0]:=Perpmin[0]; Space[0,1]:=Perpmin[1];
            end;

         If Feret > FeretMax then begin
            SetColor(Green);
            FeretMax:=Feret;
            Xmax[0]:=PosX[good[0]]; Xmax[1]:=PosX[good[1]];
            Ymax[0]:=PosY[good[0]]; Ymax[1]:=PosY[good[1]];
            MaxAngle:=Angle;
            Space[1,0]:=Perpmin[0]; Space[1,1]:=Perpmin[1];
            end;

         end;                          {ends 'angle' loop}

{Draw Minimum Feret Diameter if debugging is 'On'}
      if debug='On' then for side:=0 to 1 do begin
         SetLineStyle(2,0,1);
         SetColor(lightBlue);
         posn:=(MinAngle-180*side);
         Xrad:=CentreX+(maxdist-Space[0,side])*Cos(posn/DegRad);
         Yrad:=CentreY+(maxdist-Space[0,side])*Sin(posn/DegRad);
         Xf[side,0]:=round(Xrad+Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,0]:=round(Yrad-Tangent*Sin(Pi/2-Posn/DegRad));
         Xf[side,1]:=round(Xrad-Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,1]:=round(Yrad+Tangent*Sin(Pi/2-Posn/DegRad));
         line(Xf[side,0],Yf[side,0],Xf[side,1],Yf[side,1]);
         end;
      OutTextXY(560,55,R2S(FeretMin,7,2));

{Draw Maximum Feret Diameter if debugging is 'On'}
      if debug='On' then for side:=0 to 1 do begin
         SetLineStyle(1,0,1);
         SetColor(LightRed);
         posn:=(MaxAngle-180*side);
         Xrad:=CentreX+(maxdist-Space[1,side])*Cos(posn/DegRad);
         Yrad:=CentreY+(maxdist-Space[1,side])*Sin(posn/DegRad);
         Xf[side,0]:=round(Xrad+Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,0]:=round(Yrad-Tangent*Sin(Pi/2-Posn/DegRad));
         Xf[side,1]:=round(Xrad-Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,1]:=round(Yrad+Tangent*Sin(Pi/2-Posn/DegRad));
         line(Xf[side,0],Yf[side,0],Xf[side,1],Yf[side,1]);
         end;
      OutTextXY(560,75,R2S(FeretMax,7,2));

      SetColor(Green);
      SetLineStyle(SolidLn,0,NormWidth);
      line(Xmax[0],Ymax[0],Xmax[1],Ymax[1]);
      line(Xmin[0],Ymin[0],Xmin[1],Ymin[1]);
end;
{*************************************************************************}
procedure SEMWalk(OutX, OutY : integer);

begin
     InfoBox;

     {Choose start position}
     Right:=(FurthX>CofGX);
     Top:=(FurthY<=CentreY);
     Steps(OutX, OutY,0);
end;                              {End 'Steps'}
{****************************************************************************}
procedure PoreWalk(OutX, OutY : integer);

begin
     InfoBox;

     {Choose start position}
     Right:=(FurthX>CofGX);
     Top:=(FurthY<=CentreY);

     Steps(OutX, OutY,0);
end;                              {End 'Steps'}
{****************************************************************************}
procedure PoreFerets;

var
   Look : File;
   FerStore : Text;
   X1, X2, X3, Y1, Y2, Y3 : comp;
   X,Y, maxdist, angle, side, posn : integer;
   Tangent, a, MaxAngle, MinAngle : integer;
   CenX, CenY, K : Integer;
   Rad, Xrad, Yrad : single;
   Distance : extended;
   Xmin, Ymin, Xmax, Ymax : array [0..1] of integer;
   GoodX, GoodY : array [0..1] of integer;
   Xf, Yf : array [0..1,0..1] of integer;
   Space : array [0..1,0..1] of extended;
   PerpMin, RadMin : array [0..1] of single;
   Fig : char;

label FileRead;
label NotThere;

begin
   MaxDist:=0;

   end;
{****************************************************************************}
procedure SEMFerets;

var
   Data : String;
   FerStore : Text;
   X1, X2, X3, Y1, Y2, Y3 : comp;
   X,Y, angle, side, posn : integer;
   Tangent, a, MaxAngle, MinAngle : integer;
   CenX, CenY, K : Integer;
   Rad, Xrad, Yrad : single;
   Xmin, Ymin, Xmax, Ymax : array [0..1] of integer;
   GoodX, GoodY : array [0..1] of integer;
   Xf, Yf : array [0..1,0..1] of integer;
   Space : array [0..1,0..1] of extended;
   PerpMin, RadMin : array[0..1] of single;
   Fig : char;
   Exists : boolean;

label
```

```
      FileRead, NotThere;

begin
   InfoBox;

   Data:=RemLet(Save,4)+'.fer';
   Assign(Look,Data);
   {$I-}
   Reset(Look);
   Close(Look);
   Exists := (IOResult = 0);
   {$I+}

   MaxDist:=0;
   Reset(Storage);
     for J:=1 to TB do Readln(Storage);          {Miss Top Border}
     for J:=TB to (HImage-BB) do begin           {Reads until BB}
         for K:=1 to LB do read(Storage,Fig);    {Miss Left Border}
         for K:=LB+1 to (WImage-RB) do begin     {Reads unitl RB}
             if GetPix=1 then begin
                Distance:=Sqrt(sqr(K-CofGX)+sqr(J-CofGY));
                if Distance>MaxDist then begin
                   MaxDist:=round(Distance);
                   FurthX:=K; FurthY:=J;
                   end;
                end;
         end;
         readln(Storage);
         end;
         close(storage);

   CenX:=round(CofGX+ExtraX); CenY:=round(CofGY+ExtraY);
   Cross(CenX,CenY);

   If Exists then goto FileRead;

   DelText(560,55,R2S(FeretMin,7,2));
   DelText(560,75,R2S(FeretMax,7,2));

   FeretMax:=0;
   FeretMin:=2*radius;

   PutPixel(FurthX+ExtraX,FurthY+ExtraY,LightGreen);
   Maxdist:=MaxDist+5;
   Tangent:=round(MaxDist*1.5);
   a:=2*Tangent;

   SetColor(Green);

   for Angle:=0 to 179 do begin          {Calculate Feret diameters}
       Rad:=MaxDist;
       for Side:=0 to 1 do begin {repeat}
           PerpMin[side]:=2*radius;
           RadMin[side]:=2*MaxDist;
           Posn:=(angle-180*side);
           arc(CenX,CenY,-(posn+2),-posn,maxdist);
           If Debug='On' then delay(wait);
           Xrad:=CenX+Rad*Cos(posn/DegRad);
           Yrad:=CenY+Rad*Sin(posn/DegRad);
           Xf[side,0]:=round(Xrad+Tangent*Cos(Pi/2-Posn/DegRad));
           Yf[side,0]:=round(Yrad-Tangent*Sin(Pi/2-Posn/DegRad));
```

```
        Xf[side,1]:=round(Xrad-Tangent*Cos(Pi/2-Posn/DegRad));
        Yf[side,1]:=round(Yrad+Tangent*Sin(Pi/2-Posn/DegRad));

Assign(Storage,save);
Reset(Storage);
for J:=1 to TB do Readln(Storage);          {Miss Top Border}
for J:=TB+1 to (HImage-BB) do begin         {Reads until BB}
    for K:=1 to LB do read(Storage,Fig);    {Miss Left Border}
    for K:=LB+1 to (WImage-RB) do begin     {Reads unitl RB}
        if GetPix=1 then begin
            X2:=K+ExtraX-Xf[side,0]; Y2:=J+ExtraY-Yf[side,0];
            b:=sqrt(sqr(X2)+sqr(Y2));
            X3:=K+ExtraX-Xf[side,1]; Y3:=J+ExtraY-Yf[side,1];
            c:=sqrt(sqr(X3)+sqr(Y3));
            d:=0.5*(a+b+c);
            perpdist:=(2/a)*sqrt(abs(d*(d-a)*(d-b)*(d-c)));
            if (perpdist < perpmin[side]) then begin
               perpmin[side]:=perpdist;
               GoodX[side]:=K+ExtraX; GoodY[side]:=J+ExtraY;
               end;
            end;        {Ends Pix=1 loop}
        end;            {Ends K loop}

    Readln(Storage);
    end;                {Ends J loop}
    Close(Storage);

    end;            {ends 'side' loop}

    RadMin[0]:=MaxDist-PerpMin[0];
    RadMin[1]:=MaxDist-PerpMin[1];

    Feret:=RadMin[0] + RadMin[1];

    If Feret < FeretMin then begin
       FeretMin:=Feret;
       Xmin[0]:=GoodX[0]; Xmin[1]:=GoodX[1];
       Ymin[0]:=GoodY[0]; Ymin[1]:=GoodY[1];
       MinAngle:=Angle;
       Space[0,0]:=RadMin[0]; Space[0,1]:=RadMin[1];
       end;

    If Feret > FeretMax then begin
       FeretMax:=Feret;
       Xmax[0]:=GoodX[0]; Xmax[1]:=GoodX[1];
       Ymax[0]:=GoodY[0]; Ymax[1]:=GoodY[1];
       MaxAngle:=Angle;
       Space[1,0]:=RadMin[0]; Space[1,1]:=RadMin[1];
       end;
    end;                    {ends 'angle' loop}

    If not Exists then goto NotThere;

    FileRead:
    DelText(560,55,R2S(FeretMin,7,2));
    DelText(560,75,R2S(FeretMax,7,2));
    Assign(FerStore,Data);
    Reset(FerStore);
    Readln(FerStore,MaxDist);
    Read(FerStore,FeretMax);
    Read(FerStore,FeretMin);
```

```
        Read(FerStore,MaxAngle);
        Read(FerStore,MinAngle);
        for J:=0 to 1 do begin
            Readln(FerStore,XMax[J]);
            Readln(FerStore,XMin[J]);
            Readln(FerStore,YMax[J]);
            Readln(FerStore,YMin[J]);
            end;
        for K:=0 to 1 do
        for J:=0 to 1 do Read(FerStore,Space[K,J]);
        Read(FerStore,FurthX);
        Read(FerStore,FurthY);
        Close(FerStore);
        Tangent:=round(MaxDist*1.5);
        SetColor(Green);
        Circle(CenX,CenY,MaxDist);
        PutPixel(FurthX+ExtraX,FurthY+ExtraY,LightGreen);

        NotThere:

{Draw Minimum Feret Diameter if debugging is 'On'}
      if debug='On' then for side:=0 to 1 do begin
         SetLineStyle(2,0,1);
         SetColor(lightBlue);
         posn:=(MinAngle-180*side);
         Xrad:=CenX+(Space[0,side])*Cos(posn/DegRad);
         Yrad:=CenY+(Space[0,side])*Sin(posn/DegRad);
         Xf[side,0]:=round(Xrad+Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,0]:=round(Yrad-Tangent*Sin(Pi/2-Posn/DegRad));
         Xf[side,1]:=round(Xrad-Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,1]:=round(Yrad+Tangent*Sin(Pi/2-Posn/DegRad));
         line(Xf[side,0],Yf[side,0],Xf[side,1],Yf[side,1]);
         end;
      OutTextXY(560,55,R2S(FeretMin,7,2));

{Draw Maximum Feret Diameter if debugging is 'On'}
      if debug='On' then for side:=0 to 1 do begin
         SetLineStyle(1,0,1);
         SetColor(LightRed);
         posn:=(MaxAngle-180*side);
         Xrad:=CenX+(Space[1,side])*Cos(posn/DegRad);
         Yrad:=CenY+(Space[1,side])*Sin(posn/DegRad);
         Xf[side,0]:=round(Xrad+Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,0]:=round(Yrad-Tangent*Sin(Pi/2-Posn/DegRad));
         Xf[side,1]:=round(Xrad-Tangent*Cos(Pi/2-Posn/DegRad));
         Yf[side,1]:=round(Yrad+Tangent*Sin(Pi/2-Posn/DegRad));
         line(Xf[side,0],Yf[side,0],Xf[side,1],Yf[side,1]);
         end;
      OutTextXY(560,75,R2S(FeretMax,7,2));

      SetColor(Green);
      SetLineStyle(SolidLn,0,NormWidth);
      line(Xmax[0],Ymax[0],Xmax[1],Ymax[1]);
      line(Xmin[0],Ymin[0],Xmin[1],Ymin[1]);

      Data:=RemLet(Save,4)+'.fer';

      Assign(FerStore,Data);
      Rewrite(FerStore);
      Writeln(FerStore,MaxDist);
      Writeln(FerStore,R2S(FeretMax,7,3));
```

```
        Writeln(FerStore,R2S(FeretMin,7,3));
        Writeln(FerStore,MaxAngle);
        Writeln(FerStore,MinAngle);
        for J:=0 to 1 do begin
            Writeln(FerStore,XMax[J]);
            Writeln(FerStore,XMin[J]);
            Writeln(FerStore,YMax[J]);
            Writeln(FerStore,YMin[J]);
            end;
        for K:=0 to 1 do
        for J:=0 to 1 do Writeln(FerStore,R2S(Space[K,J],7,3));
        Writeln(FerStore,FurthX);
        Writeln(FerStore,FurthY);
        Close(FerStore);
  end;
  {***************************************************************************}
  end.
```

VARIABLE.PAS - Contains all the variable names to shared globally within program

```
unit variables;

interface

type
    info = array [1..2000] of integer;
    check = array [1..2000] of boolean;

var
   {System Defaults}
   Root, GraphDir, Path, Name, Save, LongSave, POV, GYR,
   Numgen, Debug, Range, Auto, Twist : string;
   OverWrite, Return : Boolean;

                          {Switches / values for:
   File names, random NUMbers, visual DEBUGging, RANGE of steplengths,
              AUTOmatic modelling, 3-D TWISTing}

   Wait : integer;                {Time Delay}
   Model : char;                  {'B'allistic, 'M'ixed or 'D'iffusion}
   Diff_Frac : single;            {DIFFusive_FRACtion}
   I, J, K, N, outer : integer;

   {Basic Agglomerate Information}
   Shape, Quality : String;       {Circle/Poly;  Regular/Irreg.}
   Sides,Rotate : integer;
   Dimensions, Runs, Number, Seed, MinSize, MaxSize : integer;
   Radius : integer;
   PosX, PosY, PosZ, Size : info;

   {Agglomerate Building}
   RealRad : single;
   AngleIn, AngleOut : single;
   Slice_In, Slice_Out : single;    {Size of circle at depth 'z'}
   Xin, Xout, Yin, Yout, Zin, Zout : longint;
   XDist,YDist,Zdist : single;
   Xstart, Ystart, Zstart : integer;
   MaxDiffMoves, MaxBalMoves : integer;
   Combined : integer;              {Combined radii of particles}
   Where : single;                  {Distance of particle form centre}
   a,b,c,d,perpdist : single;       {Perp. dist. const.}
   Hit : check;
   NotFloor, Floor : boolean;
   Flag : boolean;

   {Monte Carlo Simulation Parameters}
   SimInfo : string;
   DownProb,StickProb : integer;
   CellCen, CellRad, Fl,Roof,LWall,RWall : integer;
   Condition : char;
   W,Beta : Single;               {Parameters (3-D Monte Carlo}
   Touch : integer;               {Number of other particles in
                                  contact with rolling particle}
   Rest : boolean;                {Is particel at REST ?}
   WasTouching : integer;

   {Agglomerate Analysis}
   RData,Stem,ParentName,PoreStore,PoreFile,PoreName,PoreStem,Remember :
string;
```

```
   Method, PoreMeth, EncMeth, PropMeth, FileType, SubType : Char;
   Feret, FeretMin, FeretMax : single;
   View : char;                {Aspect of agglomerate}
   Steplength, Perim : array [1..35] of single;
   Fraction, LowFrac, UppFrac, OldLow, OldUpp, Gap : single;
   WalkDir : char;             {Direction of filt. struc.walk}
   Increments : integer;
   Used : integer;             {Previous USE of multiple steplengths}
   lina,linb,r : single;             {Linear regression constants}
   AvPore : single;            {Average Porosity}
   CofGX, CofGY, CofGZ : single;   {Centre of Gravity}
   MaxDist, Ripple : integer;        {Number of points for enclosing circle}
   Conf : single;
   Convert : Integer;
   Aborted, InvPore, PoreMeasure, Multi, PoreMulti, NewPore : Boolean;
   PoreArea : integer;
   NumFiles : integer;         {Number of files to automatically check}
   BestFit : string;           {Highest confidence of fit}

   {SEM Images}
   TB,BB,LB,RB : integer;         {Values for image borders}
   FurthX,FurthY : integer;       {Outermost pixels}
   HImage, WImage : integer;      {Height and width of image}
   PixNum : extended;             {Total number of pixels - CofG}
   MaxRad : integer;              {Maximum radius for Enc. Circle}
   ExtraX, ExtraY : integer;      {Extra space needed at top}

   {Ray Tracing Settings}
   RayHeight,RayWidth,Frames : integer;

   {Random number / Time log}
   hr, mn, sc : string;
   h,m,s,hund : word;
   GoTime, EndTime : double;
   Time : single;

   {File Information / Headers}
   Title, DimTxt, TimeTxt, TimeStr : String;
   Storage, Instruct, Gyrfile, Keep, Local, Other, Network, DataFile,
Graphfile : text;
   NoDrivers, NetThere, LocalThere, Otherthere : boolean;
   Created : boolean;          {Flag for having created sub-directories}
   Code : integer;

   {Pascal Defaults}
   GraphDriver, GraphMode : Integer;

   {Basic Functions}
   Col : integer;         {Colour short-cut / Number of menu choices}
   yesno, Option : char;         {Yes/No trapper / Key catch}

const
     DegRad=180/Pi;                {Converts angles from degrees to radians}
     CentreX=240; CentreY=240; CentreZ=240;

implementation

begin
Radius:=240;
end.
```

APPENDIX B

Summary of simulations

APPENDIX B – SUMMARY OF SIMULATIONS

This appendix shows an overview of the simulations (both seed agglomerate and filtration simulations) created for the project. It also shows the type of fractal and physical measurements applied to the various structures.

The number of particles in each simulation was changed to match the space-filling properties (on the screen) of the simulations, with the filtration simulations occupying less space per particle than the seed agglomeration simulations, and the three-dimensional filtration simulations occupying less space than the two-dimensional simulations.

Seed agglomerates

- Two dimensions

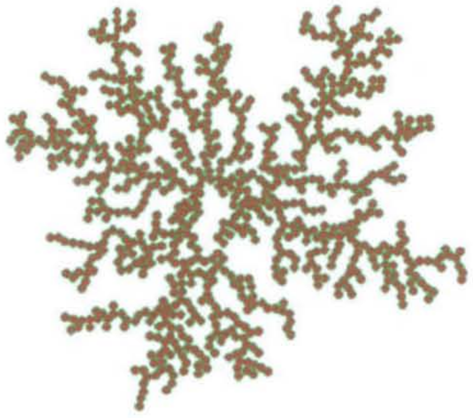


Figure B.1 - Example of 2-D seed agglomeration

Parameter investigated:	Diffusion influence
Step change of parameter:	0% to 10% in 5% steps
Number of particles:	800
Size range:	3 to 3 (mono-dispersed) pixel radius
Number of agglomerates for each step:	40
Total number of structures:	40 x 21 = 840

Measurements performed, with methods:

Perimeter roughness (fractal)	Structured walk

Density (fractal)	Enclosing circle, Radius of gyration
Porosity (physical, 2-D)	% 'Empty space' analysis

Total number of analyses: 3360

- Three dimensions

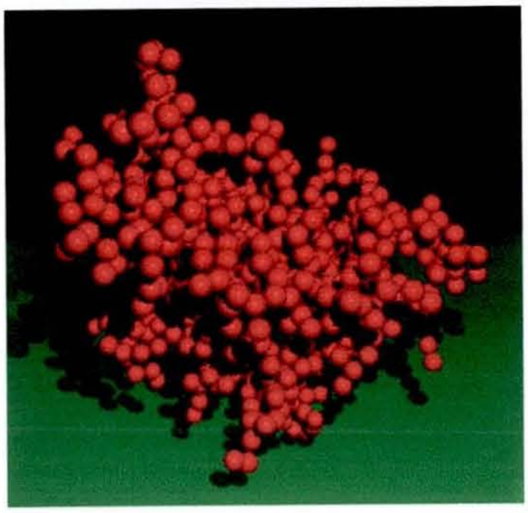


Figure B.2 – Example of 3-D seed agglomeration

Parameter investigated:	Diffusion influence
Step change of parameter:	0% to 10% in 5% steps
Number of particles:	800
Size range:	3 to 3 (mono-dispersed) pixel radius
Number of agglomerates for each step:	40
Total number of structures:	40 x 21 = 840

Measurements performed, with methods:

Perimeter roughness (fractal)	Structured walk (on 2-D projections)
Density (fractal)	Enclosing sphere, Radius of gyration
Porosity (physical, 3-D)	% 'Empty space' analysis

Total number of analyses: 5040

Filtration simulations

- Two dimensions

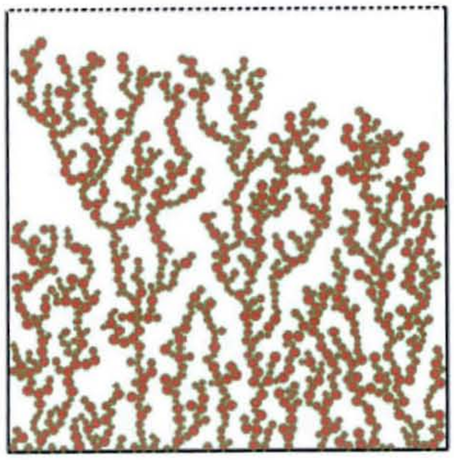


Figure B.3 – Example of 2-D filtration simulation

Parameters investigated:	Sticking probability Downward probability
Step change of parameter:	0% to 10% in 5% steps Note: The full range of both parameters was not investigated for each size distribution listed below. Initial results showed that the sticking probability had more impact on the structure characteristic than the downward probability.
Number of particles:	1000
Size ranges:	2 to 2 (mono-dispersed) pixel radius 3 to 3 (mono-dispersed) pixel radius 3 to 5 pixel radius 5 to 5 (mono-dispersed) pixel radius
Number of agglomerates for each parameter change:	20
Total number of structures:	4 x 21 x 40 = 3360

Measurements performed, with methods:

Surface roughness (fractal)	Structured walk (in two directions)
Porosity (physical, 2-D)	% 'Empty space' analysis

Total number of analyses: 10080

- Three dimensions

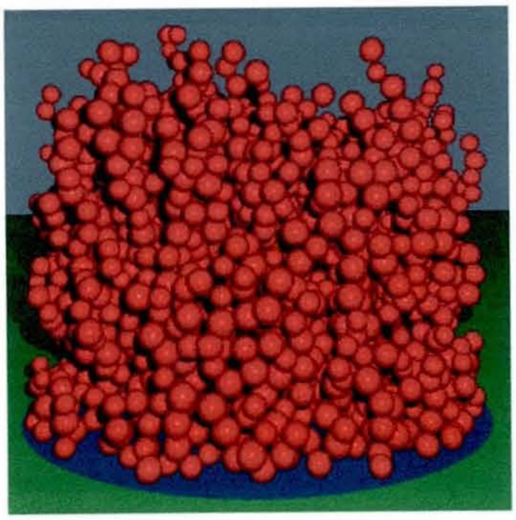


Figure B.4 - Example of 3-D filtration simulation

Parameters investigated:	Sticking probability Downward probability
Step change of parameter:	0% to 10% in 5% steps Note: The full range of both parameters was not investigated for each size distribution listed below. Initial results showed that the sticking probability had more impact on the structure characteristic than the downward probability.
Number of particles:	2000
Size ranges:	8 to 8 (mono-dispersed) pixel radius 8 to 10 pixel radius 3 to 12 pixel radius
Number of agglomerates for each parameter change:	20
Total number of structures:	3 x 21 x 40 = 2520

Measurements performed, with methods:

Surface roughness (fractal)	Structured walk (in two directions, on three 2-D projections)
Porosity (physical, 3-D)	% 'Empty space' analysis

Total number of analyses:	17640

APPENDIX C

Experimental datasheets

Appendix C – Experimental datasheets

This appendix contains the experimental datasheets from both the calcite and talc filtration tests. It also contains the Coulter® Counter particle sizing results from the two minerals.

NOTE: There are slight differences between some of the datasheets. This is due to the technique for cake characterisation evolving as the work progressed.

CALCITE

Data Sheet   Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*206.80*	**g**
Density, ρ	*1000*	*2580*	**kg m^{-3}**
volume	8.00E-04	8.02E-05	m^3

Total Mass	1.01	kg
Total Volume	8.80E-04	m^3
Solids conc.	234.96	kg m^{-3}
≡	9.11%	v/v
≡	20.54%	w/w
Viscosity, μ	*0.001*	**Pa s**
Diameter	*10.4*	**cm**
Area	8.49E-03	m^2
Pressure	*1*	**bar**
	100000	Pa
Slope of plot	2.78E+08	
Intercept	1.40E+05	
R^2	9.91E-01	

α	***1.707E+10***	**m kg^{-1}**
R$_m$	***1.19E+11***	**m^{-1}**

Data Sheet　　　　Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*516.10*	g
Density, ρ	*1000*	*2580*	$kg\ m^{-3}$
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc.	516.08	$kg\ m^{-3}$
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1*	bar
	100000	Pa
Slope of plot	1.01E+09	
Intercept	1.17E+05	
R^2	1.00E+00	

α	***2.822E+10***	$m\ kg^{-1}$
R_m	***9.901E+10***	m^{-1}

1BAR_02.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.6*	g
Density, ρ	*1000*	*2580*	$kg\ m^{-3}$
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1047.64	$kg\ m^{-3}$
≡	20.02%	v/v
≡	39.24%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1*	bar
	100000	Pa
Slope of plot	6.39E+08	
Intercept	7.47E+04	
R^2	9.98E-01	
α	***8.805E+09***	$m\ kg^{-1}$
R_m	***6.348E+10***	m^{-1}

Mass of *wet* cake	*85.84*	g
Mass of *dry* cake	*53.85*	g
Mass of liquid	31.99	g
Volume of solid	2.09E-05	m^3
Volume of liquid	3.20E-05	m^3
Cake conc^n	39.48%	
ε	0.6052	
Moisture ratio, m	1.5941	
Effective c	1047.64	$kg\ m^{-3}$
Slurry c	645.75	$kg\ m^{-3}$

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

1BAR_03.XLS

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.0*	g
Density, ρ	*1000*	*2580*	$kg\ m^{-3}$
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1065.74	$kg\ m^{-3}$
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1*	bar
	100000	Pa
Slope of plot	1.09E+09	
Intercept	1.13E+05	
R^2	9.98E-01	

α	***1.474E+10***	$m\ kg^{-1}$
R_m	***9.557E+10***	m^{-1}

Mass of *wet* cake	*67.03*	g
Mass of *dry* cake	*41.58*	g
Mass of liquid	25.45	g
Volume of solid	1.61E-05	m^3
Volume of liquid	2.55E-05	m^3
Cake conc^n	38.77%	
ε	0.6123	
Moisture ratio, m	1.6121	
Effective c	1065.74	$kg\ m^{-3}$
Slurry c	645.00	$kg\ m^{-3}$

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

1BAR_04.XLS

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	$kg\ m^{-3}$
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	977.71	$kg\ m^{-3}$
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1*	bar
	100000	Pa
Slope of plot	8.62E+08	
Intercept	1.07E+05	
R^2	9.97E-01	

α	*1.272E+10*	$m\ kg^{-1}$
R_m	*9.083E+10*	m^{-1}

Mass of *wet* cake	*71.08*	g
Mass of *dry* cake	*46.54*	g
Mass of liquid	24.54	g
Volume of solid	1.80E-05	m^3
Volume of liquid	2.45E-05	m^3
Cake conc^n	42.37%	
ε	0.5763	
Moisture ratio, m	1.5273	
Effective c	977.71	$kg\ m^{-3}$
Slurry c	645.13	$kg\ m^{-3}$

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

1BAR_05.XLS

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*517.3*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.01E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1043.94	kg m^{-3}
≡	20.04%	v/v
≡	39.27%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1*	bar
	100000	Pa
Slope of plot	1.05E+09	
Intercept	7.53E+04	
R^2	9.97E-01	

α	***1.458E+10***	m kg^{-1}
R$_m$	***6.397E+10***	m^{-1}

Mass of *wet* cake	*102.40*	g
Mass of *dry* cake	*64.46*	g
Mass of liquid	37.94	g
Volume of solid	2.50E-05	m^3
Volume of liquid	3.79E-05	m^3
Cake conc^n	39.71%	
ε	0.6029	
Moisture ratio, m	1.5886	
Effective c	1043.94	kg m^{-3}
Slurry c	646.63	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*200.20*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	7.76E-05	m^3

Total Mass	1.00	kg
Total Volume	8.78E-04	m^3
Solids conc.	228.12	kg m^{-3}
≡	8.84%	v/v
≡	20.02%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1.5*	bar
	150000	Pa
Slope of plot	1.80E+08	
Intercept	5.29E+04	
R^2	9.99E-01	

α	***1.708E+10***	**m kg^{-1}**
R_m	***6.742E+10***	**m^{-1}**

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*515.60*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	906.90	kg m^{-3}
≡	19.99%	v/v
≡	39.19%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1.5*	bar
	150000	Pa
Slope of plot	8.05E+08	
Intercept	7.75E+04	
R^2	9.99E-01	

α	***1.921E+10***	m kg^{-1}
R$_m$	***9.874E+10***	m^{-1}

Mass of *wet* cake	*30.50*	g
Mass of *dry* cake	*21.05*	g
Mass of liquid	9.45	g
Volume of solid	8.16E-06	m^3
Volume of liquid	9.45E-06	m^3
Cake conc^n	46.33%	
ε	0.5367	
Moisture ratio, m	1.4489	
Effective c	906.90	kg m^{-3}
Slurry c	515.68	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

1'5B_02.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*0.80*	*0.20*	**kg**
Density, ρ	*1000*	*2580*	**kg m^{-3}**
volume	8.00E-04	7.75E-05	m^3

Total Mass	1.00	kg
Total Volume	8.78E-04	m^3
Solids conc.	227.92	kg m^{-3}
≡	8.83%	v/v
≡	20.00%	w/w
Viscosity, μ	*0.001*	**Pa s**
Diameter	*10.4*	**cm**
Area	8.49E-03	m^2
Pressure	*2*	**bar**
	200000	Pa
Slope of plot	1.16E+08	
Intercept	4.21E+04	
R^2	9.96E-01	

α	***1.474E+10***	**m kg^{-1}**
R	***7.157E+10***	**m^{-1}**

	Filtrate	Solid	
Weight	*0.80*	*0.42*	**kg**
Density, ρ	*1000*	*2580*	**kg m^{-3}**
volume	8.00E-04	1.64E-04	m^3

Total Mass	1.22	kg
Total Volume	9.64E-04	m^3
Solids conc.	439.42	kg m^{-3}
≡	17.03%	v/v
≡	34.62%	w/w
Viscosity, μ	*0.001*	**Pa s**
Diameter	*10.4*	**cm**
Area	8.49E-03	m^2
Pressure	*2*	**bar**
	200000	Pa
Slope of plot	2.54E+08	
Intercept	9.05E+04	
R^2	0.9906124	

α	***1.668E+10***	**m kg^{-1}**
R	***1.537E+11***	**m^{-1}**

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*0.80*	*0.20*	**kg**
Density, ρ	*1000*	*2580*	**kg m^{-3}**
volume	8.00E-04	7.57E-05	m^3

Total Mass	1.00	kg
Total Volume	8.76E-04	m^3
Solids conc.	223.05	kg m^{-3}
≡	8.65%	v/v
≡	19.62%	w/w
Viscosity, μ	*0.001*	**Pa s**
Diameter	*10.4*	**cm**
Area	8.49E-03	m^2
Pressure	*2*	**bar**
	200000	Pa
Slope of plot	1.46E+08	
Intercept	5.10E+04	
R^2	0.9981139	

α	***1.891E+10***	**m kg^{-1}**
R	***8.657E+10***	**m^{-1}**

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*515.20*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc.	515.36	kg m^{-3}
≡	19.98%	v/v
≡	39.17%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	4.09E+08	
Intercept	7.28E+04	
R^2	9.92E-01	

α	***2.289E+10***	**m kg^{-1}**
R_m	***1.236E+11***	**m^{-1}**

Data Sheet Material: Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.3*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1010.07	kg m^{-3}
≡	20.01%	v/v
≡	39.22%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	3.88E+08	
Intercept	7.29E+04	
R^2	9.99E-01	

α	***1.108E+10***	**m kg^{-1}**
R$_m$	***1.239E+11***	**m^{-1}**

Mass of *wet* cake	*68.85*	g
Mass of *dry* cake	*44.15*	g
Mass of liquid	24.70	g
Volume of solid	1.71E-05	m^3
Volume of liquid	2.47E-05	m^3
Cake conc^n	40.93%	
ε	0.5907	
Moisture ratio, m	1.5595	
Effective c	1010.07	**kg m^{-3}**
Slurry c	645.38	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

2BAR_06.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*520.5*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.02E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1061.70	kg m^{-3}
≡	20.14%	v/v
≡	39.42%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	3.84E+08	
Intercept	9.86E+04	
R^2	9.98E-01	

α	***1.043E+10***	m kg^{-1}
R_m	***1.675E+11***	m^{-1}

Mass of *wet* cake	*76.15*	g
Mass of *dry* cake	*47.74*	g
Mass of liquid	28.41	g
Volume of solid	1.85E-05	m^3
Volume of liquid	2.84E-05	m^3
Cake conc^n	39.44%	
ε	0.6056	
Moisture ratio, m	1.5951	
Effective c	1061.70	kg m^{-3}
Slurry c	650.63	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1034.84	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	4.48E+08	
Intercept	8.08E+04	
R^2	9.98E-01	

α	***1.249E+10***	**m kg^{-1}**
R$_m$	***1.373E+11***	**m^{-1}**

Mass of *wet* cake	*85.57*	g
Mass of *dry* cake	*54.03*	g
Mass of liquid	31.54	g
Volume of solid	2.09E-05	m^3
Volume of liquid	3.15E-05	m^3
Cake conc^n	39.90%	
ε	0.6010	
Moisture ratio, m	1.5837	
Effective c	1034.84	**kg m^{-3}**
Slurry c	645.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

2BAR_08.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1034.82	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	4.13E+08	
Intercept	6.61E+04	
R^2	9.97E-01	

α	***1.152E+10***	**m kg^{-1}**
R$_m$	***1.123E+11***	**m^{-1}**

Mass of *wet* cake	*82.37*	g
Mass of *dry* cake	*52.01*	g
Mass of liquid	30.36	g
Volume of solid	2.02E-05	m^3
Volume of liquid	3.04E-05	m^3
Cake conc^n	39.90%	
ε	0.6010	
Moisture ratio, m	1.5837	
Effective c	1034.82	**kg m^{-3}**
Slurry c	645.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

2BAR_09.XLS

	Filtrate	Solid	
Weight	*800.0*	*516.6*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^{3}

Total Mass	1.32	kg
Total Volume	1.00E-03	m^{3}
Solids conc^{n}	1007.63	kg m^{-3}
$\equiv$	20.02%	v/v
$\equiv$	39.24%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^{2}
Pressure	*3*	bar
	300000	Pa
Slope of plot	2.82E+08	
Intercept	7.59E+04	
R^{2}	9.98E-01	

α	*1.214E+10*	m kg^{-1}
R$_{m}$	*1.934E+11*	m^{-1}

Mass of *wet* cake	*71.07*	g
Mass of *dry* cake	*45.67*	g
Mass of liquid	25.40	g
Volume of solid	1.77E-05	m^{3}
Volume of liquid	2.54E-05	m^{3}
Cake conc^{n}	41.07%	
ε	0.5893	
Moisture ratio, m	1.5562	
Effective c	1007.63	kg m^{-3}
Slurry c	645.75	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^{n}, 2 : Effective conc^{n}

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1040.37	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	3.01E+08	
Intercept	7.76E+04	
R^2	9.95E-01	
α	***1.254E+10***	**m kg^{-1}**
R$_m$	***1.977E+11***	**m^{-1}**

Mass of *wet* cake	*79.54*	g
Mass of *dry* cake	*50.06*	g
Mass of liquid	29.48	g
Volume of solid	1.94E-05	m^3
Volume of liquid	2.95E-05	m^3
Cake conc^n	39.69%	
ε	0.6031	
Moisture ratio, m	1.5889	
Effective c	1040.37	kg m^{-3}
Slurry c	645.13	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

3BAR_02.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.2*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1029.70	kg m^{-3}
≡	20.01%	v/v
≡	39.22%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	2.98E+08	
Intercept	6.82E+04	
R^2	9.96E-01	

α	***1.255E+10***	**m kg^{-1}**
R_m	***1.739E+11***	**m^{-1}**

Mass of *wet* cake	*71.37*	g
Mass of *dry* cake	*45.21*	g
Mass of liquid	26.16	g
Volume of solid	1.75E-05	m^3
Volume of liquid	2.62E-05	m^3
Cake conc^n	40.11%	
ε	0.5989	
Moisture ratio, m	1.5786	
Effective c	1029.70	**kg m^{-3}**
Slurry c	645.25	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

3BAR_03.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.0*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids concⁿ	1041.92	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	3.17E+08	
Intercept	6.35E+04	
R^2	9.95E-01	

α	***1.317E+10***	m kg^{-1}
R_m	***1.618E+11***	m^{-1}

Mass of *wet* cake	*72.58*	g
Mass of *dry* cake	*45.63*	g
Mass of liquid	26.95	g
Volume of solid	1.77E-05	m^3
Volume of liquid	2.70E-05	m^3
Cake concⁿ	39.62%	
ε	0.6038	
Moisture ratio, m	1.5906	
Effective c	1041.92	kg m^{-3}
Slurry c	645.00	kg m^{-3}

Use value for c: *2*

1 : Slurry concⁿ, 2 : Effective concⁿ

3BAR_04.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*873.8*	*516.0*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.74E-04	2.00E-04	m^{3}

Total Mass	1.39	kg
Total Volume	1.07E-03	m^{3}
Solids conc^{n}	889.18	kg m^{-3}
≡	18.62%	v/v
≡	37.13%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^{2}
Pressure	*3*	bar
	300000	Pa
Slope of plot	2.85E+08	
Intercept	6.34E+04	
R^{2}	9.93E-01	

α	***1.39E+10***	m kg^{-1}
R$_{m}$	***1.616E+11***	m^{-1}

Mass of *wet* cake	*67.10*	g
Mass of *dry* cake	*42.77*	g
Mass of liquid	24.33	g
Volume of solid	1.66E-05	m^{3}
Volume of liquid	2.43E-05	m^{3}
Cake conc^{n}	40.52%	
ε	0.5948	
Moisture ratio, m	1.5689	
Effective c	889.18	kg m^{-3}
Slurry c	590.50	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^{n}, 2 : Effective conc^{n}

3BAR_05.XLS

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*873.8*	*517.2*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.74E-04	2.00E-04	m^3

Total Mass	1.39	kg
Total Volume	1.07E-03	m^3
Solids conc^n	905.54	kg m^{-3}
≡	18.66%	v/v
≡	37.18%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	2.95E+08	
Intercept	6.64E+04	
R^2	9.96E-01	

α	***1.411E+10***	m kg^{-1}
R$_m$	***1.692E+11***	m^{-1}

Mass of *wet* cake	*69.18*	g
Mass of *dry* cake	*43.64*	g
Mass of liquid	25.54	g
Volume of solid	1.69E-05	m^3
Volume of liquid	2.55E-05	m^3
Cake conc^n	39.84%	
ε	0.6016	
Moisture ratio, m	1.5852	
Effective c	905.54	kg m^{-3}
Slurry c	591.87	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

3BAR_06.XLS

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*515.6*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1043.73	kg m^{-3}
≡	19.99%	v/v
≡	39.19%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	4.03E+08	
Intercept	6.17E+04	
R^2	9.88E-01	

α	*1.674E+10*	m kg^{-1}
R_m	*1.572E+11*	m^{-1}

Mass of *wet* cake	*60.68*	g
Mass of *dry* cake	*38.08*	g
Mass of liquid	22.60	g
Volume of solid	1.48E-05	m^3
Volume of liquid	2.26E-05	m^3
Cake conc^n	39.51%	
ε	0.6049	
Moisture ratio, m	1.5935	
Effective c	1043.73	kg m^{-3}
Slurry c	644.50	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

3BAR_07.XLS

Data Sheet　　　Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*775.50*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	3.01E-04	m^3

Total Mass	1.58	kg
Total Volume	1.10E-03	m^3
Solids conc.	704.63	kg m^{-3}
≡	27.31%	v/v
≡	49.22%	w/w
Viscosity, μ	*0.001*	**Pa s**
Diameter	*10.4*	**cm**
Area	8.49E-03	m^2
Pressure	*4.15*	**bar**
	415000	Pa
Slope of plot	3.60E+08	
Intercept	6.50E+04	
R^2	1.00E+00	

α	***3.059E+10***	**m kg^{-1}**
R_m	***2.293E+11***	**m^{-1}**

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*206.30*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	8.00E-05	m^3

Total Mass	1.01	kg
Total Volume	8.80E-04	m^3
Solids conc.	234.44	kg m^{-3}
≡	9.09%	v/v
≡	20.50%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*4.2*	bar
	420000	Pa
Slope of plot	8.52E+07	
Intercept	4.11E+04	
R^2	9.98E-01	

α	***2.204E+10***	**m kg^{-1}**
R_m	***1.467E+11***	**m^{-1}**

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.00*	*516.00*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc.	516.00	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*4*	bar
	400000	Pa
Slope of plot	2.60E+08	
Intercept	1.15E+05	
R^2	9.96E-01	

α	***2.904E+10***	**m kg^{-1}**
R$_m$	***3.914E+11***	**m^{-1}**

4BAR_03.XLS

Data Sheet Material: Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	998.61	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*4*	bar
	400000	Pa
Slope of plot	2.47E+08	
Intercept	4.61E+04	
R^2	9.99E-01	

α	***1.428E+10***	**m kg^{-1}**
R$_m$	***1.567E+11***	**m^{-1}**

Mass of *wet* cake	*62.97*	g
Mass of *dry* cake	*40.66*	g
Mass of liquid	22.31	g
Volume of solid	1.58E-05	m^3
Volume of liquid	2.23E-05	m^3
Cake conc^n	41.40%	
ε	0.5860	
Moisture ratio, m	1.5487	
Effective c	998.61	**kg m^{-3}**
Slurry c	645.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

4BAR_06.XLS

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1027.78	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*4*	bar
	400000	Pa
Slope of plot	2.28E+08	
Intercept	5.20E+04	
R^2	9.95E-01	

α	***1.281E+10***	m kg^{-1}
R$_m$	***1.765E+11***	m^{-1}

Mass of *wet* cake	*71.68*	g
Mass of *dry* cake	*45.45*	g
Mass of liquid	26.23	g
Volume of solid	1.76E-05	m^3
Volume of liquid	2.62E-05	m^3
Cake conc^n	40.18%	
ε	0.5982	
Moisture ratio, m	1.5771	
Effective c	1027.78	kg m^{-3}
Slurry c	645.13	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n , 2 : Effective conc^n

4BAR_07.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1023.16	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*4*	bar
	400000	Pa
Slope of plot	2.41E+08	
Intercept	7.02E+04	
R^2	9.99E-01	

α	***1.361E+10***	m kg^{-1}
R$_m$	***2.384E+11***	m^{-1}

Mass of *wet* cake	*93.97*	g
Mass of *dry* cake	*59.75*	g
Mass of liquid	34.22	g
Volume of solid	2.32E-05	m^3
Volume of liquid	3.42E-05	m^3
Cake conc^n	40.36%	
ε	0.5964	
Moisture ratio, m	1.5727	
Effective c	1023.16	kg m^{-3}
Slurry c	645.13	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n , 2 : Effective conc^n

Data Sheet Material: Calcite

	Filtrate	Solid	
Weight	*800.00*	*516.20*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	992.38	kg m^{-3}
≡	20.01%	v/v
≡	39.22%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*5*	bar
	500000	Pa
Slope of plot	1.90E+08	
Intercept	3.64E+04	
R^2	9.99E-01	

α	*1.381E+10*	m kg^{-1}
R_m	*1.547E+11*	m^{-1}

Mass of *wet* cake	*128.35*	g
Mass of *dry* cake	*83.23*	g
Mass of liquid	45.12	g
Volume of solid	3.23E-05	m^3
Volume of liquid	4.51E-05	m^3
Cake conc^n	41.69%	
ε	0.5831	
Moisture ratio, m	1.5421	
Effective c	992.38	kg m^{-3}
Slurry c	645.25	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

5BAR_01.XLS

Data Sheet   Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1026.23	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*5*	bar
	500000	Pa
Slope of plot	1.98E+08	
Intercept	5.28E+04	
R^2	9.95E-01	

α	*1.394E+10*	m kg^{-1}
R$_m$	*2.242E+11*	m^{-1}

Mass of *wet* cake	*76.86*	g
Mass of *dry* cake	*48.78*	g
Mass of liquid	28.08	g
Volume of solid	1.89E-05	m^3
Volume of liquid	2.81E-05	m^3
Cake conc^n	40.24%	
ε	0.5976	
Moisture ratio, m	1.5756	
Effective c	1026.23	kg m^{-3}
Slurry c	645.13	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

5BAR_02.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1026.14	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*5*	bar
	500000	Pa
Slope of plot	1.96E+08	
Intercept	4.09E+04	
R^2	9.98E-01	

α	***1.376E+10***	**m kg^{-1}**
R_m	***1.737E+11***	**m^{-1}**

Mass of *wet* cake	*62.14*	g
Mass of *dry* cake	*39.44*	g
Mass of liquid	22.70	g
Volume of solid	1.53E-05	m^3
Volume of liquid	2.27E-05	m^3
Cake conc^n	40.24%	
ε	0.5976	
Moisture ratio, m	1.5756	
Effective c	1026.14	**kg m^{-3}**
Slurry c	645.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

5BAR_03.XLS

Data Sheet Material: Calcite

	Filtrate	Solid	
Weight	*800.00*	*516.60*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	997.03	kg m^{-3}
≡	20.02%	v/v
≡	39.24%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*6*	bar
	600000	Pa
Slope of plot	1.66E+08	
Intercept	3.60E+04	
R^2	9.99E-01	

α	***1.443E+10***	**m kg^{-1}**
R$_m$	***1.835E+11***	**m^{-1}**

Mass of *wet* cake	*72.18*	g
Mass of *dry* cake	*46.70*	g
Mass of liquid	25.48	g
Volume of solid	1.81E-05	m^3
Volume of liquid	2.55E-05	m^3
Cake conc^n	41.53%	
ε	0.5847	
Moisture ratio, m	1.5456	
Slurry c	645.75	kg m^{-3}
Effective c	997.03	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

6BAR_01.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1031.92	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*6*	bar
	600000	Pa
Slope of plot	1.64E+08	
Intercept	4.36E+04	
R^2	9.93E-01	

α	***1.375E+10***	**m kg^{-1}**
R_m	***2.223E+11***	**m^{-1}**

Mass of *wet* cake	*50.64*	g
Mass of *dry* cake	*32.03*	g
Mass of liquid	18.61	g
Volume of solid	1.24E-05	m^3
Volume of liquid	1.86E-05	m^3
Cake conc^n	40.02%	
ε	0.5998	
Moisture ratio, m	1.5810	
Effective c	1031.92	**kg m^{-3}**
Slurry c	645.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

6BAR_02.XLS

Data Sheet

Material : Calcite

	Filtrate	Solid	
Weight	*800.0*	*516.1*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.00E-04	2.00E-04	m^3

Total Mass	1.32	kg
Total Volume	1.00E-03	m^3
Solids conc^n	1027.18	kg m^{-3}
≡	20.00%	v/v
≡	39.21%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*6*	bar
	600000	Pa
Slope of plot	1.70E+08	
Intercept	4.83E+04	
R^2	9.93E-01	

α	***1.432E+10***	m kg^{-1}
R$_m$	***2.463E+11***	m^{-1}

Mass of *wet* cake	*65.08*	g
Mass of *dry* cake	*41.28*	g
Mass of liquid	23.80	g
Volume of solid	1.60E-05	m^3
Volume of liquid	2.38E-05	m^3
Cake conc^n	40.20%	
ε	0.5980	
Moisture ratio, m	1.5766	
Effective c	1027.18	kg m^{-3}
Slurry c	645.13	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

6BAR_03.XLS

Data Sheet Material : Calcite

	Filtrate	Solid	
Weight	*810.0*	*516.0*	g
Density, ρ	*1000*	*2580*	kg m^{-3}
Volume	8.10E-04	2.00E-04	m^3

Total Mass	1.33	kg
Total Volume	1.01E-03	m^3
Solids conc^n	1009.08	kg m^{-3}
≡	19.80%	v/v
≡	38.91%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*6*	bar
	600000	Pa
Slope of plot	1.63E+08	
Intercept	4.68E+04	
R^2	9.92E-01	

α	***1.395E+10***	**m kg^{-1}**
R$_m$	***2.383E+11***	**m^{-1}**

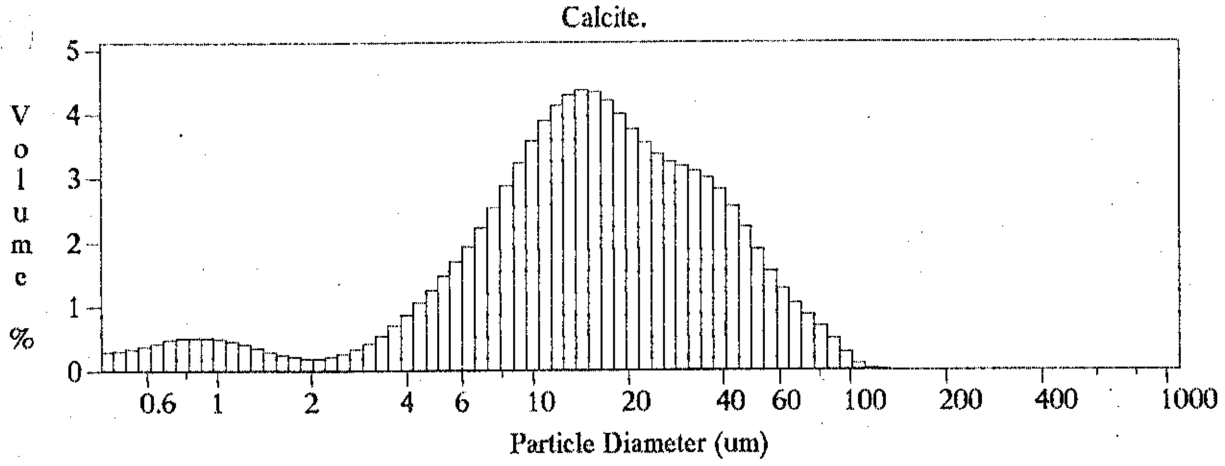
Mass of *wet* cake	*68.55*	g
Mass of *dry* cake	*43.42*	g
Mass of liquid	25.13	g
Volume of solid	1.68E-05	m^3
Volume of liquid	2.51E-05	m^3
Cake conc^n	40.11%	
ε	0.5989	
Moisture ratio, m	1.5788	
Effective c	1009.08	**kg m^{-3}**
Slurry c	637.04	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

6BAR_04.XLS

File name:	97193.$01	Group ID:	97193
Sample ID:	Calcite.		
Operator:	SGG	Run number:	1
Comments:	Dispersed in distilled water.		
Start time:	10:36 6 Feb 1997	Run length:	60 Seconds
Obscuration:	7%		
Optical model:	Fraunhofer		
Fluid:	Water		
LS 130	Micro-volume module		
Software:	1.53	Firmware:	1.3 1.8



Volume %	Particle Diameter um <
10.00	4.221
25.00	8.673
50.00	15.47
75.00	28.37
90.00	45.40

Particle Diameter um	Diff. Volume %	Cum. < Volume %
0.429	0.59	0.00
0.515	0.70	0.59
0.618	0.89	1.29
0.741	1.01	2.18
0.889	0.99	3.19
1.067	0.85	4.18
1.280	0.64	5.03
1.536	0.47	5.67
1.842	0.39	6.14
2.210	0.48	6.54
2.652	0.75	7.02
3.181	1.25	7.77
3.817	1.93	9.02
4.579	2.73	10.95
5.494	3.64	13.69
6.591	4.75	17.32
7.907	6.12	22.07
9.487	7.47	28.19
11.38	8.43	35.66
13.65	8.70	44.09
16.38	8.20	52.79
19.65	7.30	60.99
23.58	6.62	68.29
28.29	6.29	74.91
33.94	5.82	81.20
40.72	4.79	87.02
48.85	3.45	91.81
58.61	2.33	95.26
70.32	1.54	97.58
84.36	0.76	99.12
101.2	0.11	99.89
121.4	0.00	100.00
145.7	0.00	100.00
174.8	0.00	100.00
209.7	0.00	100.00
251.6	0.00	100.00

TALC

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.89E-05	m^{3}

Total Mass	1.04	kg
Total Volume	8.89E-04	m^{3}
Solids conc^{n}	425.57	kg m^{-3}
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^{2}
Pressure	*1*	bar
	100000	Pa
Slope of plot	5.17E+09	
Intercept	1.40E+04	
R^{2}	9.96E-01	

α	***1.752E+11***	m kg^{-1}
R$_{m}$	***1.191E+10***	m^{-1}

Mass of *wet* cake	*27.39*	g
Mass of *dry* cake	*14.17*	g
Mass of liquid	13.22	g
Volume of solid	5.17E-06	m^{3}
Volume of liquid	1.32E-05	m^{3}
Cake conc^{n}	28.11%	
ε	0.7189	
Moisture ratio, m	1.9330	
Effective c	425.57	kg m^{-3}
Slurry c	304.63	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^{n}, 2 : Effective conc^{n}

Data Sheet Material: Talc

	Filtrate	Solid	
Weight	*1200.0*	*365.5*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	1.20E-03	1.33E-04	m^3

Total Mass	1.57	kg
Total Volume	1.33E-03	m^3
Solids conc^n	441.42	kg m^{-3}
$\equiv$	10.00%	v/v
$\equiv$	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1.1*	bar
	110000	Pa
Slope of plot	3.22E+09	
Intercept	2.45E+05	
R^2	1.00E+00	

α	***1.157E+11***	m kg^{-1}
R$_m$	***2.291E+11***	m^{-1}

Mass of *wet* cake	*44.37*	g
Mass of *dry* cake	*21.99*	g
Mass of liquid	22.38	g
Volume of solid	8.02E-06	m^3
Volume of liquid	2.24E-05	m^3
Cake conc^n	26.38%	
ε	0.7362	
Moisture ratio, m	2.0177	
Effective c	441.42	kg m^{-3}
Slurry c	304.58	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

1BAR_02.XLS

Data Sheet Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*313.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	1.14E-04	m^3

Total Mass	1.11	kg
Total Volume	9.14E-04	m^3
Solids conc^n	618.60	kg m^{-3}
≡	12.51%	v/v
≡	28.17%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1*	bar
	100000	Pa
Slope of plot	6.46E+09	
Intercept	5.15E+05	
R^2	9.96E-01	

α	***1.508E+11***	**m kg^{-1}**
R_m	***4.374E+11***	**m^{-1}**

Mass of *wet* cake	*45.75*	g
Mass of *dry* cake	*23.66*	g
Mass of liquid	22.09	g
Volume of solid	8.63E-06	m^3
Volume of liquid	2.21E-05	m^3
Cake conc^n	28.09%	
ε	0.7191	
Moisture ratio, m	1.9336	
Effective c	618.60	**kg m^{-3}**
Slurry c	392.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

1BAR_03.XLS

	Filtrate	Solid	
Weight	*800.0*	*313.4*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	1.14E-04	m^3

Total Mass	1.11	kg
Total Volume	9.14E-04	m^3
Solids conc^n	609.10	kg m^{-3}
≡	12.50%	v/v
≡	28.15%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*1*	bar
	100000	Pa
Slope of plot	4.46E+09	
Intercept	3.52E+05	
R^2	9.98E-01	

α	***1.058E+11***	**m kg^{-1}**
R_m	***2.993E+11***	**m^{-1}**

Mass of *wet* cake	*45.88*	g
Mass of *dry* cake	*24.01*	g
Mass of liquid	21.87	g
Volume of solid	8.76E-06	m^3
Volume of liquid	2.19E-05	m^3
Cake conc^n	28.59%	
ε	0.7141	
Moisture ratio, m	1.9109	
Effective c	609.10	**kg m^{-3}**
Slurry c	391.75	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

	Filtrate	Solid	
Weight	*800.0*	*243.9*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.90E-05	m^3

Total Mass	1.04	kg
Total Volume	8.89E-04	m^3
Solids conc^n	435.29	kg m^{-3}
≡	10.01%	v/v
≡	23.36%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	2.13E+09	
Intercept	-8.21E+03	
R^2	1.00E+00	

α	***1.414E+11***	**m kg^{-1}**
R$_m$	***-1.39E+10***	**m^{-1}**

Mass of *wet* cake	*63.11*	g
Mass of *dry* cake	*31.83*	g
Mass of liquid	31.28	g
Volume of solid	1.16E-05	m^3
Volume of liquid	3.13E-05	m^3
Cake conc^n	27.07%	
ε	0.7293	
Moisture ratio, m	1.9827	
Effective c	435.29	**kg m^{-3}**
Slurry c	304.88	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

Data Sheet

Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*313.3*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	1.14E-04	m^3

Total Mass	1.11	kg
Total Volume	9.14E-04	m^3
Solids conc^n	587.41	kg m^{-3}
≡	12.50%	v/v
≡	28.14%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	2.84E+09	
Intercept	2.39E+05	
R^2	9.99E-01	

α	***1.394E+11***	**m kg^{-1}**
R$_m$	***4.068E+11***	**m^{-1}**

Mass of *wet* cake	*58.30*	g
Mass of *dry* cake	*31.50*	g
Mass of liquid	26.80	g
Volume of solid	1.15E-05	m^3
Volume of liquid	2.68E-05	m^3
Cake conc^n	30.00%	
ε	0.7000	
Moisture ratio, m	1.8511	
Effective c	587.41	**kg m^{-3}**
Slurry c	391.63	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

	Filtrate	Solid	
Weight	*800.0*	*313.3*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	1.14E-04	m^3

Total Mass	1.11	kg
Total Volume	9.14E-04	m^3
Solids conc^n	599.83	kg m^{-3}
≡	12.50%	v/v
≡	28.14%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*2*	bar
	200000	Pa
Slope of plot	3.25E+09	
Intercept	2.20E+05	
R^2	9.97E-01	

α	***1.565E+11***	**m kg^{-1}**
R$_m$	***3.737E+11***	**m^{-1}**

Mass of *wet* cake	*53.76*	g
Mass of *dry* cake	*28.50*	g
Mass of liquid	25.26	g
Volume of solid	1.04E-05	m^3
Volume of liquid	2.53E-05	m^3
Cake conc^n	29.16%	
ε	0.7084	
Moisture ratio, m	1.8863	
Effective c	599.83	**kg m^{-3}**
Slurry c	391.63	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

Data Sheet Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	$kg\ m^{-3}$
Volume	8.00E-04	8.89E-05	m^3

Total Mass	1.04	kg
Total Volume	8.89E-04	m^3
Solids conc^n	416.94	$kg\ m^{-3}$
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	1.70E+09	
Intercept	4.57E+04	
R^2	9.97E-01	

α	***1.767E+11***	$m\ kg^{-1}$
R_m	***1.164E+11***	m^{-1}

Mass of *wet* cake	*54.55*	g
Mass of *dry* cake	*28.95*	g
Mass of liquid	25.60	g
Volume of solid	1.06E-05	m^3
Volume of liquid	2.56E-05	m^3
Cake conc^n	29.20%	
ε	0.7080	
Moisture ratio, m	1.8843	
Effective c	416.94	$kg\ m^{-3}$
Slurry c	304.63	$kg\ m^{-3}$

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

3BAR_01.XLS

Data Sheet

Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.89E-05	m^3

Total Mass	1.04	kg
Total Volume	8.89E-04	m^3
Solids conc^n	367.22	kg m^{-3}
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	1.10E+09	
Intercept	1.83E+05	
R^2	9.97E-01	

α	***1.297E+11***	m kg^{-1}
R_m	***4.668E+11***	m^{-1}

Mass of *wet* cake	*55.66*	g
Mass of *dry* cake	*35.69*	g
Mass of liquid	19.97	g
Volume of solid	1.30E-05	m^3
Volume of liquid	2.00E-05	m^3
Cake conc^n	39.46%	
ε	0.6054	
Moisture ratio, m	1.5595	
Effective c	367.22	kg m^{-3}
Slurry c	304.63	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

Data Sheet

Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.89E-05	m^{3}

Total Mass	1.04	kg
Total Volume	8.89E-04	m^{3}
Solids conc^{n}	410.17	kg m^{-3}
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^{2}
Pressure	*3*	bar
	300000	Pa
Slope of plot	1.26E+09	
Intercept	1.40E+05	
R^{2}	1.00E+00	

α	***1.326E+11***	m kg^{-1}
R$_{m}$	***3.572E+11***	m^{-1}

Mass of *wet* cake	*25.66*	g
Mass of *dry* cake	*13.91*	g
Mass of liquid	11.75	g
Volume of solid	5.07E-06	m^{3}
Volume of liquid	1.18E-05	m^{3}
Cake conc^{n}	30.16%	
ε	0.6984	
Moisture ratio, m	1.8447	
Effective c	410.17	kg m^{-3}
Slurry c	304.63	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^{n}, 2 : Effective conc^{n}

Data Sheet Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.89E-05	m^{3}

Total Mass	1.04	kg
Total Volume	8.89E-04	m^{3}
Solids conc^{n}	417.71	kg m^{-3}
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^{2}
Pressure	*4*	bar
	400000	Pa
Slope of plot	1.07E+09	
Intercept	8.26E+04	
R^{2}	9.88E-01	

α	***1.485E+11***	**m kg^{-1}**
R$_{m}$	***2.805E+11***	**m^{-1}**

Mass of *wet* cake	*69.77*	g
Mass of *dry* cake	*36.94*	g
Mass of liquid	32.83	g
Volume of solid	1.35E-05	m^{3}
Volume of liquid	3.28E-05	m^{3}
Cake conc^{n}	29.10%	
ε	0.7090	
Moisture ratio, m	1.8887	
Effective c	417.71	**kg m^{-3}**
Slurry c	304.63	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^{n}, 2 : Effective conc^{n}

4BAR_01.XLS

Data Sheet Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.89E-05	m^3

Total Mass	1.04	kg
Total Volume	8.89E-04	m^3
Solids conc^n	363.80	kg m^{-3}
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*3*	bar
	300000	Pa
Slope of plot	2.26E+09	
Intercept	-3.49E+05	
R^2	9.51E-01	

α	***2.694E+11***	**m kg^{-1}**
R_m	***-8.88E+11***	**m^{-1}**

Mass of *wet* cake	*14.68*	g
Mass of *dry* cake	*9.57*	g
Mass of liquid	5.11	g
Volume of solid	3.49E-06	m^3
Volume of liquid	5.11E-06	m^3
Cake conc^n	40.59%	
ε	0.5941	
Moisture ratio, m	1.5340	
Effective c	363.80	**kg m^{-3}**
Slurry c	304.63	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

4BAR_07.XLS

	Filtrate	Solid	
Weight	*800.0*	*247.3*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	9.02E-05	m^{3}

Total Mass	1.05	kg
Total Volume	8.90E-04	m^{3}
Solids conc^{n}	399.80	kg m^{-3}
≡	10.13%	v/v
≡	23.61%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^{2}
Pressure	*5*	bar
	500000	Pa
Slope of plot	7.70E+08	
Intercept	8.13E+04	
R^{2}	9.95E-01	

α	***1.391E+11***	**m kg^{-1}**
R$_{m}$	***3.453E+11***	**m^{-1}**

Mass of *wet* cake	*21.81*	g
Mass of *dry* cake	*12.58*	g
Mass of liquid	9.23	g
Volume of solid	4.59E-06	m^{3}
Volume of liquid	9.23E-06	m^{3}
Cake conc^{n}	33.21%	
ε	0.6679	
Moisture ratio, m	1.7337	
Effective c	399.80	**kg m^{-3}**
Slurry c	309.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^{n}, 2 : Effective conc^{n}

	Filtrate	Solid	
Weight	*800.0*	*247.3*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	9.02E-05	m^3

Total Mass	1.05	kg
Total Volume	8.90E-04	m^3
Solids conc^n	400.80	kg m^{-3}
≡	10.13%	v/v
≡	23.61%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*5*	bar
	500000	Pa
Slope of plot	1.20E+09	
Intercept	1.30E+04	
R^2	9.99E-01	

α	***2.169E+11***	**m kg^{-1}**
R$_m$	***5.515E+10***	**m^{-1}**

Mass of *wet* cake	*53.19*	g
Mass of *dry* cake	*30.57*	g
Mass of liquid	22.62	g
Volume of solid	1.12E-05	m^3
Volume of liquid	2.26E-05	m^3
Cake conc^n	33.02%	
ε	0.6698	
Moisture ratio, m	1.7399	
Effective c	400.80	**kg m^{-3}**
Slurry c	309.13	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

Data Sheet Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.89E-05	m^{3}

Total Mass	1.04	kg
Total Volume	8.89E-04	m^{3}
Solids conc^{n}	388.61	kg m^{-3}
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^{2}
Pressure	*6*	bar
	600000	Pa
Slope of plot	9.01E+08	
Intercept	5.39E+04	
R^{2}	9.94E-01	

α	***2.008E+11***	m kg^{-1}
R$_{m}$	***2.747E+11***	m^{-1}

Mass of *wet* cake	*18.12*	g
Mass of *dry* cake	*10.60*	g
Mass of liquid	7.52	g
Volume of solid	3.87E-06	m^{3}
Volume of liquid	7.52E-06	m^{3}
Cake conc^{n}	33.96%	
ε	0.6604	
Moisture ratio, m	1.7094	
Effective c	388.61	kg m^{-3}
Slurry c	304.63	kg m^{-3}

Use value for c: *2*

1 : Slurry conc^{n}, 2 : Effective conc^{n}

6BAR_01.XLS

Data Sheet

Material: Talc

	Filtrate	Solid	
Weight	*800.0*	*243.7*	g
Density, ρ	*1000*	*2741.6*	kg m^{-3}
Volume	8.00E-04	8.89E-05	m^3

Total Mass	1.04	kg
Total Volume	8.89E-04	m^3
Solids conc^n	397.75	kg m^{-3}
≡	10.00%	v/v
≡	23.35%	w/w
Viscosity, μ	*0.001*	Pa s
Diameter	*10.4*	cm
Area	8.49E-03	m^2
Pressure	*6*	bar
	600000	Pa
Slope of plot	8.33E+08	
Intercept	6.07E+04	
R^2	9.93E-01	

α	***1.814E+11***	**m kg^{-1}**
R$_m$	***3.095E+11***	**m^{-1}**

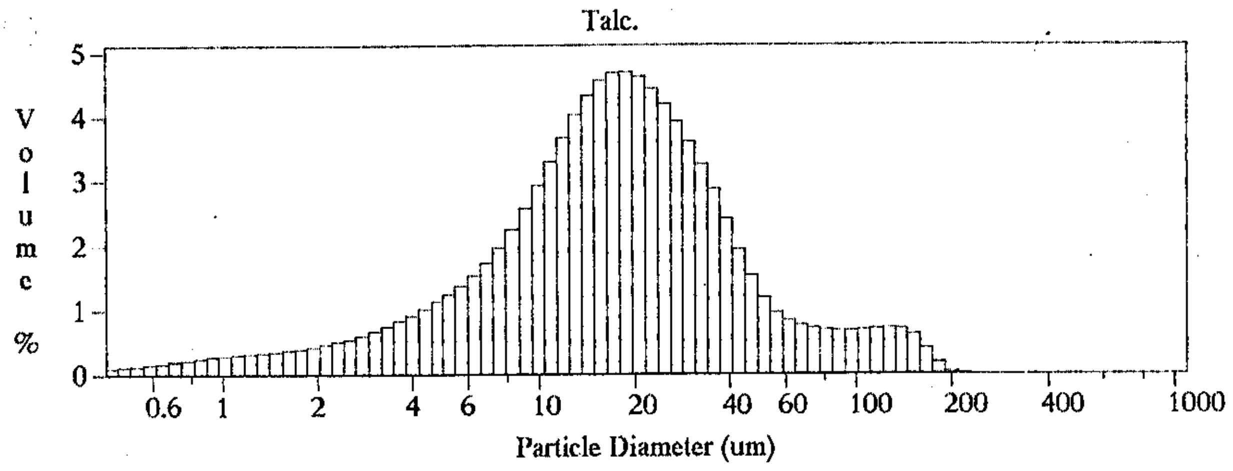
Mass of *wet* cake	*59.99*	g
Mass of *dry* cake	*33.92*	g
Mass of liquid	26.07	g
Volume of solid	1.24E-05	m^3
Volume of liquid	2.61E-05	m^3
Cake conc^n	32.18%	
ε	0.6782	
Moisture ratio, m	1.7686	
Effective c	397.75	**kg m^{-3}**
Slurry c	304.63	**kg m^{-3}**

Use value for c: *2*

1 : Slurry conc^n, 2 : Effective conc^n

6BAR_02.XLS

File name:	97192.$01	Group ID:	97192
Sample ID:	Talc.		
Operator:	SGG	Run number:	1
Comments:	Dispersed in distilled water.		
Start time:	10:18 6 Feb 1997	Run length:	61 Seconds
Obscuration:	11%		
Optical model:	Fraunhofer		
Fluid:	Water		
LS 130	Micro-volume module		
Software:	1.53	Firmware:	1.3 1.8



Volume %	Particle Diameter um <
10.00	4.315
25.00	9.644
50.00	17.27
75.00	29.15
90.00	50.95

97192.$01

Particle Diameter um	Diff. Volume %	Cum. < Volume %
0.429	0.21	0.00
0.515	0.27	0.21
0.618	0.37	0.48
0.741	0.48	0.85
0.889	0.56	1.32
1.067	0.63	1.88
1.280	0.69	2.51
1.536	0.77	3.20
1.842	0.88	3.96
2.210	1.05	4.84
2.652	1.27	5.89
3.181	1.57	7.16
3.817	1.94	8.73
4.579	2.37	10.68
5.494	2.91	13.05
6.591	3.70	15.95
7.907	4.84	19.65
9.487	6.25	24.49
11.38	7.70	30.74
13.65	8.89	38.44
16.38	9.39	47.33
19.65	9.03	56.71
23.58	8.10	65.75
28.29	6.87	73.85
33.94	5.27	80.72
40.72	3.47	86.00
48.85	2.14	89.46
58.61	1.58	91.61
70.32	1.41	93.19
84.36	1.36	94.61
101.2	1.39	95.97
121.4	1.42	97.35
145.7	1.02	98.77
174.8	0.21	99.79
209.7	0.00	100.00
251.6	0.00	100.00

APPENDIX D

SEM's

APPENDIX D - SEM'S

This appendix contains the Scanning Electron Micrographs (SEM's) characterised as part of this work, which includes both individual particles and sectioned filter cakes.

<u>Individual Particles</u>

The images in this section are negatives of the analysed micrographs, to give clarity of presentation.

Table 1 - Calcite Particles

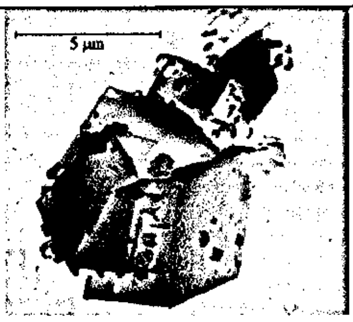
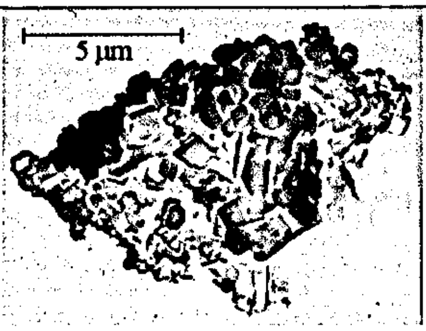
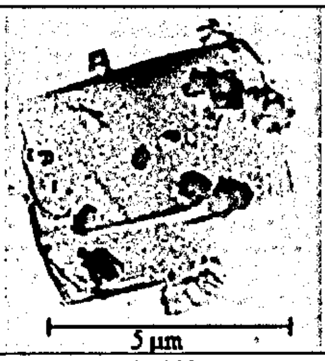
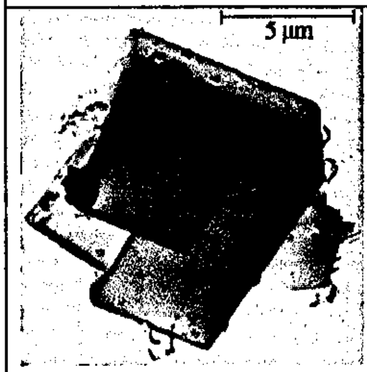
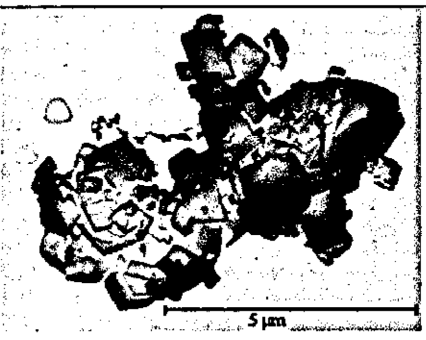
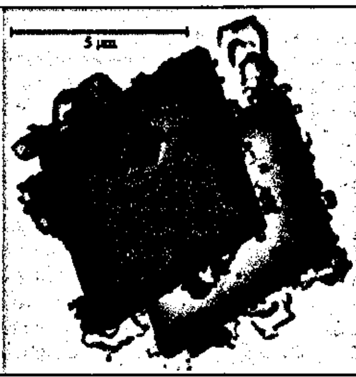
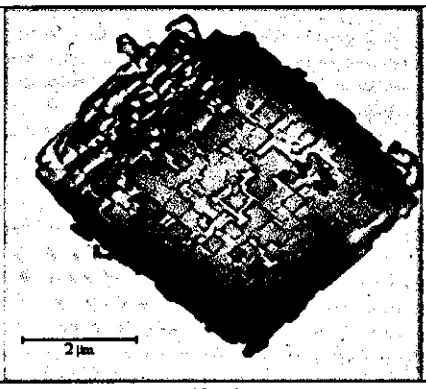
		
6248	6249a	6249b
		
6249c	6250	6251
		
	6252	

Table 2 - Talc Particles

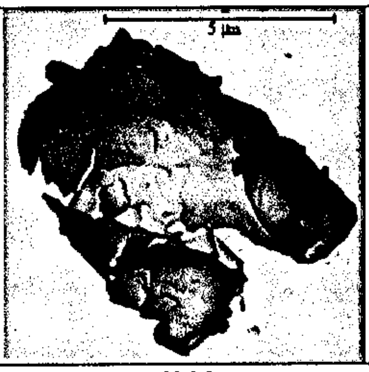
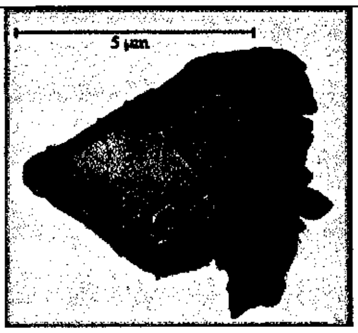
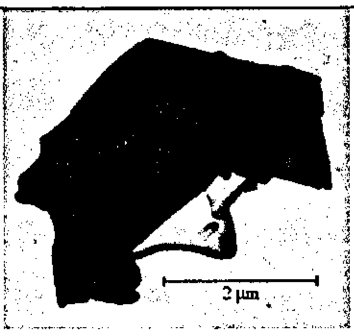
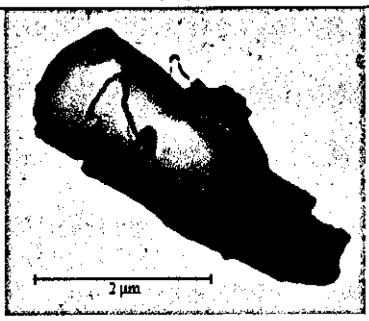
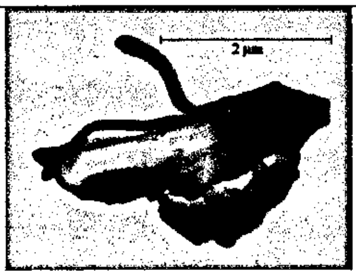
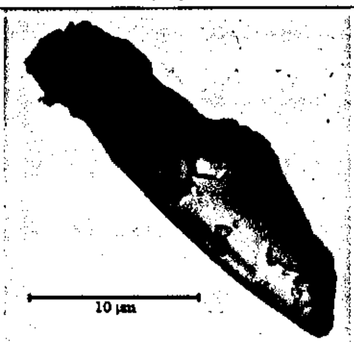
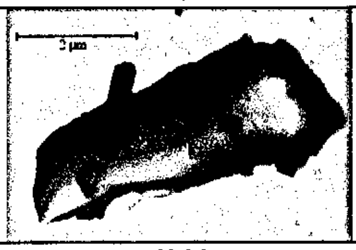
		
6292	6293	6294
		
6295	6296	6297
		
	6298	

Table 3 presents the structured walk fractal dimensions (D_s) for each of the particles, as shown in Tables 1 and 2.

Table 2 – Structured walk fractal dimensions for individual particles

Calcite		Talc	
I.D.	D_s	I.D.	D_s
6248	1.111	6292	1.059
6249a	1.095	6293	1.077
6249b	1.092	6294	1.089
6249c	1.079	6295	1.047
6250	1.253	6296	1.203
6251	1.081	6297	1.100
6252	1.055	6298	1.066

Sectioned filter cakes

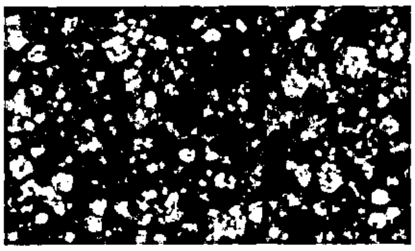
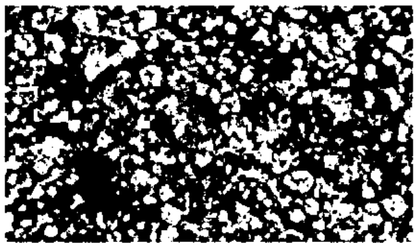
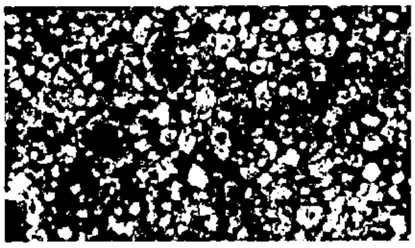
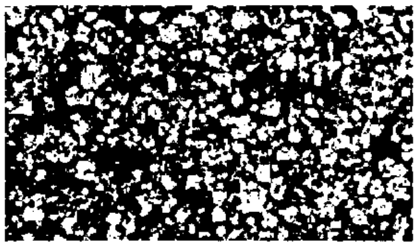
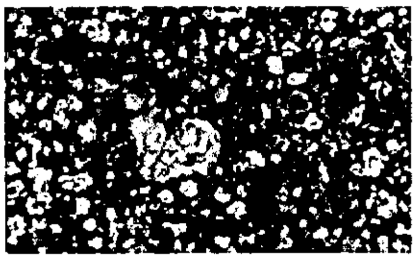
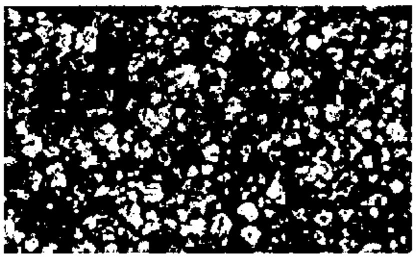
These images show the original sectioned filter cakes. As explained previously, the cakes characterised were formed using calcite in suspension. Two micrographs of each section were taken for examination.

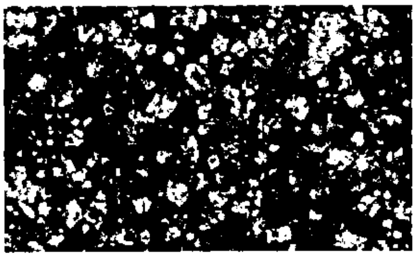
The light-coloured areas in each micrograph represent the suspension mineral, whilst the dark-coloured areas are the epoxy resin used to set the cake after sampling.

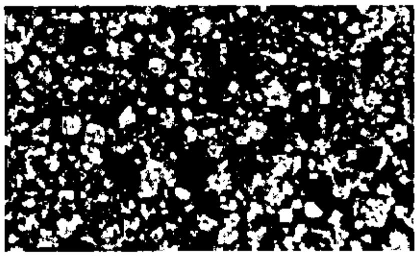
The code sequence below each figure describes the experimental parameters used to form the cake, e.g. C01061a denotes:

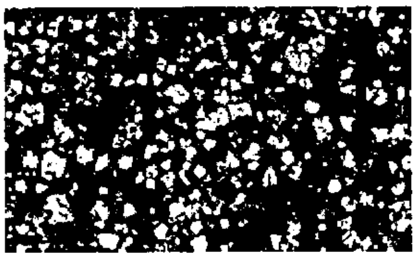
C	Calcite
01	Pressure - 1 bar gauge
06	Experiment number 6
1	Section number 1
a	Picture a

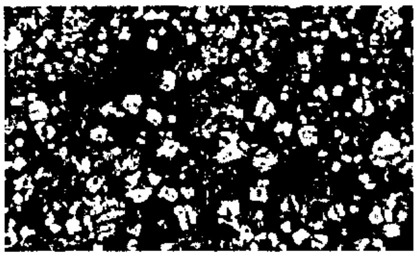
Table 4 – Sectioned Calcite filter cakes

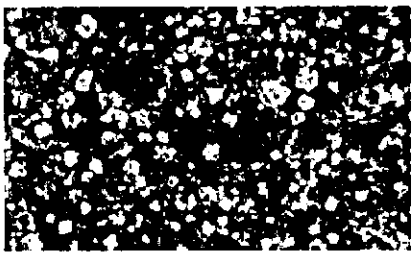
 C01061a	 C01061b
 C01062a	 C01062b
 C01063a	 C01063b

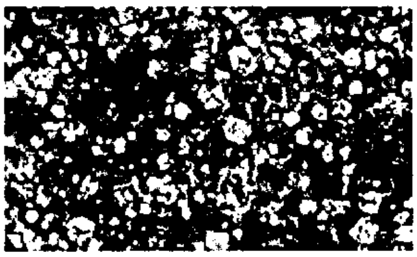

C01064a

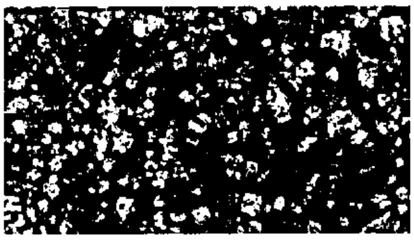

C01064b

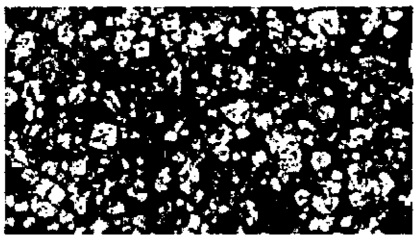

C02081a

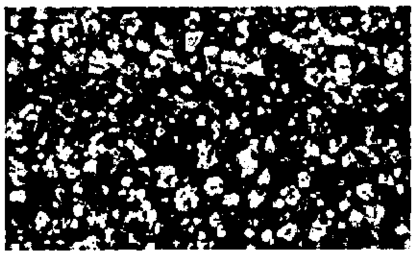

C02081b

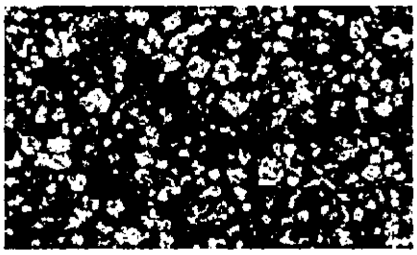

C02082a

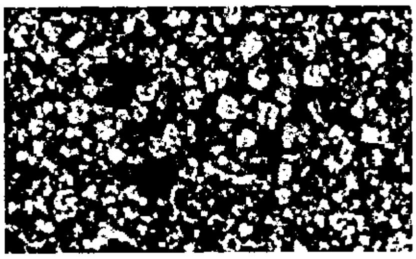

C02082b

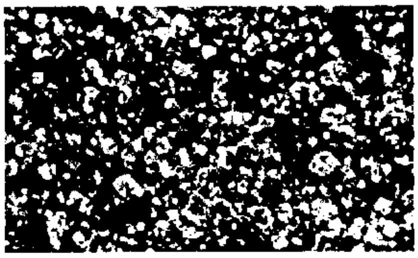

C02083a

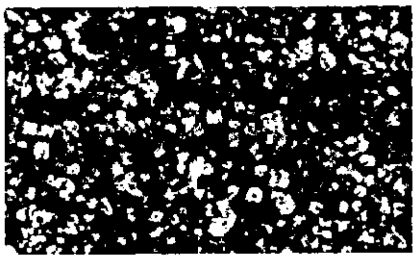

C02083b

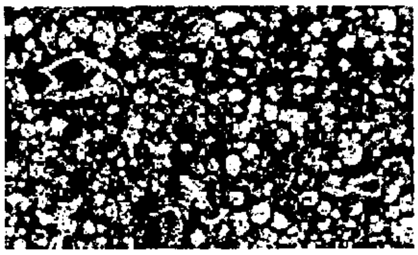

C02091a

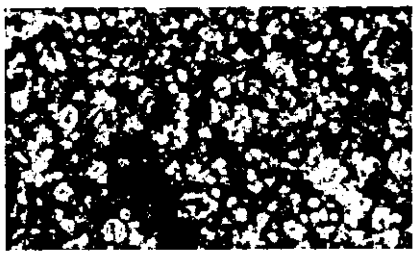

C02091b

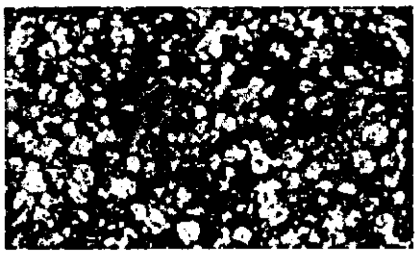

C03051a

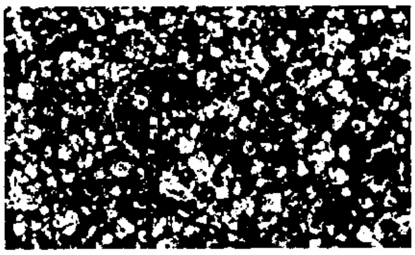

C03051b

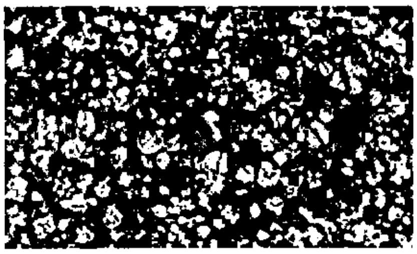

C03052a

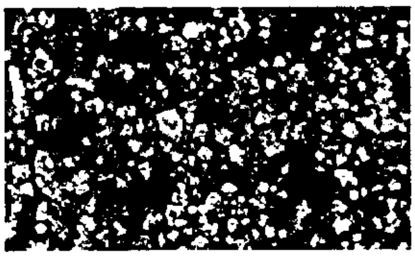

C03052b

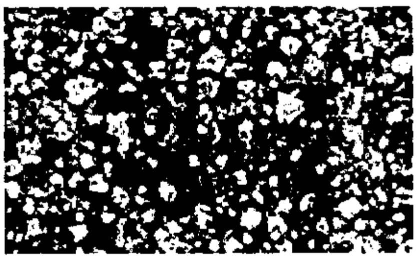

C03053a

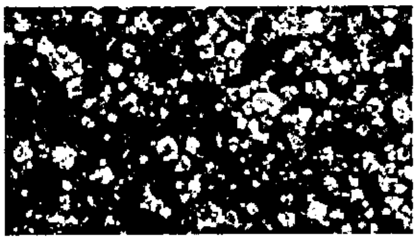

C03053b

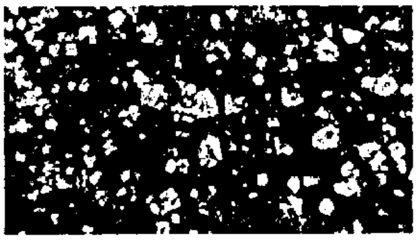

C03054a

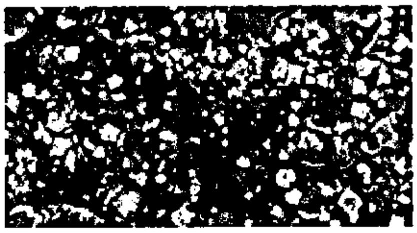

C03054b

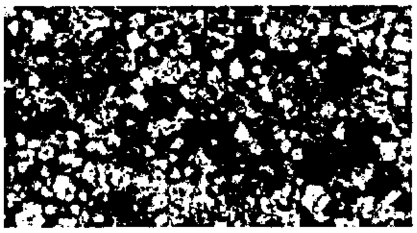

C03061a

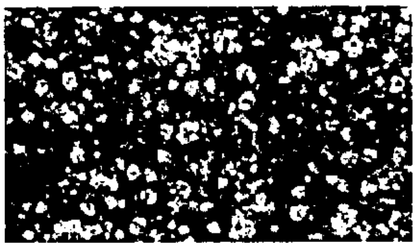

C03061b

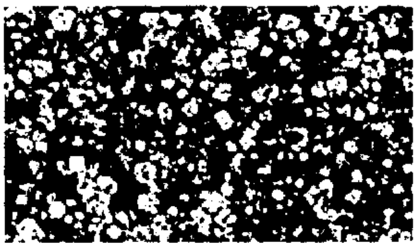

C04081a

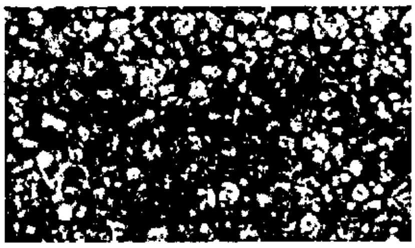

C04081b

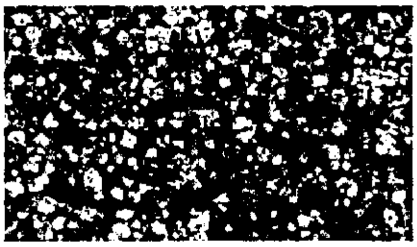

C05031a

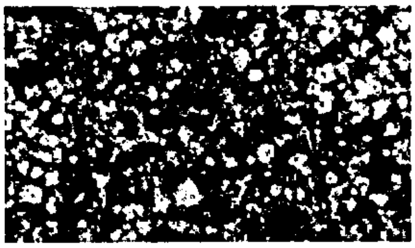

C05031b

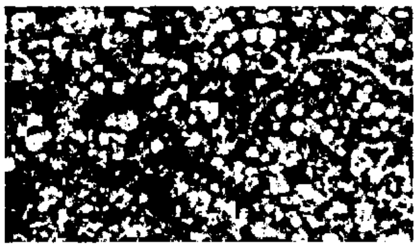

C05032a


C05032b


C05033a


C05033b


C06031a


C06031b

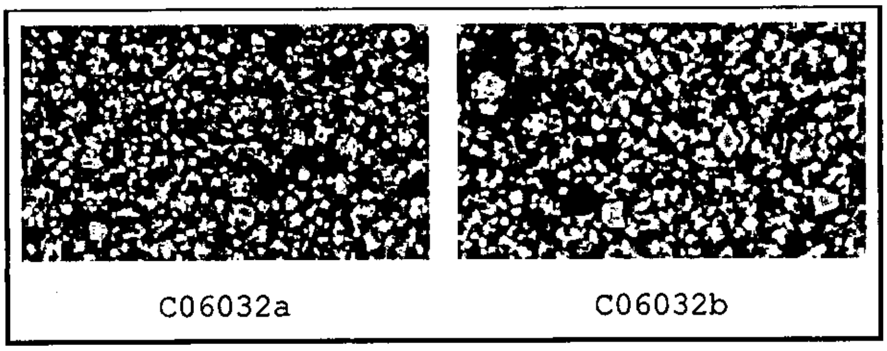


Table 5 gives the box counting (D_b) fractal dimension as well as the two-dimensional porosity (Σ) for the sectioned cakes presented in Table 4. The nomenclature for the cakes is as described above.

Table 5 – Results for fractal and physical properties of sectioned filter cakes

I.D.				D_b	Σ
01	06	1	A	2.141	0.7617
			B	2.141	0.5925
		2	A	2.141	0.6091
			B	2.142	0.5598
		3	A	2.135	0.6960
			B	2.135	0.7057
		4	A	2.135	0.7513
			B	2.133	0.6783
02	08	1	A	2.126	0.7338
			B	2.124	0.7196
		2	A	2.126	0.7233
			B	2.126	0.6833
		3	A	2.138	0.7911
			B	2.141	0.7151
	09	1	A	2.142	0.7175
			B	2.141	0.7230
03	05	1	A	2.133	0.6696
			B	2.135	0.6696
		2	A	2.135	0.7451
			B	2.142	0.4815
		3	A	2.142	0.6585
			B	2.141	0.6868
		4	A	2.142	0.6698
			B	2.142	0.6578
	06	1	A	2.139	0.7413
			B	2.135	0.6467
04	08	1	A	2.137	0.7384
			B	2.135	0.7740
05	03	1	A	2.139	0.7195

Cont'd...

			B	2.143	0.6316
		2	A	2.135	0.7480
			B	2.141	0.6653
		3	A	2.142	0.7061
			B	2.142	0.7134
06	03	1	A	2.142	0.6992
			B	2.142	0.6126
		2	A	2.142	0.6944
			B	2.141	0.6069

APPENDIX E

Published papers

Appendix E – Published Papers

This appendix contains published papers relating to the work in this thesis

1. Tarleton, E.S. and Brock, S.T.H., 1997, Fractal dimensions of computer simulated agglomerates, *IChemE Research Event*, 8-9 April, Nottingham, Chameleon Press, London, 473-476

2. Tarleton, E.S. and Brock, S.T.H., 1997, The fractal properties of two and three dimensional computer simulated agglomerates, *ECCE1*, 4-7 May, Florence, Italy, 2891-2894

3. S.T.H. Brock and E.S. Tarleton, 1998, The use of fractal dimensions in filtration, *World Congress of Particle Technology*, 6-9 July, Brighton

4. S.T.H. Brock and E.S. Tarleton, 1998, Fractal dimensions and their use in characterising filtration, *IChemE Research Event*, 7-8 April, Newcastle

FRACTAL DIMENSIONS OF COMPUTER SIMULATED AGGLOMERATES

ES Tarleton and STH Brock
Department of Chemical Engineering, Loughborough University,
Loughborough, Leics., LE11 3TU, UK

As a step towards classifying the fractal nature of filter cakes, particle agglomerates have been grown onto seed particles using a new computer simulation. The agglomeration process was controlled by varying the amount of diffusion influence in the growth mechanism. The perimeter and density fractal dimensions of simulated agglomerates comprising up to 800 particles were measured using three different automated techniques. The perimeter dimension was found to increase markedly with larger diffusion influence, whilst the density of the structure, measured using the enclosing circles and radius of gyration methods, decreased as the level of diffusion increased. The importance of these results to the characterisation of cake filtration processes is discussed.

Keywords: Agglomeration, filtration, fractals, structure

INTRODUCTION

Much work has been performed in recent years with regard to fractals [1]. Although the majority has apparently been of little practical use, some work has been performed to investigate the relationships between fractal dimension and the characteristics of particulate systems [2-4]. This is of particular interest here as the cakes formed during so called 'dead-end' filtration may be thought of as growing particle agglomerates on a filter surface.

The data in this paper show a summary of the results from an ongoing research project which examines the relationship between fractal dimension and the filtration characteristics of suspensions and filter cakes. The work aims to identify better methods of characterising cake structure and thus provide more accurate filtration analysis and scale-up procedures than are currently available. The importance of particle motion to the structure of agglomerates, and hence filter cakes is highlighted.

COMPUTER MODEL DEVELOPMENT AND AGGLOMERATE ANALYSIS

Although the computer program used to create agglomerates is not described in detail here, it essentially comprised a set of modular routines capable of defining, growing and analysing agglomerates for set numbers of particles with given size distributions. Each of the 15 modules in the program was constructed and tested prior to insertion in the main program, and the main program was further tested with relatively small agglomerates of large particles to ensure correct growth and analysis. Agglomerate growth was simulated in both two and three dimensional space using up to 800 circular or spherical particles where the degree of diffusion influence was varied over the range 0% (i.e. pure ballistic motion) to 100% diffusion in 5% increments. Attachment of particles was determined from geometric considerations whilst the variable diffusion was achieved by allowing particles to move through the control space in either a pre-described or random direction dependent on the proportion of diffusion influence. An example of a 2-D agglomerate with 50% diffusion is shown in Figure 1.

The 2-D agglomerate structures were analysed by three different techniques. The first, the structured walk, may be likened to measuring the agglomerate perimeter with a pair of "dividers" at sequentially varied steplengths. The gradient measured from a log-log plot of steplength (λ) against measured perimeter (P), known as a "Richardson plot", yields the (structural) perimeter fractal dimension (D_p) as given by eqn. (1) [3]

$$P = \lambda^{(1-D_p)} \tag{1}$$

With the enclosing circle analysis technique, the area (A) of the particles contained within progressively larger circles is measured and a density fractal dimension (D_d) determined from the gradient of the mass (M)-length (l) relationship according to

$$M \sim l^{D_d} \tag{2}$$

In general, dense, more closely packed structures have a density fractal dimension approaching the Euclidean dimension, whilst less dense structures exhibit lower fractal dimensions.

Agglomerates were also analysed using a radius of gyration (R_g) technique where

$$R_g = \sqrt{\frac{I_g}{A}} \tag{3}$$

and I_g is the second moment of area for the agglomerate. A series of radius of gyrations are calculated by progressively increasing the number of particles counted in the agglomerate up to the maximum number of particles. The radius of gyration is plotted against the number of particles on a log-log scale and the density fractal dimension is given by the reciprocal of the slope of the resultant straight line.

The 3-D agglomerates were analysed through 2-D projections in the case of structured walk and by substituting volume for area in the cases of enclosing circle and radius of gyration.

RESULTS

In order to examine the influence of diffusion on agglomerate/cake growth, a total of 840 agglomerrates have been built and analysed in both 2-D and 3-D using the computer program described. For the 2-D case agglomerates containing up to 800 circular particles were grown with varying degrees of diffusion influence between 0 and 100% (i.e. a total of 420 simulations). Due to the statistical nature of agglomerate growth it was necessary to impose error levels to identify wholly representative results, and for the current purpose, structured walk analyses were considered valid when $r^2<0.10$, enclosing circle analyses when $r^2<0.02$ and radius of gyration analyses when $r^2<0.002$. Thus, each of points on Figure 2-4 represents an average of the results within the respective error limits.

When the structured walk technique was used to analyse the range of simulated agglomerates a change in the perimeter fractal dimension was observed. A gradual variation was seen at lower diffusion levels with a steeper variation becoming apparent at approximately 70% diffusion (see Figure 2). The fractal dimension increased from 1.25 to 1.45 for the agglomerates analysed, with the more rapid change corresponding to a fractal dimension of 1.27. The changes in agglomerate structure were also visually apparent with agglomerates built using lower levels of diffusion appearing more dense.

The enclosing circle analysis gave numerical values to the observed changes in particle density within the structures. Figure 3 shows that the density fractal dimension decreased from a maximum value close to the Euclidean dimension of 2.00 to 1.75 as the level of diffusion influence increased from 0% to 100%. A pronounced change in gradient was again observed at approximately 70% diffusion and a density fractal dimension of 1.94.

In Figure 4 the radius of gyration analysis shows similar trends to the enclosing circle data presented in Figure 3. The density fractal dimension is again seen to decrease from a value close to the Euclidean dimension to 1.74 with varying diffusion influence, the steeper change in gradient again occurring at approximately 70% diffusion and a density fractal dimension of 1.91.

Although no data for 3-D simulations are presented in this paper, the general trends observed for 2-D simulations and analyses were repeated for corresponding 3-D simulations involving spheres, rather than circles. The sharp changes in fractal dimension seen at levels of diffusion approaching 70% were not repeated as more gradual changes in the fractal dimensions were recorded. The 3-D data will be the subject of a future paper.

CONCLUSIONS

The results presented in this paper are a product of the first year in a planned three year research program aimed at examining and characterising the fractal nature of filter cakes. The results obtained to date in both 2-D and 3-D have shown how the degree of diffusion influence can alter measured fractal dimensions, with steeper changes being observed in the region of 70% diffusion for 2-D. The next stage of the project requires comparison of the computer simulated agglomerates with "real" agglomerates obtained by the sampling of filter cakes from a well controlled filtration system.

Ultimately, it is hoped that the fractal techniques outlined here will offer a better way of characterising cake structure and filtration characteristics. Currently porosity and resistance measurements are gathered and correlated against pressure to characterise the compressibility of filtration systems and provide scale-up parameters. It is known that even small changes in measured porosity can significantly alter filtration rates [5] and it may prove beneficial to use another, more accurate, descriptor of cake structure such as the fractal dimension.

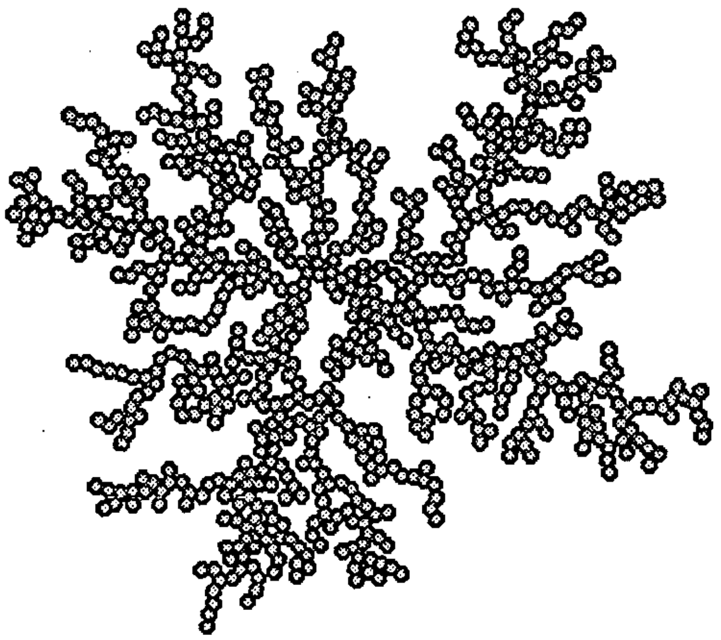


Figure 1: Example of agglomerate simulation with 50% diffusion

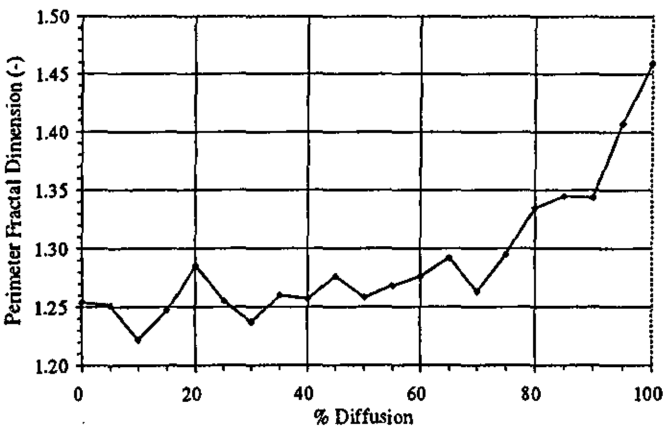


Figure 2: Structured walk analysis for 2-D simulations

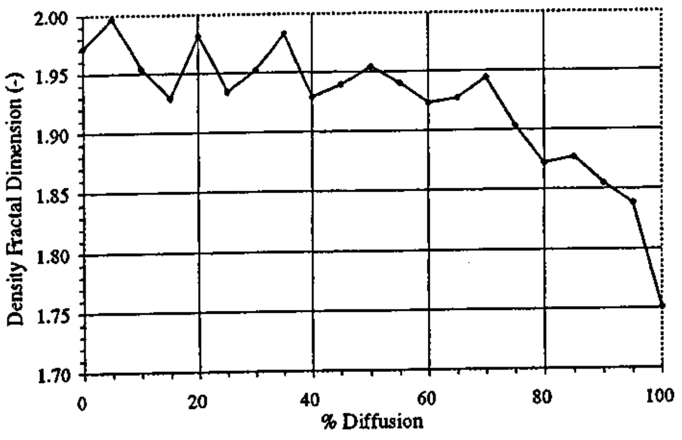


Figure 3: Enclosing circle analysis for 2-D simulations

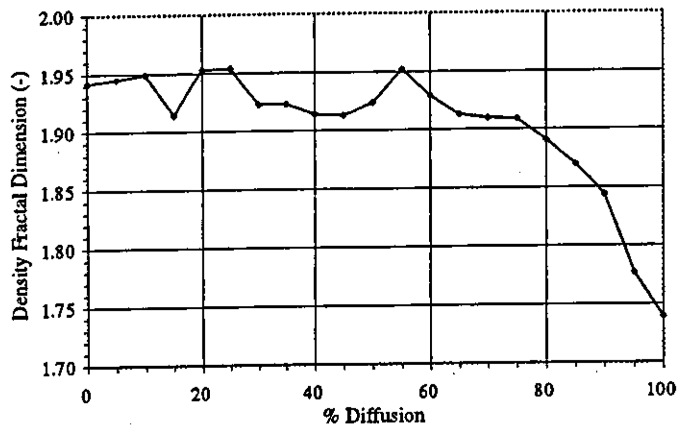


Figure 4: Radius of gyration analysis for 2-D simulations

NOMENCLATURE

A	=	Area	(m^2)
D_d	=	Density fractal dimension	(-)
D_p	=	Perimeter fractal dimension	(-)
I_g	=	Second moment of area	(m^4)
l	=	Length	(m)
M	=	Mass	(kg)
P	=	Perimeter length	(m)
r	=	Regression correlation coefficient	(-)
R_g	=	Radius of gyration	(m)
λ	=	Steplength	(m)

REFERENCES

1. B.H. Kaye, *A Random Walk through Fractal Dimensions*, VCH, Weinheim, Germany, 1994.
2. E. Schmidt, Experimental investigations into the compression of dust cakes deposited on filter media, 1995, Filt. and Sep., 32(8): 789-793.
3. G.A. Bayles, G.E. Klinzing and S-H Chiang, 1987, Determination of the fractal dimensions of coal filter cakes and their relationship to cake permeability, *Part. Sci. & Tech.*, 5: 371-382.
4. P. Meakin, 1988, Models for colloidal aggregation, *Ann. Rev. Phys. Chem.*, 39: 237-267.
5. S.A. Willmer, E.S. Tarleton and R.G. Holdich, 1995, Influence of particulate and process variables in compressible cake filtration, *Proc. Filtech Conf.*, pp149-159, Filtration Society, Karlsruhe, Germany.

THE FRACTAL PROPERTIES OF TWO & THREE DIMENSIONAL COMPUTER SIMULATED AGGLOMERATES

E.S. Tarleton and S. Brock

Department of Chemical Engineering, Loughborough University,
Loughborough, Leics., LE11 3TU, UK

Abstract

As a step towards classifying the fractal nature of filter cakes, particle agglomerates have been grown onto seed particles in both two and three dimensions using a new computer simulation. The agglomeration process was controlled by varying the amount of diffusion influence in the growth mechanism. The perimeter and density fractal dimensions of simulated agglomerates comprising up to 800 particles were measured using three different automated techniques. The perimeter dimension was found to increase with larger diffusion influence, whilst the density of the structure, measured using the enclosing circles and radius of gyration methods, decreased as the level of diffusion increased. The importance of these results to the characterisation of cake filtration processes is discussed.

Introduction

Much work has been performed in recent years with regard to fractals (Kaye, 1994). Although the majority has apparently been of little practical use, some work has been performed to investigate the relationships between fractal dimension and the characteristics of particulate systems (Bayles et al, 1987; Schmidt, 1995). This is of particular interest here as the cakes formed during so called 'dead-end' filtration may be thought of as growing particle agglomerates on a filter surface.

The data in this paper show a summary of the results from an ongoing research project which examines the relationship between fractal dimension and the filtration characteristics of suspensions and filter cakes. The work performed aims to identify better methods of characterising cake structure to provide more accurate filtration analysis and scale-up procedures than are currently available. The data provided in this paper highlights the importance of particle motion to the structure of agglomerates, and hence filter cakes.

Computer model development and agglomerate analysis

The computer program used to create and analyse agglomerates comprised a set of modular routines capable of defining, growing and analysing agglomerates. Agglomerate growth was simulated in both two and three dimensional space using 800 circular or spherical particles where the degree of diffusion influence was varied over the range 0% (i.e. pure ballistic motion) to 100% diffusion in 5% increments. An example of both 2-D and 3-D agglomerates with 50%

diffusion are shown in Figure 1. The 3-D agglomerate (on the right hand side of Figure 1 has been rendered using "ray-tracing" software to enable the depth of the structure to be seen more clearly.

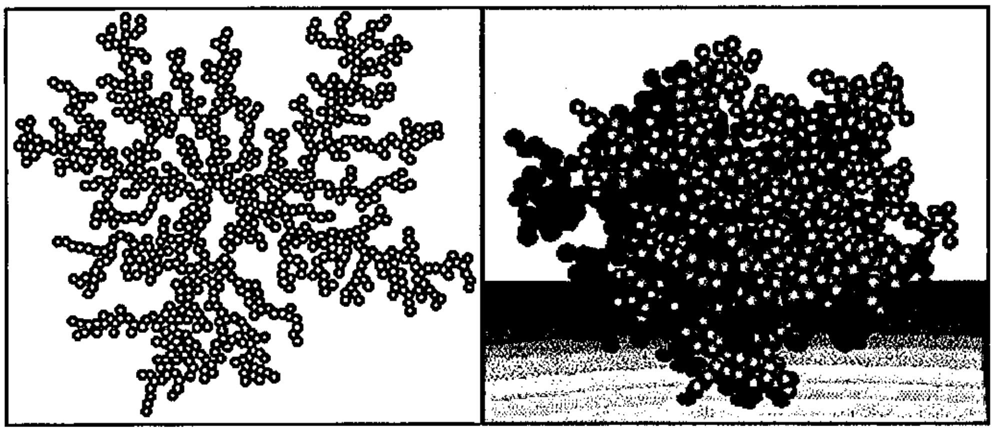


Figure 1: Example of 2-D and 3-D agglomerates (50% diffusion)

In order to examine the influence of diffusion on agglomerate/cake growth, a total of 840 agglomerates have been built and analysed in both 2-D and 3-D using the computer program. For 2-D simulations, agglomerates containing up to 800 circular particles were grown with varying degrees of diffusion influence between 0 and 100% (i.e. a total of 420 simulations). In the case of the 3-D simulations, the circular particles were substituted by spheres. The perimeter and density fractal dimensions of all the simulated agglomerates were determined using the three different automated techniques of structured walk, enclosing boundary (circle or sphere) and radius of gyration respectively.

The structured walk technique measures the perimeter ruggedness of an agglomerate by measuring the perimeter with a pair of virtual 'dividers' set at progressively varying steplengths. As the steplength is decreased, so more of the detail of the perimeter becomes apparent. The measured perimeter generally shows a log-log relationship with the steplength. The enclosing boundary technique involves encompassing an agglomerate with progressively larger circles/spheres and measuring the area/volumes of the particles contained therein. The second density fractal dimension (i.e. radius of gyration) is measured by evaluating the second moment of area/volume of an increasing number of particles within the agglomerate where the seed particle of the agglomerate is the starting point. Both enclosing circle and radius of gyration techniques use logarithmic relationships to calculate the density fractal dimension. Due to the statistical nature of agglomerate growth it was necessary to impose error levels to identify wholly representative results, and for the current purpose, structured walk analyses were considered valid when $r^2<0.10$, enclosing circle analyses when $r^2<0.02$ and radius of gyration analyses when $r^2<0.002$. Thus, each of the points on Figures 2-4 represents an average of the results within the respective error limits.

Results

Figure 2 shows that when the structured walk technique was used to analyse the simulated agglomerates, a change in the perimeter fractal dimension was observed. For the 2-D agglomerates, a gradual variation was seen at lower diffusion levels with a steeper variation becoming apparent at approximately 70% diffusion. The fractal dimension increased from 1.25 to 1.45 for the agglomerates analysed, with the more rapid change corresponding to a fractal dimension of 1.27. The analysis of 3-D structures showed similar trends to 2-D, although the changes in perimeter ruggedness were less pronounced. The results for the 3-D structured walk analysis have been obtained by averaging the perimeter fractal dimension for three projections of each agglomerate (arbitrarily defined here as the front, side and top views). The fractal dimension was again seen to change more rapidly around 70% diffusion, but only climbing from 1.19 to a maximum of 1.25, with a fractal dimension of 1.21 corresponding to 70% diffusion.

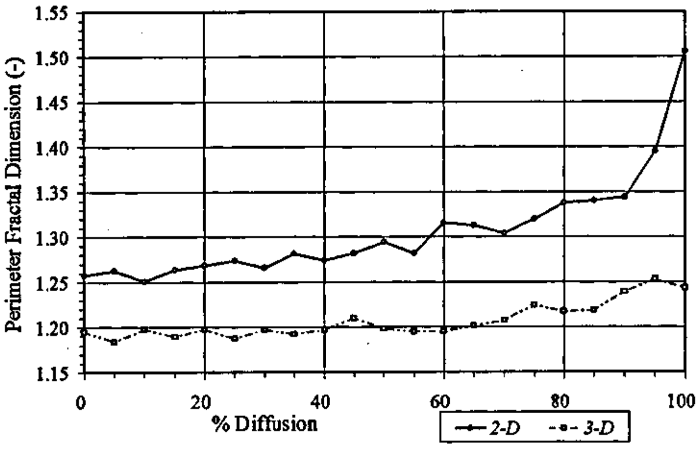


Figure 2: Structured walk analysis for 2-D and 3-D simulations

Figure 3 shows how density fractal dimensions decreased as the level of diffusion influence increased from 0% to 100%. Over this range the enclosing circle fractal dimension reduced from a maximum value close to the Euclidean dimension of 2.00 to 1.75, whilst the radius of gyration analysis shows similar trends, with the density fractal dimension decreasing from 1.94 to 1.74 with varying diffusion influence. Both of the methods showed a pronounced change in gradient at approximately 70% diffusion corresponding to a density fractal dimension of 1.94 for the enclosing circle method and 1.91 for the radius of gyration technique.

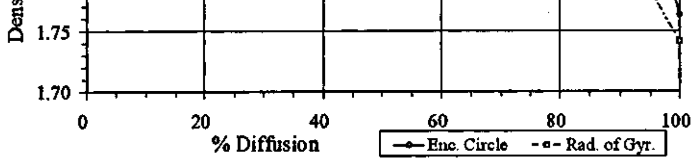


Figure 3: Enclosing circle and radius of gyration analyses for 2-D simulations

The density analyses of the 3-D structures showed similar trends to the results obtained for 2-D simulations whereby the density fractal dimension decreased with increasing diffusion influence. However, the sharp decrease corresponding to 70% diffusion was not observed. Instead, both the density fractal dimensions decreased in a relatively steady manner. The results of the radius of gyration analyses showed a steeper change in fractal dimension than shown with the enclosing sphere technique. The density fractal dimensions for the latter fall from 3.05 (essentially the Euclidean dimension) to a minimum of 2.50 with an increase in diffusion influence from 0% to 100%. The enclosing sphere procedure, meanwhile, showed only a decrease from 2.72 to 2.46 over the same range. The comparison between the two methods is shown in Figure 4.

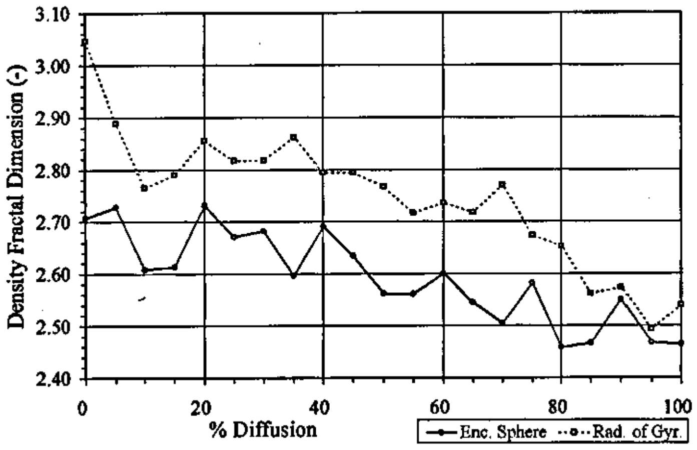


Figure 4: Enclosing circle and radius of gyration analyses for 3-D simulations

Conclusions

The results presented in this paper are a product of the first year in a planned three year research program aimed at examining and characterising the fractal nature of filter cakes. The results obtained to date show how the degree of diffusion influence can alter measured fractal dimensions, with steeper changes being observed in the region of 70% diffusion for 2-D. The next stage of the project requires comparison of the computer simulated agglomerates with "real" agglomerates obtained by the sampling of filter cakes from a well controlled filtration system. Ultimately, it is hoped that the fractal techniques outlined here will offer a better way of characterising cake structure and filtration characteristics and scale-up parameters.

References

1. B.H. Kaye, *A Random Walk through Fractal Dimensions*, VCH, Weinheim, Germany, 1994.
2. G.A. Bayles, G.E. Klinzing and S-H Chiang, 1987, *Part. Sci. & Tech.*, 5: 371-382.
3. E. Schmidt, 1995, Filt. and Sep., 32(8): 789-793.

THE USE OF FRACTAL DIMENSIONS IN FILTRATION

S.T.H. Brock & E.S. Tarleton
Department of Chemical Engineering, Loughborough University,
Loughborough, Leicestershire, LE11 3TU, UK
email : s.t.h.brock@Lboro.ac.uk

To enable the fractal characterisation of structures in filtration systems, a computer program has been written to simulate and measure the characteristics of particles growing on a filter surface in two dimensions. Two analysis techniques have been used to give both a roughness factor (using a perimeter fractal dimension) and density factor (using an enclosing circle fractal dimension) for the simulations. These techniques have been used to characterise both the overall structure of the simulated cake and the interstitial spaces between particles. Results have shown a correlation between simulation parameters and fractal dimension.

Keywords: Fractal dimension, filtration, simulation

INTRODUCTION

Much work has been performed in recent years with regard to fractals[1]. Although most has apparently been of little practical use, some work has been performed to investigate the relationship between fractal dimension and the characteristics of particulate systems[2-4]. This area which is of interest here, has been examined to determine the role of fractal dimension in characterising dead-end filtration systems.

The data in this paper show a summary of the results from an ongoing research project examining the relationship between fractal dimension and the filtration characteristics of suspensions and filter cakes. The work aims to identify better methods of characterising cake structure and thus provide more accurate filtration analysis and scale-up procedures than those currently available. The importance of particle motion to the structure of agglomerates, and hence filter cakes is highlighted.

SIMULATION OF CAKE GROWTH

A number of simulations have previously been performed, using a variety of methods. An example is that of Giona and Patierno, 1992[5], who used a Monte Carlo approach to model the build up of a particle structure on a surface. A typical structure was built on a square lattice (on-lattice simulation) and used particles of unit size, that is to say each particle only occupied one square of the lattice. An example of an on-lattice simulation is shown in Figure 1(a). The particle's movement is restricted to adjacent squares in the lattice, i.e. it can only move up, down left or right for any given step. The movement of the particle is governed by a downward probability, with values between 25% (particle is free to move in any of the four possible directions) and 100% (the particle will always move in the downwards direction).

In the current study, cake structures have been formed on a simulated filter surface by particles moving in two dimensions with a given downward and sticking probability. The downward probability is a value determining the likely direction of movement of the particle, whilst the sticking probability is the probability that a particle will stick to another on contact. In the simulation, particles are released from a random position at the top of the filter cell and allowed to move within the restrictions of the cell through a number of discrete steps until contact is made with another particle or the base of the cell containing the filter surface.

In contrast to many previous simulations, the model developed by the authors is an off-lattice simulation, that is to say with maximum freedom, a particle is allowed to move anywhere within a 360° radius. An example of an off-lattice simulation is shown in Figure 1 (b). The particles used in the simulation are also circular, as opposed to single pixels, which better imitates particles in a filtration system. The motion of particles in the free space of the cell is controlled by the downward probability which is defined at the beginning of the simulation as a number between 0 and 100%. Zero per cent downward probability allows the particle to move in any direction from the current position. One hundred per cent downward probability allows the particle to only move vertically downwards. The value of downward probability is infinitely variable between these two extremes, though in practice, intervals of 5% were used for the simulations.

The freedom of movement of a particle is defined by equation (1):

$$F = (100 - P) / 100 * 360 \tag{1}$$

where P is the downward probability and F the freedom of movement in degrees. This means that a particle can move in an arc of F/2 degrees either side of the vertically downwards direction. The direction in which a particle moves is defined by equation (2):

$$\theta = \Psi(F) - (F / 2) \tag{2}$$

where θ is the actual direction (degrees from the vertical) in which the particle moves and $\Psi(F)$ simply gives a random value between zero and F itself. The downward probability is fixed for the entire simulation, but the direction for a particle is re-calculated after each successive step as the simulation progresses.

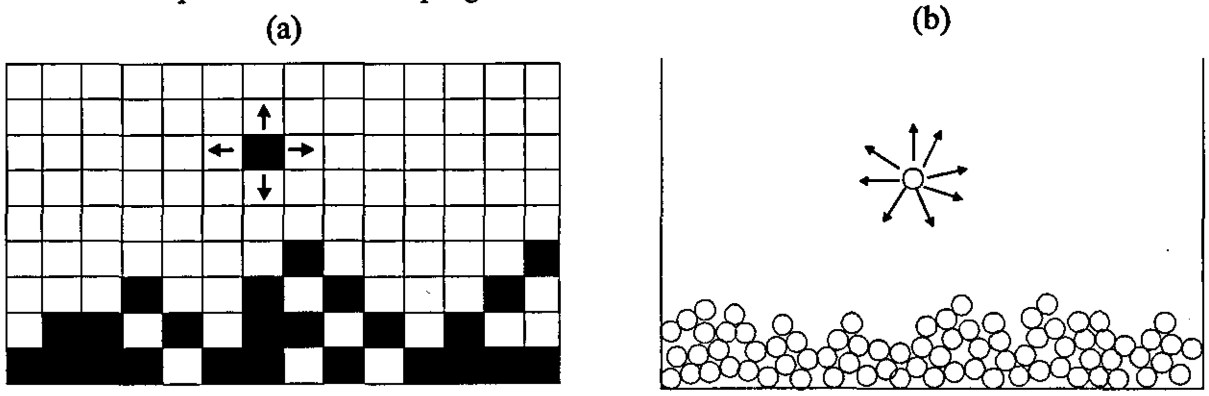


Figure 1 - Typical on-lattice (a) and off-lattice (b) simulations

When a particle hits a wall of the cell, a new direction is calculated until the particle moves in a direction away from the wall. In this way a particle will move around the cell until contact is made with the base or another particle. When a particle touches the floor of the cell containing the filter medium, its position is instantly fixed and recorded by the program. If, however, a particle contacts another particle already at the surface of the growing cake, the sticking probability must be considered. This is another simulation parameter that is fixed at the start of the program, and calculated for each particle collision. As the falling particle contacts another, a random number between 0 and 1 (i.e. 0 and 100%) is generated. If the number is less than or equal to the sticking probability then the particle will remain at rest where it lands. If, the random value is greater than the sticking probability, however, the landing particle will roll over the particle(s) in the cake, as indicated in Figure 2.

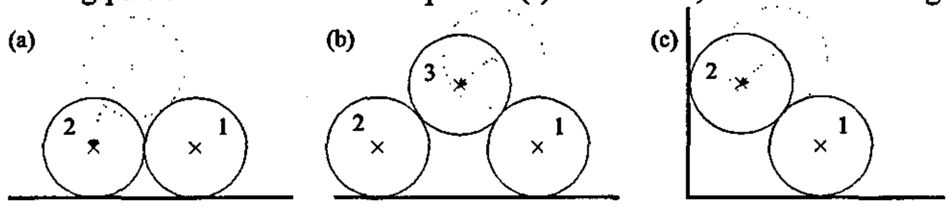


Figure 2 - Particles rolling to rest at the base of the filter cell

A particle is considered to rest if it has two points of contact, or contact with the filter medium. Figure 2 a) shows the rolling mechanism if the falling particle (2) lands on top of another particle (1) and rolls to the floor unhindered. Figure 2 b) shows the mechanism if the falling particle (3) contacts another (2) after contacting the initial particle (1). *Figure 2* c) shows the mechanism if particle (2) has the wall as a point of contact after rolling off particle (1). The falling particle is shown rolling in one particular direction in *Figure 2*, however, should it land on the other side of a target particle, it will obviously roll in the opposite direction.

As well as being able to alter the downward and sticking probabilities, the size of the particles used in the simulation could be set. A size range was specified using a minimum and maximum particle radius, and choosing the size of each particle between the two. Coupled with the other probability parameters, this offers a wide scope for varying the simulations.

Using the method described, cakes comprising up to 10^3 particles have been built in a virtual filter cell. Simulations have been performed with varying combinations of downward and sticking probabilities, twenty simulations being generated for each pair of parameters. Using powerful PC's allows a large number of simulations to be carried out and analysed in a short time, enabling a broad picture of cake properties to be built up.

MEASURES OF CAKE PROPERTIES

After a cake structure has been built, it can be analysed using a variety of methods. Both the top surface and the overall structure of the cake can be categorised. The top surface has been analysed using the structured walk method to give a roughness fractal dimension. The overall structure of the cake is characterised by its porosity, which may be measured in two directions, vertically (base to cake surface) and horizontally (wall to wall) to enable a detailed picture of cake structure to be built up.

As well as the overall structure of the cake, the interstitial spaces within the cake may be analysed. Two fractal dimensions can be found for the pore spaces. Firstly a roughness dimension similar to that for the surface, and secondly a density fractal dimension, measured using the enclosing circle technique. The latter shows how the area of the pore space is distributed. Up to 99 pore spaces could be analysed for each set of simulation parameters.

Using these methods of analysis, correlations were made between the simulation parameters and cake properties. Each of the analysis techniques was used on all the cakes simulated for each set of parameters and average values taken to represent that particular property for the cakes concerned.

FRACTAL DIMENSION

A fractal dimension can be measured for the surface of the cake and the interstitial spaces between the particles within the cake. The former is characterised using the structured walk (roughness fractal dimension) technique and the latter using both the structured walk and enclosing circle (density fractal dimension) techniques.

A structured walk is performed by stepping over the surface of the filter cake using virtual chords of varying lengths. As the steplength decreases, so more of the detail of the cake is shown and the measurable perimeter increases. The relationship between steplength and perimeter is given by equation (3):

$$P = \lambda^{(1-D_S)} \tag{3}$$

where P is the perimeter measured using a steplength λ. By plotting the perimeter obtained against steplength on a log-log scale, the fractal dimension, D_S can be calculated; the slope of the plot having a value of $1-D_S$. The walk can be carried out in both directions (i.e. left to right and right to left) over the surface of the cake and an average of the two results taken. Good agreement between the two methods was observed in the majority of cases, although the technique can be sensitive to large fissures etc. in the cake structure. A high value for the fractal dimension indicates that the surface of the cake is rougher, as the perimeter increases at a greater rate with decreasing steplength. The structured walk technique applied to the surface of a typical simulated filter cake is shown in Figure 3, with the results of the analysis for a left-to-right structured walk.

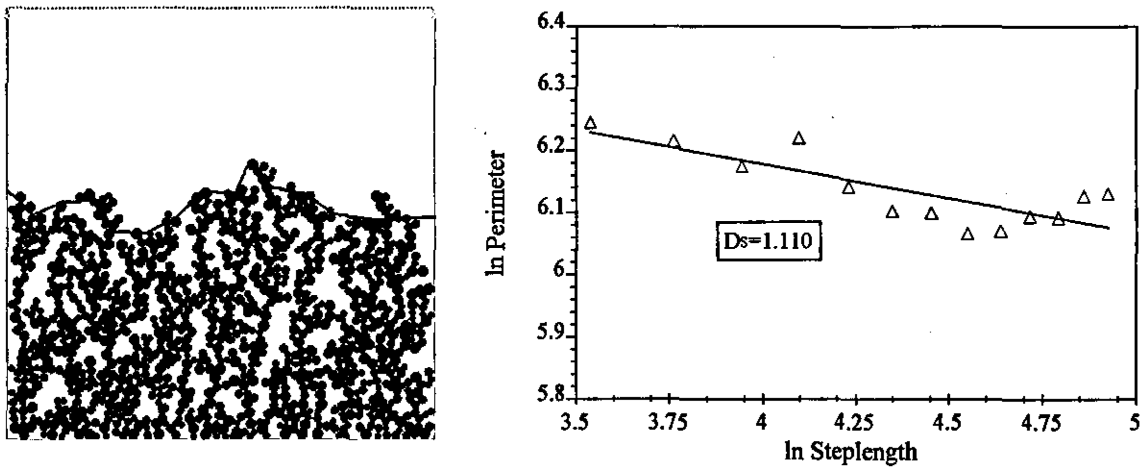


Figure 3 - Structured walk technique for the surface analysis of a simulated filter cake.

Figure 4 shows that the perimeter fractal dimension rises steadily as the sticking probability is increased. As would be expected, at low values of sticking probabilities, the fractal dimension is close to the Euclidean dimension (1.00), but increases as the particles in the simulation become more likely to stick on contact with another particle. These results are also visually noted in the overall structure of the cake. Simulations with a low sticking probability show a smooth surface as the particles roll to rest, whereas large dendritic structures are formed as particles stick together without rolling. This dendrite formation is also seen in the simulations of Giona and Patierno (1986)[5], which shows an increasingly open structure with low downward probability, although as Figure 4 shows, the downward probability has less of an effect on the structure than sticking probability. This open structure has also been seen in previous work involving the growth of agglomerates onto a seed particle (Tarleton and Brock, 1997)[6], which showed that as the motion of the particles is changed from a straight line (ballistic) to random walk (diffusive) the structure of agglomerates built is more open (less densely packed), with a higher perimeter fractal dimension.

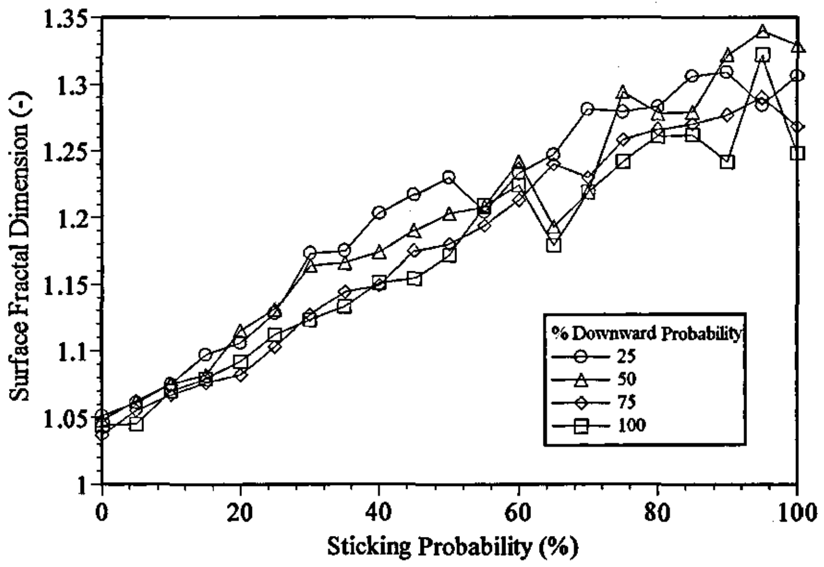


Figure 4 - Structured walk analysis of simulated filter cakes

The structured walk was also used for analysing the interstitial spaces between particles in the simulated cakes. The technique required a pore space to be isolated by the computer program, and a subsequent structured walk around the outside of the space. In a similar way to the surface analysis, the fractal dimension was found by plotting the perimeter obtained against steplength used. The resulting dimension gives the roughness factor for the space. The structured walk fractal dimension for pore spaces was found to vary between 1.15 and 1.20.

The enclosing circle fractal dimension was also used to characterise the interstitial spaces. This method gives a value for the density fractal dimension, which is a description of the distribution of area of the pore space around its centre of gravity. As its name suggests, the method requires parts of the pore space to be enclosed by circles. These circles have their centre at the centre of gravity of the pore space and radiate outwards towards the outer edge of the pore space. The amount of space occupied by the pore space (measured in pixels) is plotted against the radius of the circle (measured in pixels) enclosing that space, again on a log-log plot, where the fractal dimension is given by equation (4):

$$a = r^{D_E} \quad (4)$$

where a is the pore area enclosed by a circle of radius r. A log-log plot yields a slope of fractal dimension, D_E. Figure 5 shows the enclosing circle technique, with the method for determining the fractal dimension.

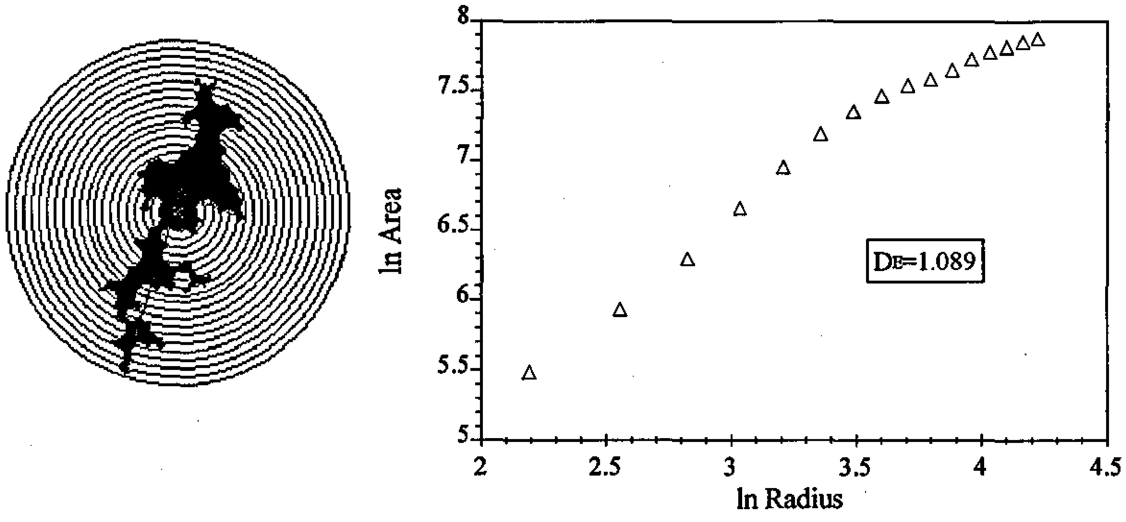


Figure 5 - Enclosing circle technique for the analysis of an individual pore space

POROSITY

The porosity of the cake was measured using three different methods, the second of which gave an overall picture of the structure for comparison between the simulation parameters.

- Method 1 : Cumulative vertical porosity. This technique involves taking slices of the cake from the base of the filter cell up to the top surface of the cake. The porosity is measured as the fraction of the slice occupied by voids for each height of slice.
- Method 2 : Vertical porosity profile. This technique analyses the porosity of the cake in the same way as the cumulative method, however, the porosity is measured for each small slice, rather than the height of all the slices together. Once all the slices have been analysed, an average is taken to give the porosity of the cake. It is this method that has been used to characterise the overall cake structure.
- Method 3 : Horizontal porosity profile. This technique uses horizontal slices across the cake, from the left hand wall across to the right to enable phenomena such as the wall effect to be observed. This effect has been studied by others such as Chan and Ng, 19867.

The vertical and horizontal methods are shown for the same cake in Figure 6. Figure 6 also shows a typical result for a vertical porosity profile. It should be noted that while the vertical profile is measured across the entire width of cell to show the porosity at that particular height, the horizontal profile only measures from the base of the filter cell up to the highest particle in the slice concerned, not the highest particle in the cake as a whole. This gives a truer value for the porosity at that point.

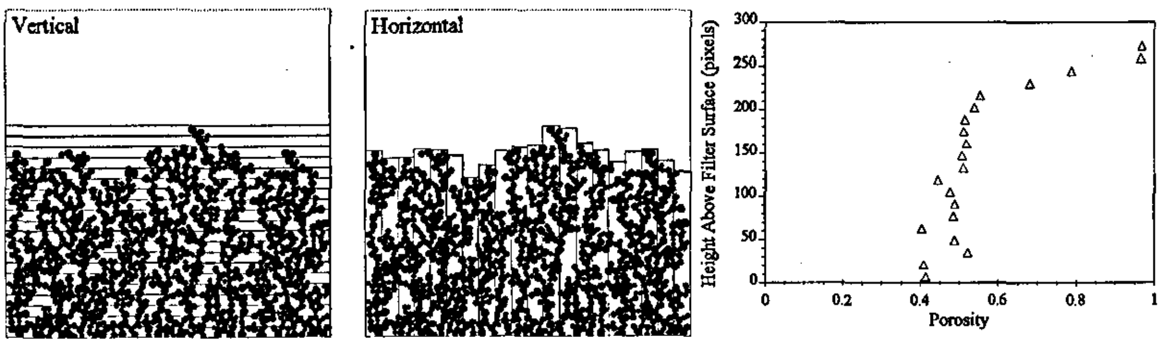


Figure 6 - Porosity measurement showing vertical and horizontal methods

Figure 7 shows how the porosity of the cake structure increases with increasing sticking probability. Again this is expected as the cake structure becomes more open as the individual particles stick to one another. The minimum value for the porosity agrees well with literature values for theoretical minimum porosity (Gray, 1968)[8]. The porosity increases from approximately 0.25 at 0% sticking probability to a maximum of approximately 0.70 as the sticking probability is increased to 100%. Again, a lesser effect is seen as the downward probability is increased, although the trend is more visible than that for the perimeter fractal dimension. As the downward probability is increased, so the structure of the cake becomes more compact and the porosity therefore decreases.

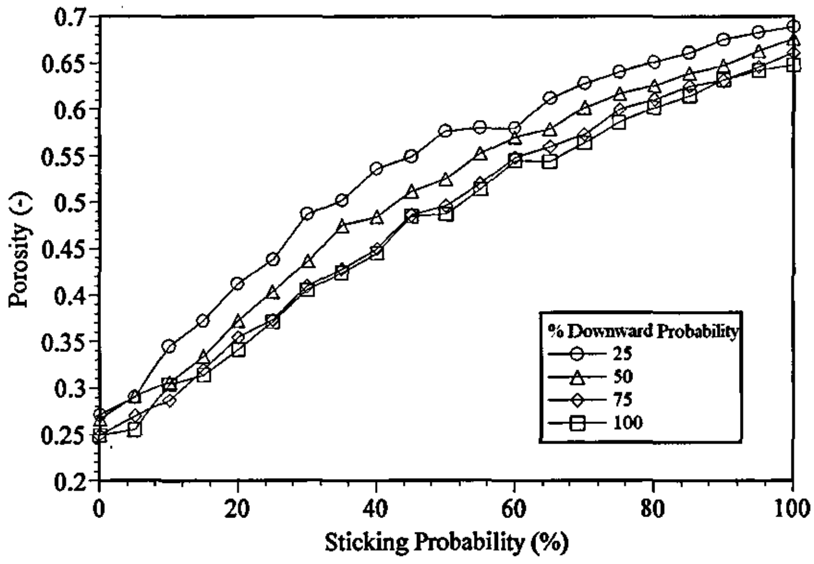


Figure 7 - Porosity analysis of simulated filter cakes

The methods of fractal and porosity analysis described here have been used to characterise a large number of simulated cakes with varying simulation parameters. The parameters investigated here were the downward and sticking probabilities of particles in the system. The results of these analyses are shown in Figures 4 and 7. The downward probability was seen to have less effect on the cake properties than the sticking probability, therefore the sticking probability was varied in 5% increments, whereas the downward probability was varied in 25% increments. The results show that as the sticking probability is increased, both the surface roughness fractal dimension and the porosity if the cake increase. The opposite is true of their relationship with downward probability, i.e. both the surface roughness fractal dimension and porosity of the cake decrease with increasing downward probability (for the same sticking probability).

CONCLUSIONS

The simulation parameters investigated to date have a quantifiable effect on the structure on the filter cakes built. Increasing the downward probability or decreasing the sticking probability makes the structure of a cake more porous, with larger dendrites forming. This will increase both the porosity and surface fractal dimension of the cake.

The porosities measured for the simulations fall into the range of porosities calculated from experimental work with calcium carbonate and talc suspensions, indicating that it should be possible to match the simulation parameters with physical constants for the systems concerned. The next stage of experimental work is to sample real filter cakes and analyse their internal structure using various fractal dimensions and compare these to three dimensional simulations.

NOMENCLATURE

a	Pore Area	pixels
D_E	Fractal Dimension (Enclosing Circle)	(-)
D_S	Fractal Dimension (Structured Walk)	(-)
F	Freedom of movement	°
P	Perimeter	pixels
r	Radius of circle	pixels
λ	Steplength of structured walk	pixels
θ	Direction of movement	°

REFERENCES

1. Kaye, B.H., 1994, *A Random Walk through Fractal Dimensions*, VCH, Weinheim, Germany
2. Schmidt, E., 1995, Experimental investigations into the compression of dust cakes deposited on filter media, *Filt. and Sep.*, 32(8): 789-793
3. Bayles, G.A., Klinzing, G.E. and Chiang, S-H., 1987, Determination of the fractal dimensions of coal filter cakes and their relationship to cake permeability, *Part. Sci. & Tech.*, 5: 371-382
4. Meakin, P., 1988, Models for colloidal aggregation, *Ann. Rev. Phys. Chem.*, 39: 237-267
5. Giona, M. and Patierno, O., 1992, Monte Carlo Simulation of Aggregation Process, *Chem. Eng. Comm.*, 121: 219 - 234
6. Tarleton, E.S. and Brock, S.T.H., 1997, Fractal dimensions of computer simulated agglomerates, *IChemE Research Event*, Chameleon Press, London, 473-476
7. Chan, S.K., Ng, K., 1986, Geometrical characteristics of a computer generated three dimensional packed column of equal and unequal sized spheres - With special reference to wall effects, *Chem. Eng. Comm.*, 48: 215-236
8. Gray, W.A., 1968, *The packing of solid particles*, Cox and Wyman Ltd., London

FRACTAL DIMENSIONS AND THEIR USE IN CHARACTERISING FILTRATION

S.T.H. Brock (s.t.h.brock@lboro.ac.uk) & E.S. Tarleton
Department of Chemical Engineering, Loughborough University,
Loughborough, Leics. LE11 3TU, UK

A computer program has been written to simulate cake growth and measure the characteristics of particles forming on a filter surface. Using this program, the fractal characterisation of the structure of filtration systems can be determined. Two techniques have been used to give both a roughness factor (using the perimeter fractal dimension) and density factor (using the enclosing circle fractal dimension) for the simulations. These techniques have been used to characterise both the overall structure of the simulated cake and the interstitial spaces between particles. Results have shown a correlation between simulation parameters and fractal dimension.

Keywords: Fractal dimension, filtration, simulation, structure, agglomeration

INTRODUCTION

The study of fractals has increased in recent years to a point where a number of applications have been suggested. Many of these, however, must be considered to be of little practical use. Although some work has been performed relating various fractal measures to system characteristics, few quantifiable results have been put forward.

The data presented in this paper show a summary of results from an ongoing research project examining the relationship between fractal dimensions and dead-end filtration characteristics of suspensions and filter cakes. In particular, the relevance of particle motion to the structure of filter cake build up is discussed. Work is being carried to determine improved methods of characterising cake structure in order to provide better filtration analysis and scale-up procedures than the current methodology and technology allow.

SIMULATION OF CAKE GROWTH

Cake growth has been simulated previously using a variety of methods. These include the Monte Carlo approach of Giona and Patierno (1986)[1] which created particulate structures on surfaces. Their "on-lattice" model, similar to that shown in Figure 1(a), allowed a descending particle to move in one of four restricted directions. In the current study, an "off-lattice" model has been used, which allows the particle to move in any random direction away from its current position. This motion is shown in schematically Figure 1(b).

The motion of particles in the authors' 2-D system is controlled by two simulation parameters, namely downward and sticking probabilities. The downward probability controls the likely direction of movement of the active particle, whereas the sticking probability determines the likelihood that the active particle will stick to the growing structure on contact. Circular particles of a uniform or non-uniform size are released from a random position at the top of a simulated filter cell and allowed to move within its confines. Under the influence of the downward probability the particles move until they contact the filter cake or the base of the filter cell. At the beginning of each simulation, the downward and sticking probabilities are set, along with the size distribution to be used. The number of particles to form the structure is also set. The current study is based on structures containing up to 10^3 particles. With the off-lattice model, the active particle has a potential arc of movement of 360° for its next step move. The extent of the arc is governed by the downward probability.

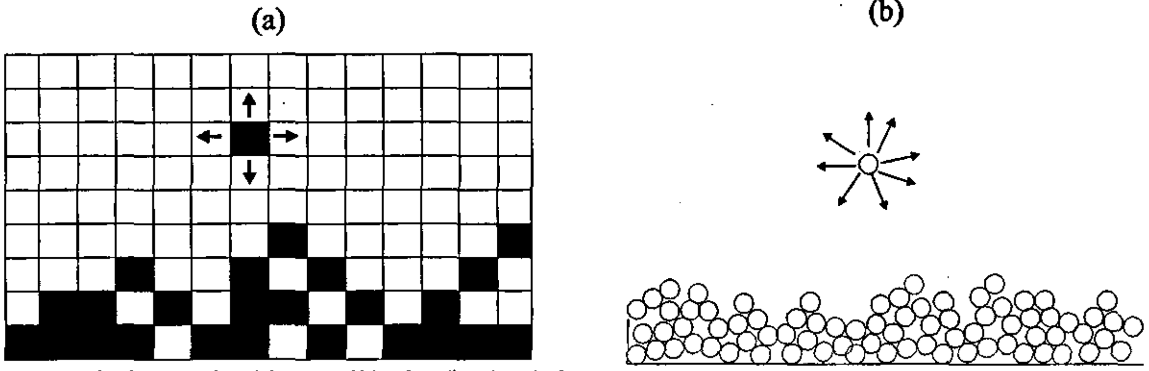


Figure 1 - Typical on-lattice (a) and off-lattice (b) simulations

The value of downward probability is set to a value between 0 and 100%. Zero percent downward probability allows the particle complete freedom of movement within the 360° arc, whilst one hundred percent downward probability forces the particle to move vertically downwards. The freedom or restriction of movement is defined by Equation (1):

$$F = (100 - P)/100 * 360 \tag{1}$$

where P is the downward probability and F the freedom of movement in degrees. Thus a particle can move anywhere within an arc of F/2 degrees either side of vertically downwards in a direction given by Equation (2):

$$\theta = \Psi(F) - (F/2) \tag{2}$$

where □ is the number of degrees from the vertical and □(F) is a function used to give a random value between zero and F. Although the value of downward probability is fixed, the direction of particle movement is re-calculated for each step.

If a particle hits the wall or top of the cell, a new direction is calculated so that the particle moves in a direction away from the boundary. The particle moves around the cell until it contacts either another particle already attached to the filter cake, or the base of the cell. In the latter case, the position of the landing particle is instantly fixed at the point of contact with the filter medium at the base. If, however, contact occurs with a fixed particle, the sticking probability is used to determine the behaviour of the landing particle. When contact occurs, a random number is chosen between zero and one (i.e. 0 and 100%). If the number chosen is less than or equal to the sticking probability defined at the start of the simulation, the particle remains at rest where it lands. If, on the other hand the random number is greater than the sticking probability, the particle rolls to rest, as in Figure 2.

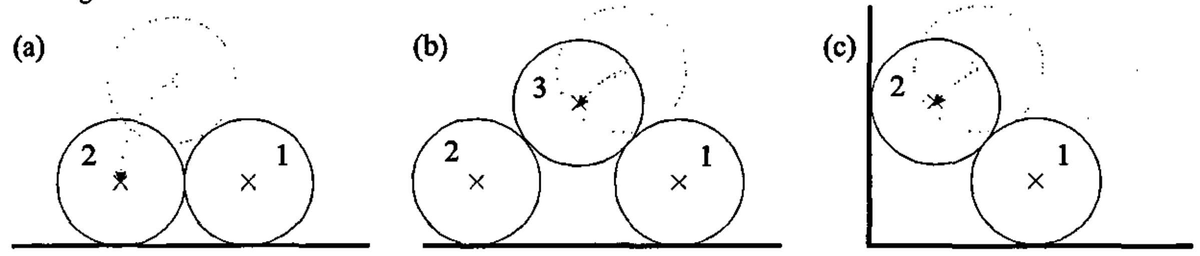


Figure 2 - Particles rolling to rest at the base of the filter cell

Figure 2(a) shows the mechanism for a falling particle (denoted 2) rolling to the left over a stationary particle (1). The falling particle will of course roll in the opposite direction if it lands on the other side of the fixed particle. The particle continues to roll (moving a fraction of a degree per step) until it contacts with the base of the cell. Geometric relations are used to determine the point at which the particle becomes fixed. Figure 2(b) shows how a falling particle (3) comes to rest touching two other particles, (1) and (2). The same movement is used, but in this case, the geometry determines when the falling particle touches particle (2) rather than the base. Finally, Figure 2(c) shows a falling particle (2) rolling off a fixed particle (1) until contact with the wall is made. In all three cases, the falling particle is deemed a rest when it has two points of contact with fixed objects. These sticking criteria apply only to simulations carried out in two dimensions, which is the work discussed here. For systems employing three dimensions, the algorithms for particle collisions are more complex.

Employing the methods outlined above and powerful PC's, a large number of simulations have been performed and analysed. Twenty simulations have been performed for each combination of downward and sticking probabilities. Combining these pairs of parameters with varying size distributions means that approximately 3,000 simulations have currently been completed and analysed.

ANALYSIS OF CAKE PROPERTIES

A number of methods have been used to analyse the computer generated cake structures. These methods are used to characterise both the surface and overall structure of the cakes. The surfaces of the simulated cakes were measured using the structured walk technique, yielding a roughness fractal dimension. The overall structure of the cakes were characterised using porosity measurements made in two planes, vertically upwards from the base of the cake and horizontally across from one wall of the cell to the other.

In addition to the above methods, the internal structure of the cake can be analysed. As the cake builds up, pore spaces are formed within the structure, which can themselves be characterised. In this case, two different fractal dimensions are obtained for the interstitial spaces. Firstly, the same method used for analysing the surface of the cake is used to give a roughness fractal dimension for the perimeter of the pore spaces. Secondly, the enclosing circle method is used to give a density fractal dimension. The latter gives an indication of how the area of the pore space is distributed around its centre of gravity for each set of simulation parameters (i.e. downward and sticking probabilities and size distribution).

Using these methods correlations have been made between the fractal and structural properties of the cakes and the simulation parameters used to create them. Each of the methods described below were used on all the cakes built (20 per set of simulation parameters) and an average taken to represent the particular property in question.

FRACTAL PROPERTIES OF SIMULATED CAKES

Fractal measures can be used to characterise both the surfaces of the simulated cake and the interstitial spaces between particles in the cake. The structured walk technique is used in both cases, whereas the enclosing circle technique is utilised solely to classify the interstitial spaces.

The structured walk technique involves stepping around the perimeter of the object in question (either the surface of the cake or an isolated pore space) using a pair of "virtual dividers". As the steplength of these dividers decreases, more detail is revealed and the perimeter increases. The relationship between the steplength of the dividers and the perimeter obtained is given by Equation 3:

$$P = \lambda^{(1-D_S)} \tag{3}$$

where P is the perimeter obtained from a structured walk using a steplength λ. A plot of perimeter against steplength on a logarithmic scale yields the structured walk fractal dimension, D_S, the slope of the plot being equal to $(1-D_S)$. The walk was carried out in both directions (i.e. left to right and right to left) across the surface of the cake, and an average of the two directions taken. Figure 3 shows a typical structured walk analysis (left to right) across the surface of a filter cake.

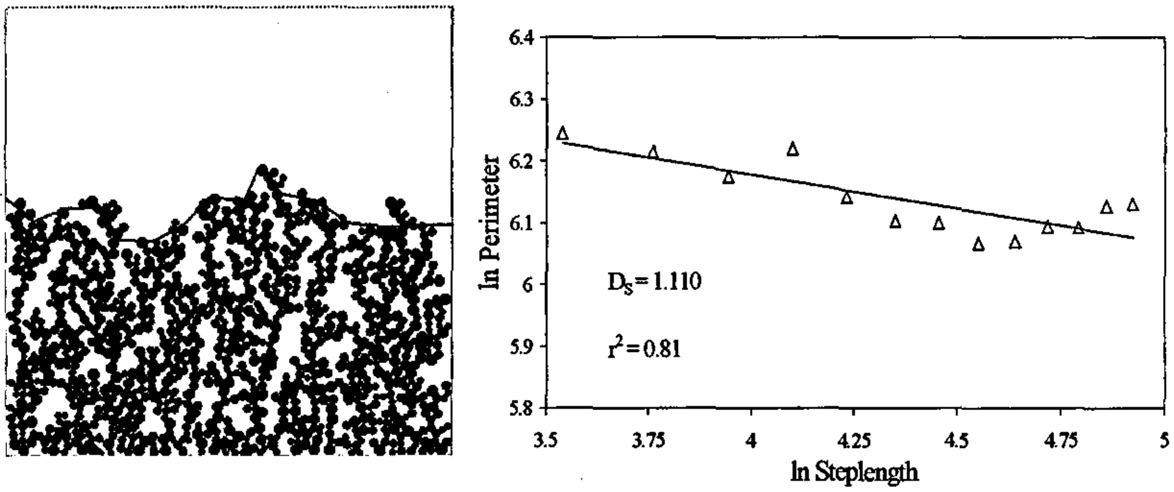


Figure 3 - Structured walk technique for the surface analysis of a typical simulated filter cake

The effect of system parameters on the fractal dimension and other measures was investigated as part of this work. It was found that as the sticking probability increased, the surface fractal dimension of the structure increased. This is the expected result, because, as the sticking probability increases, the structure takes on a more open structure. Similar results have been noted in the work of Giona and Patierno (1986)[1]. The structured walk fractal dimension for the surface of the simulated filter cakes rises from close to the Euclidean dimension of 1.0 at low values of sticking probability to approximately 1.3 at 100% sticking probability. These results are shown in Figure 4, which also shows that varying the downward probability has less of an effect on the structure of the simulated cake than the variation seen with changing sticking probability. As the downward probability is increased, the cake becomes more densely packed, giving a lower surface fractal dimension. Similar structural variation has been noted in previous work involving the growth of agglomerates onto seed particles (Tarleton and Brock, 1997)[2]. Simulations built in this way showed that as the motion of the active particle moves away from ballistic towards diffusion limited control (equivalent to a decrease in downward probability), the structure of the resulting structure is more open (less dense), again giving a higher fractal dimension.

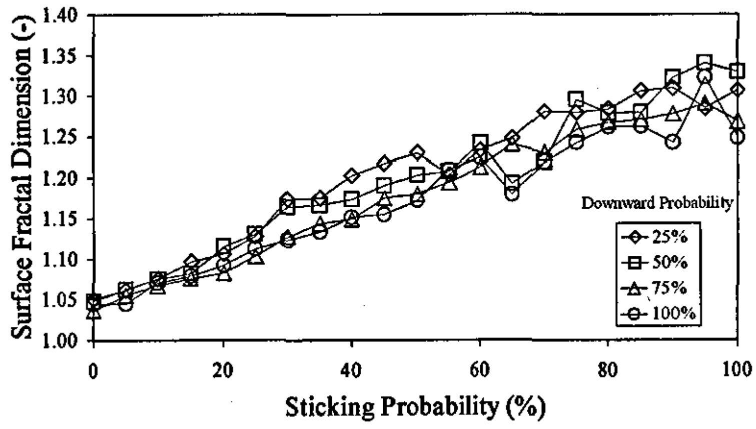


Figure 4 - Structured walk analysis of simulated filter cakes

The structured walk technique was also used to characterise the interstitial spaces between particles in the system. This analysis has shown that the perimeter fractal dimension for the individual pore spaces varies from 1.15 to 1.20. A second method for analysing the pore spaces was the enclosing circle method, which gives a density fractal dimension, rather than a roughness dimension. Again, the pore space is isolated, and the centre of gravity calculated. Then, circles of increasing size are created, and the area of pore space contained within those circles calculated. The relationship between circle enclosed pore area and radius is given by Equation 4:

$$a = r^{D_E} \tag{4}$$

The pore area is then plotted against circle radius again on a logarithmic plot giving a slope equal to the enclosing circle fractal dimension, D_E. The enclosing circle fractal dimension of individual pore spaces varied between 1.02 and 1.24.

POROSITY

As porosity is a principal characterising property of filtration, it was decided to incorporate a facility to measure porosity as part of the simulation package. The porosity of the system was obtained using three different methods:

- Method 1: Cumulative vertical porosity measurement. This method requires vertical "slices" of the cake to be taken from the base of filter cell up to the surface of the filter cake. The porosity was calculated simply as the fraction of the slice occupied by voids for each slice taken.
- Method 2: Vertical porosity profile - used as the defining value of porosity for each of the analyses. The vertical porosity profile uses the same principle as the first method, taking vertical slices up the height of the cake. The porosity of each of the slices is taken separately, and an average taken for all the slices. This average was used as the porosity value for the cake.
- Method 3: Horizontal porosity profile. As with the other two methods, the horizontal porosity was taken as the fraction of voids in each slice of the cake, but in the case of the horizontal porosity profile, slices were taken across the width of the filter cell. The porosity was then measured from the base of the cell to the height of the cake at that part of the cell. This method enables such phenomena as wall effect to be studied. The wall effect is an established cake property (see Chan and Ng (1986)[3], amongst others).

Both the vertical and horizontal methods are demonstrated in Figure 5 as well as a typical vertical porosity profile. As can be seen in Figure 5, the porosity at the top of the cake increases as the dendritic of the cake becomes more pronounced.

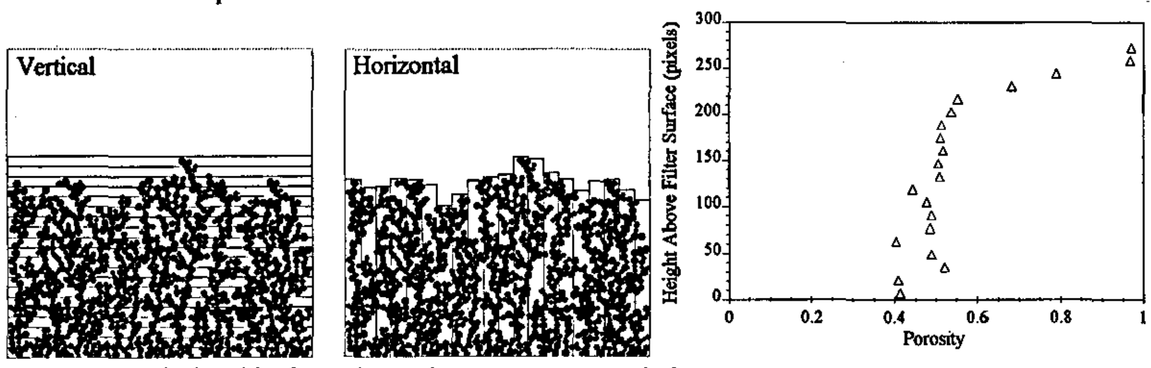


Figure 5 - Vertical and horizontal porosity measurement techniques

The porosity was examined with varying simulation parameters, and again the sticking probability had a greater effect on the porosity than the downward probability. As can be seen in Figure 6, the porosity rises from a minimum value of approximately 0.25 at 0% sticking probability (this agrees well with literature values (Gray, 1968)[4] to a maximum of 0.70 for 100% sticking probability. Dead-end filtration experiments with mineral suspensions of calite and talc have yielded average porosity values between 0.6 and 0.7. The sequential variation of porosity with downward probability is more apparernt in Figure 6 than in Figure 4.

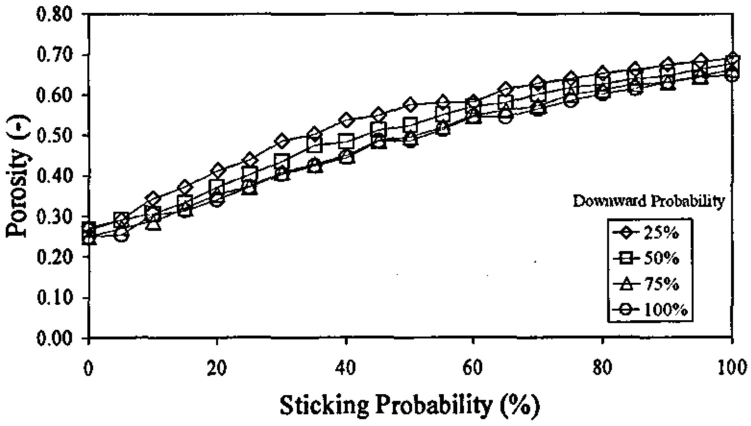


Figure 6 - Porosity analysis of simulated filter cakes

CONCLUSIONS

The results discussed in this paper show relationships between the simulation parameters and cake structure. It can be seen that both surface fractal dimension and porosity decrease with increasing sticking probability, due the more compact, less dendritic nature of the cake. As the downward probability is increased, however, the dendritic nature and compactness of the structure decrease, thereby decreasing both the surface fractal dimension and the overall porosity of the system.

As the porosity data falls within experimental values, the next stage of the work is to analyse the fractal dimensions and physical characteristics (such as porosity) of filtration systems through the use of electron microscopy and image analysis software. The computer program written for this work has the ability to measure these properties in the same way as for the simulated filter cakes. The next stage of the work is to simulate filter cakes in three, rather than two dimensions.

NOMENCLATURE

a	Pore area	pixels
D_E	Fractal dimension (Enclosing circle)	(-)
D_S	Fractal dimension (Structured walk)	(-)
F	Freedom of movement	°
P	Perimeter	pixels
r	Radius of circle	pixels
λ	Steplength of structured walk	pixels
θ	Direction of movement	°

REFERENCES

1. Giona, M and Patierno, O., 1992, Monte Carlo simulation of aggregation process, *Chem. Eng. Comm.*, 121: 219-234
2. Tarleton, E.S. and Brock, S.T.H., 1997, Fractal dimensions of computer simulated agglomerates, IChemE Research Event, Chameleon Press, London, 473-476
3. Chan, S.K. and Ng, K., 1986, Geometrical characteristics of a computer generated three dimensional packed column of equal and unequal sized spheres - With special reference to wall effects, *Chem. Eng. Comm.*, 48: 215-236
4. Gray, W.A., 1968, *The packing of solid particles*, Cox and Wyman Ltd, London