2010

A Continuous Mathematical Model of the One-Dimensional Sedimentation Process of Flocculated Sediment Particles

Sebastian Fernando Torrealba
University of Kentucky, storreal@gmail.com

ABSTRACT OF DISSERTATION

Sebastian Fernando Torrealba

The Graduate School

University of Kentucky

2010

A Continuous Mathematical Model of the One-Dimensional Sedimentation Process of Flocculated Sediment Particles

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Agriculture
at the University of Kentucky

By
Sebastian Fernando Torrealba

Lexington, Kentucky

Director: Dr. Richard C. Warner, Professor of Biosystems and Agricultural Engineering

Lexington, Kentucky
2010

ABSTRACT OF DISSERTATION

A Continuous Mathematical Model of the One-Dimensional Sedimentation Process of Flocculated Sediment Particles

A new continuous one-dimensional sedimentation model incorporating a new continuous flocculation model that considers aggregation and fragmentation processes was derived and tested. Additionally, a new procedure to model sediment particle size distribution (PSD) was derived. Basic to this development were three different parametric models: Jaky, Fredlund and the Gamma probability distribution (GPD) were chosen to fit three different glass micro-spheres PSDs having average particle sizes of 7, 25 and 35 microns. The GPD provided the best fit with the least parameters. The bimodal GPD was used to fit ten sediment samples with excellent results (< 5% average error). A continuous flocculation model was derived using the method of moments for solving the continuous Smoluchowski coagulation equation with fragmentation. The initial sediment PSD was modeled using a bimodal GPD. This new flocculation model resulted in a new general moments' equation that considers aggregation and fragmentation processes, which is represented by a system of ordinary differential equations. The model was calibrated using a genetic algorithm with initial and flocculated PSDs of four sediment samples and four anionic polyacrylamides flocculants. The results show excellent correlation between predicted and observed values ($R^2 > 0.9878$). A new continuous one-dimensional sedimentation model that resulted in a scalar hyperbolic conservation law was derived from the well-known Kynch kinematic sedimentation model. The model was calibrated using column tests results with glass micro-spheres particles. Two different glass micro-spheres particle size distributions (PSDs) were used with average diameters of 7 and 37 microns. Excellent values of coefficient of determination ($R^2 > 0.89$, except for one test replicate) were obtained for both the small and large glass micro-spheres PSDs. These results suggest that the proposed sedimentation model can be expanded to model the sedimentation process inside a sediment pond.

KEYWORDS: mathematical representation of particle size distribution, sedimentation theory, aggregation and fragmentation, flocculation model, sediment pond design.

Sebastian F. Torrealba

January 21st, 2010

A Continuous Mathematical Model of the One-Dimensional Sedimentation Process of Flocculated Sediment Particles

By

Sebastian Fernando Torrealba

Richard C. Warner
Director of Dissertation

Dwayne R. Edwards
Director of Graduate Studies

January 21st, 2010

RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of part may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the dissertation in whole or in part also required the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.

Name Date

DISSERTATION

Sebastian Fernando Torrealba

The Graduate School

University of Kentucky

2010

A Continuous Mathematical Model of the One-Dimensional Sedimentation Process of Flocculated Sediment Particles

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Agriculture
at the University of Kentucky

By
Sebastian Fernando Torrealba

Lexington, Kentucky

Director: Dr. Richard C. Warner, Professor of Biosystems and Agricultural Engineering

Lexington, Kentucky
2010

To my wife Laurentia.

To my parents.

ACKNOWLEDGMENTS

The I would like to thank Dr. Richard Warner for being such an inspiring person in my life. Since I first lay foot into this wonderful country he has been there for me, helping me, mentoring me, guiding me, and most of all he’s been there for me as a friend. I would like to thank him also for all the effort and time he spent with me working towards completing my dissertation. This dissertation wouldn’t have been possible without him.

I would also like to thank Carlos Jara for all the effort and time he spent reviewing the mathematical derivations that were provided always in a timely manner and for his priceless comments and advises. I greatly appreciate all the unconditional help he provided night and day.

All the lab work behind this dissertation wouldn’t have been possible without the help of Otto Hoffmann. I would like to thank him specially for improving the withdrawal system of the settling columns that considerable improved the quality and accuracy of the column tests data.

Additionally I would like to thank Dr. Dwayne Edwards, Dr. Stephen Workman and Dr. James Fox for their continuous guidance and assistance.

Finally, I would like to thank my wife Laurentia for being so patient during this long process. She always believed in me and pushed me to levels I would have never reached otherwise.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: Introduction

Construction, commercial forestry, agriculture and surface mining activities have the potential to cause surface water pollution if not sufficiently controlled. Traditionally erosion and sediment transport has been addressed through application of Best Management Practices (BMPs). A sediment pond is a frequently used BMP designed to retain sediment. The performance of a sediment pond depends on various factors such as physical and chemical sediment characteristics, inflow hydrograph and sedimentgraph, and pond geometry. Sediment ponds in general are capable of retaining sediment particles larger than fine sands (0.02 mm in diameter) leaving the smaller particles such as silts and clays partially treated. Recently published environmental regulations replace simply providing BMPs at construction sites with a defined turbidity limit [1.1]. Sediment ponds are expected to require flocculation to meet new effluent regulations. Modeling of the performance of sediment ponds needs to be improved to accommodate flocculated sediments. Sedimentation modeling based on physically-based processes should extend predictive capabilities.

A primary factor controlling the performance of a sediment pond is the particle size distribution (PSD) of the incoming sediment. Existing techniques that are used to enhance the performance of a sediment pond vary from inserting baffles and turbidity curtains that increase the residence time and reduce the dead space to the recent introduction of flocculation. The least common yet potentially the most efficient method to remove fine sediments is flocculation. Flocculation is the process where larger particles or flocs are formed from binding smaller particles. The binding mechanism can be of natural origin or artificially induced through the use of a chemical flocculant. Even though chemically induced flocculation is a common practice in the water treatment industry to remove water borne sediments it has seen very limited applications to sediment ponds. Induced flocculation is still considered a novel approach to enhancing sediment pond performance. Through flocculation the sediment particle size distribution (PSD) is shifted to the coarser side thus increasing particle settling velocities. No known successful flocculation model has been developed that can be used to design and predict the performance of a sediment pond employing induced flocculation.

Few cases can be found in the literature where sediment pond flocculation has been modeled ([1.2] [1.3] [1.4]). These approaches however use a discrete binary system of differential equations derived from the well-known Smoluchowski coagulation equation that requires a numerical method to approximate the solution. Obtaining an approximate solution to these type of models requires extensive computational time since each equation in the system tracks the changes in the initial PSD. To increase predictive accuracy the number of particle sizes modeled needs to be increased which increases the required computational time. There is a yet to be determined the tradeoff among modeling accuracy, number of modeled particle and computations time. Furthermore, each approximate solution of the flocculation model needs to be updated at each incremental time and space step of the numerical solution of the sediment pond model.

In an effort to develop the foundation of a comprehensive sediment pond model that includes flocculation, first a new flocculation model that requires less computational effort and yet provides the same or better degree of accuracy is needed. Secondly, a physically-based sediment pond model also needs to be developed that is computationally efficient while supplying accurate results, especially when combining with the flocculation model. Developing the framework of a new sedimentation/flocculation model was the primary objective of this research. Three interrelated research areas, addressed in this research are:

1. Mathematical representation of the particle size distribution (PSD) of sediment particles,
2. Flocculation model, and
3. Sedimentation model.

The dissertation is divided into the following chapters:

Chapter 2 provides a brief introduction to the mathematical representation of the PSD of sediment. This chapter also compares the performance of three different unimodal models to fit glass micro-spheres' PSDs. The Gamma probability distribution was introduced as a unimodal and bimodal model to represent the PSD of sediments providing excellent results.

Chapter 3 presents the Smoluchowski coagulation equation and the extensions to fragmentation and its continuous moments' representation. The moments' representation of the continuous aggregation and fragmentation equation was used in conjunction with the Gamma multimodal distribution to derive a new flocculation model. The primary advantage of this new model is that the mathematical problem is reduced to a scalar continuous form that is readily solved in a computationally efficient manner.

In Chapter 4 the flocculation model presented in Chapter 3 was calibrated using flocculated PSD obtained from four different sediment samples with anionic polyacrylamides flocculants from four different manufacturers.

Chapter 5 first provides an introduction of the Kynch sedimentation theory that is used to derive the sedimentation model of a polydispersed system of sediment particles. The resulting model is a system of conservation laws, where each equation representing a different particle size specie. Using this approach a new one-dimensional sedimentation model was developed using the continuous representation of the PSD instead of the discrete form. The result is a scalar conservation law that is more cost-effective to solve than the system of equations alternative.

Finally, Chapter 6 lists possible future research ideas that can extend the presented research.

CHAPTER 2: Multimodal Model Representation of the Sediment Particle Size Distribution Using the Gamma Probability Distribution

Abstract

Particle size distribution information is needed to model the sedimentation process. The accuracy of sedimentation models that consider particle-to-particle interaction is enhanced with a continuous representation of the PSD. Three different parametric models use to mathematically represent PSD data: Jaky, Fredlund and Gamma distributions were chosen to fit three different glass micro-spheres PSDs having average particle sizes of 7, 25 and 35 μm. An X-Ray particle size analyzer, that employs the sedimentation theory, provided a total of 117 data points for each PSD. The square root of the average sum of square errors was used to evaluate the goodness of fit of the three tested models. The Gamma distribution provided the best fit and had the least parameters. A multimodal form of the Gamma distribution was used which followed Durner's approach and Fredlund's bimodal formula. The Gamma multimodal form was validated by fitting it to bimodal and trimodal PSD data created from combining two and three different glass micro-spheres samples for various mass proportions. The projected mass proportions were obtained using multi-linear regression analysis that yielded coefficient of determination (R^2) over 0.993 and 0.9667 for the bimodal and trimodal tests;

respectively. The relative average errors between the observed and projected mass proportions were within 7% to 8% in average, which is normal considering the systematic errors embedded in sample preparation, equipment precision and error propagation. Ten soil samples were used to determine the efficacy of the unimodal and bimodal forms of the GPD. Model performance, based on the average error or deviation, was below 14.72% and 4.84% for the unimodal and bimodal forms, respectively.

2.1 Introduction

The Particle Size Distribution (PSD) can be used to predict numerous physical properties of a soil. Through mathematical relationships the PSD can be used to accurately represent the hydraulic conductivity of a soil. PSD data is also used for soil classification purposes, the most simple being the USDA textural triangle where only the silt, sand and clay fractions are needed. The USCS requires a more complete sieve analysis in addition to Atterberg limits.

The PSD of a sediment sample is a relevant input in the design of a sediment detention pond where the fine fraction (silt and finer fraction) of the PSD determines the performance or sediment trapping efficiency. Different methods exist to obtain the PSD of a sediment sample (finer fraction generally passing sieve #200, 0.075 mm in size opening), which are based on sedimentation techniques using Stokes' law and optical methods (laser diffraction). Sedimentation techniques provide the distribution of Stokes' diameters whereas optical methods provide the distribution of geometric sizes. The most common sedimentation procedure is the hydrometer method, where the size distribution of silts and clays is determined through variations of the density of the sediment and water solution over time. The hydrometer test is a simple and inexpensive test; however it requires a great amount of time to measure the mass fraction of small clay particles (diameter smaller than 0.002 mm) and the resolution of the resulting curve is low (small amount of points are obtained based on the precision of the equipment). New and more advanced lab equipment are available that obtain the PSD by measuring the radiation absorption of particles suspended in a solution. This type of equipment provides a much higher resolution and accuracy of the finer fraction than the more traditional tests, and in a fraction of the time.

Numerous benefits can be found from mathematically representing the PSD of a soil. The mathematical expression describing the PSD can provide an easier and more effective way of classifying soils. Other benefits include having more rigorous tools for the statistical analysis of the PSD. The hydraulic properties of a partially saturated soil, the pore size distribution (the complement of the PSD), can be mathematically analyzed to determine the flow properties of water through the void structure of the soil. Many authors have provided different models relating PSD to hydraulic conductivity of partially saturated soils, and a limited list can be found in Fredlund et al. (2000) [2.7].

Durner (1994) [2.6] represented the hydraulic conductivity function by superimposing unimodal retention curves which are equivalent to the pore size distribution curves. Durner provided the concept behind the multimodal model that will be discussed later in this paper. In sedimentation theory, mathematically representing the PSD of sediments can lead to the development of particle-to-particle interaction models such as based on an eroded PSD model or a flocculated PSD model.

2.2 Background

There are several possible approaches one can use to mathematically represent the PSD of soil particles. The simplest method is to fit a certain parametric function to the PSD data (e.g. a probability distribution). Given the analogy between the PSD and a theoretical probability distribution (e.g. a normal distribution), the PSD can be represented using a cumulative probability distribution (CPD) *F*(*x*) or using a probability density function (PDF) *f*(*x*), where:

$$F(x) = \int_{-\infty}^{x} f(\tau)d\tau \qquad (2.1)$$

In soil mechanics it is common to use the cumulative representation plotted in a semi-log scale (Figure 2 – 1).

Soils can have a uniform, well graded or gap graded PSD. The PSD of a uniform and well graded soil has one mode (unimodal) determined by the inflexion point in the CPD or global maxima in the PDF whereas a gap graded soil has a PSD with two (bimodal) or more modes (multimodal). The difference between a well graded and a uniform soil is that the latter possesses PSD data with smaller variance. A multimodal PSD is centered around multiple size classes and thus revealing multiple inflexion points (Figure 2 - 1).

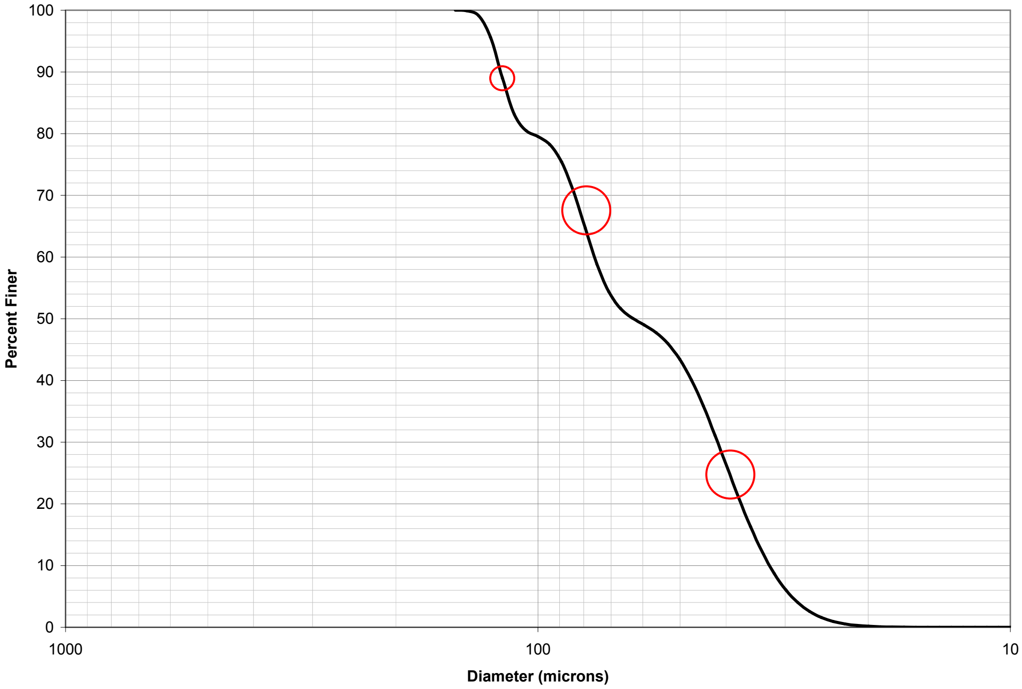

Figure 2 - 1: Example of a multi-modal PSD having three inflexion points (marked with red circles).

2.2.1 Unimodal PSD Models

Various parametrical unimodal models have been proposed in the literature, most of which were reviewed and summarized in [2.7], [2.8], and [2.13]. One of the initial approaches to mathematically represent the PSD without using a known probability distribution was conducted by Jaky [2.9], where a single parameter exponential function was proposed:

$$F_J(x) = \exp\left\{-\frac{1}{k^2}\left[\ln\left(\frac{x}{x_{\max}}\right)\right]^2\right\}, \tag{2.2}$$

where: k is a parameter, x is the particle diameter and $x_{\max}$ is the maximum diameter. Jaky's model provides good results when fitting a PSD with few data points [2.9]. For PSD data with higher degree of resolution it is necessary to increase the number of fitting parameters. Buchan [2.4] studied the validity of the simple two-parameter lognormal distribution (LND) in fitting soil PSD data and concluded that the LND would yield good approximations on about half of the USDA soil textural classes, namely for soils with low clay content. The lack of compatibility with the higher clay content soils is due to the fact that the LND is symmetrical and thus would not perform well with soils having PSD with *heavy tails* (skewed distribution). Thus, for sediments where the PSD tends to tail heavily towards the fine fraction, the use of the LND would not be recommended.

Another theoretical probability distribution function that has been used to fit PSD data is the Weibull distribution [2.15], in which the modified [2.1] expression is given by the following three-parameter formula:

$$F_{MW}(x) = c_1 + (1 - c_1)\left\{1 - \exp\left(-c_2 D(x)^{c_3}\right)\right\} \tag{2.3}$$

where $D(x) = (x - x_{\min})/(x_{\max} - x_{\min})$, c_i (i = 1,2,3) are parameters and $x_{\min}$ and $x_{\max}$ are the lower and upper bound of the particles sizes; respectively.

Fredlund, et. al. [2.7] developed an empirical formula that uses four parameters to fit unimodal PSD data. Fredlund's PSD formula is given by:

$$F_{Fr}(x) = \frac{1}{\left\{\ln\left[\exp(1) + \left(\frac{a_{gr}}{x}\right)^{n_{gr}}\right]\right\}^{m_{gr}}} \left\{1 - \left[\frac{\ln\left(1 + \frac{d_{rgr}}{x}\right)}{\ln\left(1 + \frac{d_{rgr}}{d_m}\right)}\right]^{\prime}\right. \tag{2.4}$$

where: F_{Fr} represents the cumulative PSD function given in percent finer by mass,

x the particle's diameter,

a_{gr} a parameter designating the inflection point on the curve,

n_{gr} a parameter related to the steepest slope on the curve,

m_{gr} a parameter related to the shape of the curve as it approaches the fines region,

d_{rgr} a parameter related to the amount of fines, and d_m the minimum allowable particle diameter.

Another approach is to model the PSD using fractal theory [2.12]. The rationale behind this approach lies in the process of fragmentation where soils and sediments result from the weathering of rock over time. A rock fragments into smaller pieces due to weathering and those smaller pieces fragment in even smaller pieces and the process continues until a soil is formed. Therefore it is expected that the PSD of a soil can be represented using fractal theory. In mathematical terms the cumulative number of particles N of sizes larger that x is given by a power law,

$$N(x) \sim x^{-\kappa}, \tag{2.5}$$

where κ is a parameter that is related to the fractal dimension *D*. A variation of (2.5) was developed by Kravchenko and Zank [2.11] to characterize the hydraulic conductivity of soils using a fractal model of the PSD,

$$F_{KZ}(x) = \eta \; x^{\frac{3\delta^2-13\delta+14}{\delta^2-5\delta+4}+1}, \tag{2.6}$$

where η and δ are fitting parameters.

2.2.2 Multimodal PSD Models

A PSD having a multimodal form or shape has multiple inflexion points or modes (Figure 2 – 1). Using a simple unimodal model may lead to significant errors when trying to fit a multimodal PSD curve.

One approach to fitting a multimodal PSD can be achieved by splitting the curve into segments, and using for example Spline polynomials as an approximation for each segment. In mathematical terms the PSD data is fitted using a function of the form:

$$F(x) = \begin{cases} F_1(x), & x_{\max} \geq x > x_1 \\ F_2(x), & x_1 \geq x > x_2 \\ \vdots & \\ F_n(x), & x_{n-1} \geq x \geq x_{\min} \end{cases} \tag{2.7}$$

where: *n* is the number of segments and x_i (i = 1...n-1) are the segment boundaries. The difficulty with fitting the PSD data by segments is to construct a continuous function by eliminating the discontinuities at the boundaries. The fitting process does not end after calibrating the parameters of each function F_i (i = 1…n) but when all the discontinuities are smoothen out, which translate into solving another set of equations involving the entire set of fitting parameters. This method can become quite difficult and time consuming.

The most logical approach to fit multimodal PSD data would then be to use a superposition of unimodal models since the sum of unimodal functions is multimodal. Let $F(x)$ be the finite sum of unimodal functions $F_i\ (x)$ (i = 1…n) and ω_i weighing functions with $\sum_{i=1}^{n} \omega_i = 1$, then

$$F(x) = \sum_{i=1}^{n} \omega_i F_i(x) \tag{2.8}$$

is multimodal.

Fredlund proposed a multimodal form of his unimodal formula (2.4) based on Durner's [2.6] modeling approach of a gap-graded soil as the linear combination of a finite number of his empirical unimodal model. In summary any gap-graded soil PSD can be fitted using the superposition of multiple weighed unimodal models.

2.3 Experimental Procedure

Soda-lime glass micro-spheres with a density of 2.50 g/cm^3 fabricated by Potters Industries were used in the experiment. Micro-spheres were chosen since they do not form aggregates thus particle-to-particle interactions are not existent. Moreover, the glass micro-spheres are industrially fabricated under high quality control levels resulting in a PSD will less standard deviation with respect to the mean diameter. Particles with mean diameter of 35, 25 and 7 microns were designated as Large, Medium and Small, respectively.

The individual (small, medium and large) PSDs of the glass micro-spheres were obtained by conducting three replicates using an X-Ray particle size analyzer (Appendix B). The particle size analyzer was a model Sedigraph 5100 built by Micromeritics. This particle size analyzer uses the sedimentation technique, which measures the gravity-induced settling rates of different size particles in a liquid of known properties. The SediGraph 5100 uses a narrow, horizontally collimated beam of X-rays to directly measure the relative mass concentration of particles in the liquid medium. This is accomplished by first measuring the intensity of a reference X-ray beam which is projected through the clear liquid medium prior to the introduction of the sample. A homogeneously dispersed mixture of a sediment sample and liquid is next circulated through the cell. The solid particles absorb some of the X-ray energy, which again is measured, this time to establish a value for full scale attenuation. Afterwards, the agitation of the mixture is stopped and the dispersion is allowed to settle while X-ray intensity is monitored. The SediGraph 5100 can measure particle diameters ranging from 300 to 0.10 microns. A minimum mass of 4 grams is needed for each sample to operate the particle size analyzer. The X-Ray particle size distribution provided a total of **117 data points** for particle sizes ranging from 150 microns through 0.18 micron.

2.4 Results

2.4.1 Unimodal PSD Model Selection

The individual glass micro-spheres' PSD can be modeled using a unimodal PSD due to the small standard deviation of the manufactured spheres. To model the three individual micro-spheres PSDs two different unimodal models, described in Section 2.2.1, were used, namely the Jacky and Fredlund models. The Jaky PSD model was selected since it only uses one parameter and thus is the simplest model considered. The Fredlund model was chosen based on its documented performance [2.8]. In addition to these models, the two parameter Gamma cumulative probability distribution (GPD) was also selected to fit the glass micro-spheres individual samples. The main reason for choosing the GPD as a PSD model is its non-symmetrical shape, which is expected to fit fine PSD data more accurately. It may prove to be insightful to compare these three unimodal models. The mathematical expression of the GPD is given by:

$$F_{\Gamma}(x) = \int_{0}^{x} \tau^{\alpha-1} \frac{\exp(-\tau/\beta)}{\beta^{\alpha}\Gamma(\alpha)} d\tau, \tag{2.9}$$

with $\alpha, \beta > 0;$ where α is the shape parameter, β is the scaling parameter and $\Gamma(\alpha)$ is the Gamma function.

The model parameters were estimated using the least squares regression method (LSRM). The results are shown in Figures 2 - 2 through 2 - 4 for the Jaky, Fredlund and GPD unimodal models, respectively.

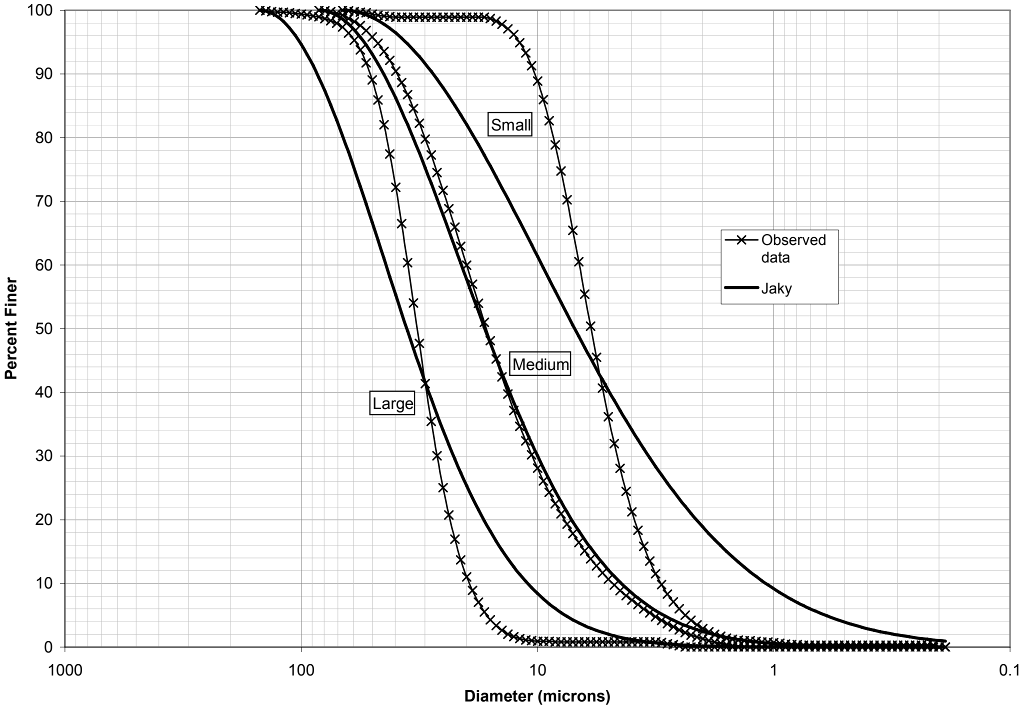

Figure 2 - 2: Jaky model fit to the glass micro-sphere PSD data

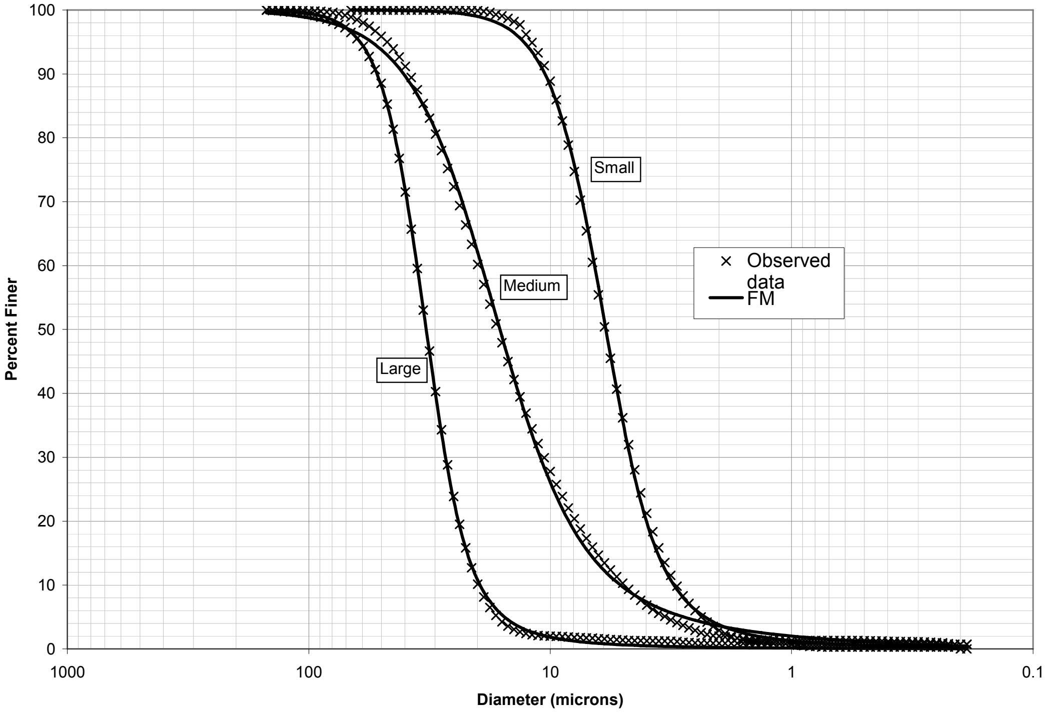

Figure 2 - 3: Fredlund model fit to glass micro-spheres PSD data

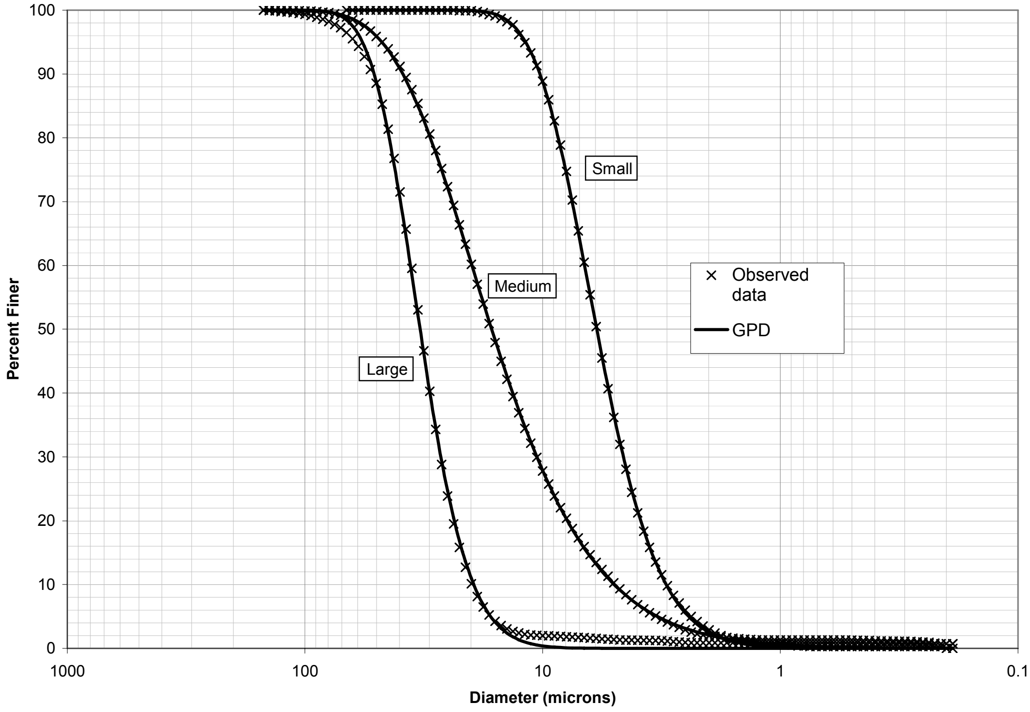

Figure 2 - 4: GPD model fit to glass micro-sphere PSD data

Since the coefficient of determination (R^2) is not applicable to non-linear regression (its interpretation may lead to significant errors), the performance of each model was evaluated by comparing the square root of the average sum of least squares of errors:

$$\bar{\varepsilon} = \sqrt{\min_{\theta} \overline{SSE}} = \left[\min_{\theta} \left\{ \frac{1}{N} \sum_{i} \left(y_i - f\left(x_i, \theta\right) \right)^2 \right\} \right] \tag{2.10}$$

where: $\overline{SSE}$ is the mean sum of least squares of errors, N is the number of data points (117 for each PSD), y_i are the observed values, $f\left(x_i, \theta\right)$ are the projected values and $\theta = \left(\theta_1, \theta_2, \ldots, \theta_p\right)^T$ are the model parameters. The parameters were estimated using the SOLVER search tool found in Excel and the resulting $\bar{\varepsilon}$ are presented in Table 2 - 1.

Table 2 - 1: Values of $\bar{\varepsilon}$ (2.10), which resulted from the non-linear regression method.

Sample	Jaky	FM	GPD
Small (7 µm)	13.195	0.777	0.285
Medium (25 µm)	1.838	1.209	0.547
Large (35 µm)	8.241	0.622	1.060
Average	7.758	0.869	0.631

The Gamma distribution and the Fredlund model provided similar overall performance when fitting the three initial glass micro-spheres PSDs. The GPD performed better for the finer mean diameter PSDs (small and medium) compared to the Fredlund model and since the performance of many sediment ponds is predominantly based on particle sizes in the range of 10 to 25 μm the GPD was selected for multimodal model development. Furthermore; the GPD model distribution has two important advantages when compared to the FM. Firstly, it only has two parameters compared to four of the FM and secondly, the GPD is a theoretical probability distribution and thus its corresponding statistics (e.g. mode, mean, variance, etc.) are known (Table 2 - 2).

Table 2 - 2: Simple forms of the Gamma distribution statistical parameters.

Statistical Parameter	Formula
Mean	$\alpha\beta$
Mode	$(\alpha-1)\beta$
Variance	$\alpha\beta^2$
Skewness	$\frac{2}{\sqrt{\alpha}}$

2.4.2 Multimodal PSD Model

Soils in nature are a combination of clays, silts, sands and gravels, which represent different particle size classes or ranges. Intuitively if the PSD of each size class could be obtained separately, the final or composite PSD would simply be the linear combination of the entire PSD set, (2.8). The weighing factors (ω_i) would then be the mass proportions of the different size classes. This is true however only if no particle-to-particle interaction occurs during the mixing process (no aggregation or breakup). The proof by induction over the number of size classes *n* is presented in Appendix A.

Procedure:

The small (7 μm), medium (25 μm) and large (35 μm) glass micro-spheres samples were combined to create composite PSDs by mixing two and three different individual glass micro-spheres' samples; respectively. By the previous definition these combined samples are predominantly bimodal and trimodal in nature. The mass proportions (ω_i) used in the composite PSDs (bimodal and trimodal) are defined and designated as targeted mass proportions. The targeted mass proportions for the composite samples created by mixing two (bimodal) and three (trimodal) tests are listed in Table 2 – 3 and Table 2 – 4, respectively.

Table 2 - 3: Targeted mass proportions (ω_i) for bimodal tests (combining two PSDs).

Sediment Type	Test Number														
	1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10	1-11	1-12	1-13	1-14	1-15
Small (7 µm)	0.50	-	0.50	0.20	-	0.20	0.80	-	0.80	0.35	-	0.35	0.65	-	0.65
Medium (25 µm)	0.50	0.50	-	0.80	0.20	-	0.20	0.80	-	0.65	0.35	-	0.35	0.65	-
Large (35 µm)	-	0.50	0.50	-	0.80	0.80	-	0.20	0.20	-	0.65	0.65	-	0.35	0.35

Table 2 - 4: Targeted mass proportions (ω_i) for trimodal tests (combining three PSDs)

Sediment Type	Test Number												
	2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8	2-9	2-10	2-11	2-12	2-13
Small (7 μm)	0.33	0.15	0.15	0.30	0.30	0.55	0.55	0.10	0.45	0.45	0.20	0.20	0.60
Medium (25 μm)	0.33	0.30	0.55	0.15	0.55	0.15	0.30	0.45	0.10	0.45	0.20	0.60	0.20
Large (35 μm)	0.33	0.55	0.30	0.55	0.15	0.30	0.15	0.45	0.45	0.10	0.60	0.20	0.20

The observed mass proportions are obtained from the weighting process of the individual samples. The observed mass proportions are expected to be very similar or equal to the targeted mass proportions. Finally, the projected mass proportions (ω_i^p) are obtained using a multi-linear regression method:

$$y(x_i) = \omega_1^p F_{\Gamma,1}(x_i) + \omega_2^p F_{\Gamma,2}(x_i) + \ldots + \omega_n^p F_{\Gamma,n}(x_i) + \varepsilon_i, \quad i = 1\ldots \tag{2.11}$$

where: $y(x_i)$ is the observed percent finer than size x_i, N is the number of PSD data points, $F_{j,\Gamma}$ is the j^{th} GPD given by (2.9), with $j = 1\ldots n$, and ε_i is the error or residual associated to particle size x_i. In matrix notation:

$$\mathbf{y} = \mathbf{F}_\Gamma \mathbf{w} + \boldsymbol{\varepsilon}, \tag{2.12}$$

where:

$$\mathbf{F}_{\Gamma\,(N\times n)} = \begin{pmatrix} F_{\Gamma,1}(x_1) & F_{\Gamma,2}(x_1) & \cdots & F_{\Gamma,n}(x_1) \\ F_{\Gamma,1}(x_2) & F_{\Gamma,2}(x_2) & \cdots & F_{\Gamma,n}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ F_{\Gamma,1}(x_N) & F_{\Gamma,2}(x_N) & \cdots & F_{\Gamma,n}(x_N) \end{pmatrix},$$

$$\mathbf{w}_{(n\times 1)} = \left(\omega_1^p, \omega_2^p, \ldots \omega_n^p\right)^T,$$

$\mathbf{y}_{(N\times 1)} = \left(y(x_1), y(x_2), \ldots, y(x_N)\right)^T$, and

$$\varepsilon_{(N\times 1)} = \left(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N\right)^T.$$

The solution to the multi-linear regression is given by solving the normal equations:

$$\left(\mathbf{F}_\Gamma{}^T \mathbf{F}_\Gamma\right)\mathbf{w} = \mathbf{F}_\Gamma{}^T \mathbf{y}. \tag{2.13}$$

Results:

The linear regression problem (2.13) is solved using the Excel Solver tool and the results (projected mass proportions) are presented in Tables 2 - 5, 2 - 6 and 2 - 7 for the bimodal tests and in Tables 2 - 8, 2 - 9 and 2 - 10 for the trimodal tests. The goodness of fitting is evaluated through the coefficient of determination (R^2).

Table 2 - 5: Mass proportions for repetition #1 of the bimodal test group

Test	**1**	**2**	**3**	**4**	**5**
w_1	0.502		0.546	0.190	
w_2	0.498	0.505		0.810	0.210
w_3		0.495	0.454		0.790
R^2	0.999942	0.999869	0.999931	0.999819	0.999728
Test	**6**	**7**	**8**	**9**	**10**
w_1	0.235	0.828		0.832	0.363
w_2		0.172	0.835		0.637
w_3	0.765		0.165	0.168	
R^2	0.999961	0.999780	0.999889	0.999906	0.999824
Test	**11**	**12**	**13**	**14**	**15**
w_1		0.404	0.675		0.691
w_2	0.376		0.325	0.693	
w_3	0.624	0.596		0.307	0.309
R^2	0.999829	0.999946	0.999828	0.999789	0.999867

Table 2 - 6: Mass proportions for repetition #2 of the bimodal test group

Test	**1**	**2**	**3**	**4**	**5**
w_1	0.509		0.537	0.187	
w_2	0.491	0.477		0.813	0.164
w_3		0.523	0.463		0.836
R^2	0.999687	0.999775	0.999670	0.999914	0.999566
Test	**6**	**7**	**8**	**9**	**10**
w_1	0.207	0.860		0.802	0.380
w_2		0.140	0.807		0.620
w_3	0.793		0.193	0.198	
R^2	0.999527	0.999772	0.999832	0.999328	0.999762
Test	**11**	**12**	**13**	**14**	**15**
w_1		0.396	0.636		0.673
w_2	0.304		0.364	0.631	
w_3	0.696	0.604		0.369	0.327
R^2	0.999704	0.999647	0.999834	0.999942	0.999461

Table 2 - 7: Mass proportions for repetition #3 of the bimodal test group

Test	**1**	**2**	**3**	**4**	**5**
w_1	0.512		0.570	0.203	
w_2	0.488	0.503		0.797	0.189
w_3		0.497	0.430		0.811
R^2	0.999932	0.999791	0.999846	0.999768	0.999514
Test	**6**	**7**	**8**	**9**	**10**
w_1	0.249	0.881		0.820	0.420
w_2		0.119	0.805		0.580
w_3	0.751		0.195	0.180	
R^2	0.999885	0.999645	0.999649	0.999936	0.999893
Test	**11**	**12**	**13**	**14**	**15**
w_1		0.420	0.670		0.726
w_2	0.350		0.330	0.651	
w_3	0.650	0.580		0.349	0.274
R^2	0.999809	0.999601	0.999948	0.999572	0.999879

Table 2 - 8: Mass proportions for repetition #1 of the trimodal test group

Test	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
w_1	0.307	0.117	0.184	0.289	0.338	0.509	0.539	0.071	0.487	0.486	0.206	0.178	0.574
w_2	0.339	0.300	0.550	0.174	0.535	0.154	0.294	0.456	0.062	0.411	0.162	0.601	0.198
w_3	0.354	0.583	0.266	0.537	0.128	0.337	0.168	0.473	0.452	0.103	0.631	0.221	0.228
R^2	0.997315	0.991525	0.998616	0.995439	0.996603	0.996324	0.993583	0.989642	0.998901	0.991934	0.988067	0.991680	0.997625

Table 2 - 9: Mass proportions for repetition #2 of the trimodal test group

Test	**1**	**2**	**3**	**4**	**5**
w_1	0.378	0.196	0.145	0.327	0.309
w_2	0.325	0.279	0.532	0.156	0.558
w_3	0.297	0.526	0.323	0.517	0.134
R^2	0.990997	0.996021	0.997197	0.989164	0.999217
Test	**6**	**7**	**8**	**9**	**10**
w_1	0.544	0.546	0.076	0.497	0.469
w_2	0.162	0.281	0.437	0.051	0.449
w_3	0.294	0.173	0.487	0.452	0.081
R^2	0.977761	0.991406	0.986632	0.996643	0.993420
Test	**11**	**12**	**13**		
w_1	0.210	0.227	0.628		
w_2	0.231	0.579	0.174		
w_3	0.559	0.194	0.198		
R^2	0.991132	0.998383	0.992098		

Table 2 - 10: Mass proportions for repetition #3 of the trimodal test group

Test	**1**	**2**	**3**	**4**	**5**
w_1	0.301	0.133	0.162	0.327	0.323
w_2	0.346	0.300	0.527	0.133	0.536
w_3	0.353	0.567	0.311	0.540	0.141
R^2	0.999103	0.994023	0.997845	0.998721	0.996580
Test	**6**	**7**	**8**	**9**	**10**
w_1	0.501	0.606	0.144	0.422	0.495
w_2	0.150	0.261	0.450	0.118	0.403
w_3	0.349	0.133	0.406	0.460	0.102
R^2	0.966756	0.993482	0.985729	0.997475	0.994498
Test	**11**	**12**	**13**		
w_1	0.207	0.218	0.600		
w_2	0.225	0.628	0.173		
w_3	0.568	0.153	0.227		
R^2	0.989990	0.991789	0.990690		

The multimodal GPD provided an excellent fit to observed bimodal and trimodal PSDs with R^2 exceeding 0.977 and 0.967 for the bimodal and trimodal tests, respectively. The observed and projected PSD graphs are illustrated in Appendix C and D for the bimodal and trimodal tests; respectively.

2.5 Fitting the GPD to Soil PSD Data

In the previous section the hypothesis that the PSD of a combined set of individual PSDs is the linear combination of the individual PSDs was tested and accepted for a combination of glass micro-sphere PSDs. Therefore, the statement described by (2.8) was validated for glass micro-spheres.

To determine if the GPD model is also applicable to sediment PSDs the GPD model was tested on ten soil PSDs. Soils' PSDs are multimodal by nature and thus the multimodal form of the GPD is recommended to fit the PSD of sediments; however this depends on the shape of the observed PSD and on the degree of targeted accuracy. To illustrate this concept, ten different sediment PSDs have been selected to use the GPD to fit the PSD data. The goodness of fit was evaluated using the average error, given by:

$$\bar{\varepsilon} = \frac{1}{N}\sum_{i}\left|F_{pred,i} - F_{obs,i}\right|, \tag{2.14}$$

where: N is the number of data points,

F_{pred} is the predicted PSD value obtained using the GPD model, and

F_{obs} is the observed PSD value obtained using the X-Ray particle size distribution.

Two cases were analyzed and compared: 1) using the unimodal form of the GPD and 2) using the bimodal form of the GPD. The curve fitting was conducted using the least sum of squares regression method with the help of the Excel Solver tool. The resulting average errors are presented in Table 2 – 11 for both unimodal and bimodal GPD models for the ten different sediment PSDs. The observed and predicted PSD curves are illustrated in Appendix E.

Table 2 - 11: Average error results after fitting the unimodal and bimodal GPD model on 10 different sediment samples.

Sediment	Average Error (%)		Percent
Sample	Unimodal	Bimodal	Reduction[(a)]
S1	14.72	1.61	89.06
S2	13.35	1.98	85.17
S3	10.66	4.84	54.60
S4	12.17	0.62	94.91
S5	0.60	0.18	70.00
S6	0.71	0.61	14.08
S7	3.97	1.13	71.54
S8	5.08	1.16	77.17
S9	5.46	0.25	95.42
S10	3.40	0.23	93.24

Notes: [(a)] Percent reduction of the average error from the unimodal to the bimodal form

As it can be inferred from the above results the improvements of fitting sediment PSD using a bimodal GPD instead of the unimodal form are substantial. The average reduction in the average error is approximately 75% for the 10 sediment PSDs tested. It is therefore recommended to use a bimodal GPD to fit sediment PSD.

2.6 Summary and Conclusions

1. Three industrially manufactured glass micro-sphere samples having mean particle sizes of 7, 25 and 35 μm were selected to fit the unimodal Jaky, Fredlund and Gamma models. Results revealed that the Fredlund model and the Gamma distribution fitted the 117 PSD data points accurately (average square root of sum of square errors of 0.631 and 0.869; respectively).
2. The Gamma distribution was selected for multimodal PSD testing since it performed slightly better than the Fredlund model and because it required only 2 parameters versus the four parameters required for the Fredlund model.
3. A multimodal form was presented as a linear combination of Gamma distributions following Durner's approach and extending the principle behind Fredlund's bimodal formula.
4. The validation of the multimodal hypothesis was conducted using bimodal and trimodal PSD data created from individual glass micro-spheres' PSD. The multimodal Gamma model was fitted to bimodal and trimodal glass micro-spheres PSD created by combining two and three different glass micro-spheres PSD samples; respectively, at varying mass proportions.
5. Results showed excellent correlation between the multimodal form (2.8) of the Gamma distribution and the observed PSD data with a coefficient of determination (R^2) of greater than 0.9993 and 0.9667 for the bimodal and trimodal test PSDs; respectively.

6. It was proven theoretically and empirically that a combination of two or more PSDs can be modeled by the linear combination of the individual PSDs.
7. The multimodal Gamma model was extended to fit sediment PSDs. Ten different sediment samples were selected to fit the unimodal and bimodal Gamma distribution models to the data. Results showed that the bimodal Gamma distribution model is an excellent (average error < 5%) model to fit sediment PSD data and that the bimodal representation resulted in a significant lower average error than the unimodal modeling approach.

CHAPTER 3: Modeling the Flocculation of Fine Sediments using the Method of Moments to Solve the Coagulation Equation – Theory

Abstract

The method of moments for solving the coagulation equation of fine sediment particles was investigated when the initial particle size distribution (PSD) was modeled using a Gamma probability distribution. General mathematical representations of the aggregation and fragmentation kernels when turbulent shear forces induce the particle aggregation and fragmentation were addressed in this analysis. A general moments' equation was derived that considers aggregation and fragmentation, which has the form of an integro-differential equation. The general moments' equation was applied to four different cases: 1) coagulation with initial unimodal Gamma PSD function, 2) coagulation with initial multimodal Gamma PSD function, 3) coagulation and fragmentation with initial unimodal Gamma PSD function and 4) coagulation and fragmentation with initial multimodal Gamma PSD function. For the aggregation cases only an analytical solution to the moments' change rate over time dM_k / dt was found. In the case of pure fragmentation the moments' change rate was simplified to an indefinite integral that can be easily evaluated numerically. Finally, expressions of the time derivative of the PSD Gamma parameters are presented for each case.

3.1 Introduction

The study of aggregation of fine sediment particles is of importance in surface water treatment. Fine sediment particles such as silts and clays are difficult to retain due to the large amount of time required to settle. Sediment from agricultural, construction and mining activities is considered one of the most important sources of contamination of surface water bodies; therefore, controlling the amount of sediment that reaches the natural water corridors is critical.

Once sediment particles are transported through runoff, only the coarser particles such as sands and coarse to medium silt can be efficiently trapped inside sediment control structures (e.g. sediment pond). The settling velocity of sediment particles can be considerably increased through aggregation or flocculation where particles bind together after contact to form a larger particle. The efficiency of a sediment control structure can be greatly improved through flocculation. An advanced sedimentation model should consider the changes in particle size distribution when flocculation occurs.

The theory of changes in particle size distribution due to aggregation is described and advanced employing the method of moments when a Gamma probability distribution is used to model the initial particle size distribution. Firstly, a brief background of the Smoluchowski's coagulation equation for binary aggregation is described. Furthermore, both aggregation and fragmentation processes are discussed and general expressions for their modeling are obtained through the use of the method of moments. Formulas are derived where the PSD parameters are expressed as ordinary differential equation with respect to time when a Gamma distribution is used to model the initial particle size distribution.

3.2 Background

3.2.1 Coagulation Equation

The Smoluchowski's coagulation equation (SCE), also referred as the *population balance equation*, models the coalescence between particles over time [3.19]. This equation finds its use in a large spectrum of applications: chemistry (polymerization), astrophysics (creation of planets), statistical physics (particle aggregation), environmental engineering (water-solid separation), aerosol science and mining (thickening), to name a few. The primary focus of this chapter is developing a flocculation model for fine sediment particles. The term aggregation of sediment particles is equivalent to the flocculation term and an aggregate represents a composite particle made from a finite set of smaller particles bounded together. Both terms will be used interchangeably within the literature.

Consider a set of particles of the same mass and same volume suspended in a liquid medium. This type of suspension is a monodisperse system since particles are of the same size and have the same physical properties. Let x_i (i = 1,2,…) be the mass of the aggregate formed by *i* particles then $x_i = ix_1$, where x_1 is the mass of 1 particle (i.e. a mono-aggregate). The *i*-th *particle number concentration* is given by the number of aggregates having mass x_i per unit mass and will be denoted by $n(x_i, t) \equiv n_i(t)$. An aggregate having mass x_i coming in contact with another aggregate of mass x_j will form a new aggregate of mass x_{i+j} at a rate given by the aggregation Kernel $K(x_i, x_j) \equiv K_{i,j}$. Obviously, this process rends the particle size distribution coarser in time. The

Smoluchowski theory of rapid coagulation results in the following initial value problem described by an infinite set of nonlinear differential equations:

$$\frac{dn_i(t)}{dt} = \frac{1}{2}\sum_{j=1}^{i-1} K_{j,i-1} n_j(t) n_{i-j}(t) - n_i \sum_{j=1}^{\infty} K_{i,j} n_j(t), \quad i = 1,2,\ldots \qquad (3.1)$$
$$s.t. \quad n_1(0) = n^0.$$

where: n^0 is the initial particle number concentration of x_1.

Monodisperse systems are rare in nature. Sediment particles for example come in a range of different sizes and shapes and therefore represent a polydisperse system of particles. The SCE is not valid to model rapid coagulation of a polydisperse system; however, an extended version of (3.1) that includes the case of polydisperse particles (i.e. particles of different sizes) having a continuous particle mass distribution was described in [3.8]. The number concentration per unit volume for a certain time where the mass varies between x_i and $x_i + \Delta x_i$ is given by $n_i = n(x_i, t)\Delta x_i$. The term $n(x_i, t)$ is the continuous particle number concentration. Replacing the continuous approximation of n_i in (1) (for *i*, *j* and *i* – *j*) then dividing by Δx_i and taking the limit of the Riemann sums over all *i*'s yields the continuous representation of the Smoluchowski's coagulations equation:

$$\frac{\partial n(x,t)}{\partial t} = \frac{1}{2}\int_0^x K(y, x-y) n(y,t) n(x-y,t) dy - n(x,t)\int_0^{\infty} K(x,y) n(y,t)c \qquad (3.2)$$
$$s.t. \quad n(x,0) = n_0(x).$$

where: x and y denote the continuous variables representing the mass of the particles and $n_0(x)$ is a continuous function that represent the initial particle number distribution. The first integral on the right-hand side of (3.2) represents the increase in the particle number concentration of particles of mass *x* when a particle of mass $x - y$ (with $y < x$) comes in contact with a particle of mass *y*. The constant ½ accounts for the symmetry of the aggregation kernel (i.e. $K(x, y) = K(y, x)$) and the second integral represents the decrease in particle number concentration of particles of mass *x* when they aggregate with other particles. Since the scope of this chapter is to model changes in PSD of sediment particles due to aggregation, only the continuous representation of the SCE will be used.

The stability of the structure of a given aggregate can be impaired when the shear forces acting on said aggregate becomes larger than the resistance provided by the bond strength between the particles. When this happens fragmentation starts and a reorganization of the particle number concentration occurs. In other words if an aggregate fragments then the number concentration of each one of the resulting fragments will increase. The SCE only considers aggregation between particles without taking fragmentation of aggregates into account. Melzak [3.15] extended the continuous coagulation equation (3.2) by including particle fragmentation for the case of droplets, thus obtaining the scalar transport equation, given by:

$$\begin{aligned}\frac{\partial n(x,t)}{\partial t} &= \frac{1}{2}\int_0^x K(y,x-y)n(y,t)n(x,t)dy - n(x,t)\int_0^\infty K(x,y)n(y,t)dy \\ &\int_x^\infty n(y,t)\,\phi(y,x)dy - \frac{n(x,t)}{x}\int_0^x y\,\phi(y,x)dy,\end{aligned} \tag{3.3}$$

where: $\phi(y,x)$ is the fragmentation kernel that accounts for the rate of particle fragmentation occurring in a unit volume. The fragmentation kernel is also symmetric, i.e. $\phi(x,y) = \phi(y,x)$. The third integral on the right-hand side of (3.3) represents the increase of the number concentration of particles of mass *x* due to fragmentation of particles of mass *y* > *x* and the fourth integral accounts for the decrease of particles of mass *x* fragmenting into particles of smaller mass.

Using a mathematical representation of the fragmentation kernel that considers laminar and/or turbulent shear forces, equation (3.3) can be used, for example, to model the aggregation and fragmentation of sediment particles within a sediment pond.

Another commonly used form of the coagulation-fragmentation equation (e.g. [3.3], [3.4], [3.13] and [3.16]) is the linear kinetic fragmentation equation, which in its continuous form is given by:

$$\begin{aligned}\frac{\partial n(x,t)}{\partial t} &= \frac{1}{2}\int_0^x K(y,x-y)n(y,t)n(x,t)dy - n(x,t)\int_0^\infty K(x,y)n(y,t)dy \\ &\varphi(x)n(x,t) + \int_x^\infty n(y,t)\varphi(y)P(x,y)dy,\end{aligned} \tag{3.4}$$

where: $\varphi(x)$ is the overall rate of fragmentation, and

$P(x, y)$ is the particle number distribution of the fragments formed from an aggregate of mass y.

There is little known about the particle number distribution of the fragments [3.13]. Peterson [3.16] provides a list of the most common fragments' distributions found in the literature. Peterson also presents a simpler general mathematical representation of this distribution that has been used by other authors ([3.3] and [3.13]) to solve the fragmentation equation for specific cases. He expressed the fragments' distribution as $P(x, y) = p(y / x) / x$ where $p(y / x)$ is the probability that a particle of mass x will result from the fragmentation of a particle of mass y and is divided by x to obtain the number concentration of fragments.

The third term in (3.4) accounts for the loss of particles of mass x due to fragmentation and the fourth term represents the gain of particles of mass x due to fragmentation of particles having mass larger than x.

Both coagulation-fragmentation models described previously are similar in the sense that they model the evolution of the particle number concentration over time due to effects of aggregation and fragmentation. Either equation is therefore valid when an appropriate choice of the analytical representation of the fragmentation kernel(s) $\phi(x, y)$ for (3.3) and $\varphi(x)$ and $P(x, y)$ for (3.4), are employed.

3.2.2 Aggregation Kernel

The rate by which aggregates form depends on various factors such as particle size distribution, dynamic properties of the liquid medium (e.g. gravitational effects, laminar or turbulent flow), electrochemical forces acting between particles, pH, temperature, etc. These factors can be divided in two groups: factors arising from the probability of particles colliding and factors accounting for the probability that two particles will aggregate after colliding. The aggregation Kernel is the mathematical representation of the aggregation factors and identifies the coagulation process.

Several authors (see [3.1] for a detailed list) have provided analytical expressions of the aggregation kernel *K*. In particular, Smoluchowski proposed an expression for the aggregation kernel in monodisperse systems where coagulation is controlled by Brownian motion:

$$K(x,y) = c_1\left(x^{1/3} + y^{1/3}\right)\left(x^{-1/3} + y^{-1/3}\right), \tag{3.5}$$

where: c_1 is a coefficient that depends on the medium temperature and viscosity and thus is a function of the Brownian diffusion coefficient~~.~~

In natural systems, sediment aggregation occurs due to a combination of Brownian motion, and laminar and turbulent flow regimes. A sediment pond is a structure designed to trap sediment particles transported from up-gradient areas through water flow. In this type of system three zones where particle-to-particle aggregation occurs are identified.

Starting from up-gradient areas, the first zone includes all the overland flow areas where laminar flow is dominant, except for concentrated flow in rills. The second zone consists of streams and/or channels (concentrated flow) where turbulent flow occurs due to high velocities. The third zone is located inside the sediment pond where Brownian motion and laminar flow occur enabling sediment particles to settle. The highest rate of aggregation occurs in the second zone since higher mixing energy increase the probability of particle-to-particle collisions. It is thus expected that laminar kernels should have smaller values than turbulent kernels for the same (x, y) pair.

Saffman and Turner [3.17] derived a formula for the aggregation kernel that accounts for the aggregation rate under homogeneous turbulent shear:

$$K(x, y) = c_2 \left(x^{1/3} + y^{1/3} \right)^3, \tag{3.6}$$

where: c_2 is a coefficient that depends on the turbulent energy dissipation and on the medium's viscosity. Other authors (e.g. [3.13], [3.18]) used the following aggregation Kernel due to turbulent shear forces:

$$K(x, y) = c_3 \left(x^{1/D} + y^{1/D} \right)^3, \tag{3.7}$$

where: c_3 is a coefficient that depends on the physical properties of the problem, and

D is the fractal dimension of the aggregate, i.e. that the size distribution of the aggregate follows a power law ($f(r) \propto r^D$, where D is the fractal dimension).

After analyzing the turbulent aggregation kernels given in (3.6) and (3.7) it is logical to conclude that a general mathematical representation of the aggregation kernel accounting for turbulent shear forces can be expressed by:

$$K(x,y) = A\left(x^r + y^r\right)^p, \quad A, r \in \mathbf{R}^+,\ p \in \mathbf{N} \tag{3.8}$$

where: $\mathbf{R}^+$ is the set of positive real numbers,

$\mathbf{N}$ is the set of positive integers, and

A, r, and p are coefficients that depend on the physical properties of the flow regime and on the physical and chemical properties of the particles.

3.2.3 Fragmentation Kernel

Fragmentation of sediment particle aggregates arises mainly due to viscous shear forces. The probability of a given aggregate breaking up into smaller particles increases with the size of the aggregate and decreases with the strength of the bond. The most commonly used fragmentation kernel is a power law relationship (e.g. [3.13], [3.18] and [3.4]):

$$\varphi(x) = Bx^{\lambda} \text{ or } \phi(x,y) = B'(x^{\nu} + y^{\nu}), \tag{3.9}$$

where: *B*, *B'*, λ and ν are coefficients that depend on the physical properties of the problem.

3.2.4 Initial PSD Function

The coagulation equation (with or without fragmentation) consists of initial value problems that require identifying an initial particle number concentration distribution or particle size distribution (PSD) data. In Chapter 2 a new method of mathematically representing the initial PSD data was presented. Results showed that a multimodal representation using the Gamma distribution was an excellent option to mathematically model sediment PSD. If the 'probability' mass density function is used instead of the cumulative distribution, then the multimodal PSD function can be written as:

$$f(x) = \sum_{i=1}^{n} \omega_i f_i(x), \tag{3.10}$$

where: *n* represent the number of modes of the PSD function

ω_i represents the mass proportion for each mode with $\sum_{i=1}^{n} \omega_i = 1$,

f_i is the associated PSD function.

The Gamma probability density function is given by:

$$f_\Gamma(x) = x^{\alpha-1} \frac{e^{-x/\beta}}{\beta^\alpha \Gamma(\alpha)}, \quad x, \alpha, \beta > 0, \tag{3.11}$$

where: x is the particle size variable

α is the shape factor

β is the scale factor.

By replacing (3.11) into (3.10), the multimodal Gamma PSD function is obtained:

$$f(x) = \sum_{i=1}^{n} \omega_i x^{\alpha_i - 1} \frac{\exp(-x/\beta_i)}{\beta_i^{\alpha_i} \Gamma(\alpha_i)}. \tag{3.12}$$

In (3.12) the particle size corresponds to either the geometrical radius of the particle obtained through optical methods or the hydraulic radius measured through sedimentation techniques that employ Stokes' Law. The latter radius is the one considered in the following expression for the terminal settling velocity of a sphere suspended in an infinite fluid medium:

$$u_\infty = \frac{2(\rho_s - \rho_f)}{9\mu} g r^2, \tag{3.13}$$

where: u_∞ is the terminal settling velocity of a sphere of radius r

ρ_s is the mass density

ρ_f is the mass density of the infinite fluid medium

μ is the fluid's viscosity

g is the acceleration of gravity.

The hydraulic radius of a particle is therefore the equivalent radius of a sphere settling at the same terminal velocity given by (3.13). The advantage of using the hydraulic radius instead of the 'geometrical' radius is that fluctuations in the mass density of flocs and particles that are extremely difficult to measure are avoided.

If the total mass of the particles in the medium $m_T = \int_0^\infty x \cdot n(x,t)dx$ is normalized, then the PSD function of the aggregated particles $f(x,t)$ is equivalent to $x\,n(x,t)$ and:

$$\int_0^\infty f(x,t)dx = \int_0^\infty x \cdot n(x,t)dx = 1. \qquad (3.14)$$

3.2.5 Solutions to the Coagulation Equation using the Method of Moments

Solving the coagulation-fragmentation equation represents an important challenge due to the fact that it is a complex integro-differential equation. There are different solution techniques that can be used [3.8], such as an analytical or exact solution to the coagulation equation or by approximations or numerical methods (deterministic and stochastic). Exact solutions to the coagulation equation in both discrete (3.1) and continuous (3.2) forms have been found for three simple types of kernels: constant ($K(x,y)=c$), additive $\left(K(x,y)=x+y\right)$ and multiplicative$\left(K(x,y)=xy\right)$. Other analytical solutions have been found for the kernel $K(x,y)=A+B(x+y)+Cxy$ through the use of power series ([3.1] and [3.6]).

An alternate approach is to solve Smoluchowski's coagulation equation for the moments of the PSD function ([3.8], [3.9], and [3.24]). Obtaining the moments of the PSD function or the particle number concentration offers only a general image of the true shape of $f(x,t)$ or $n(x,t)$ since in most cases it is difficult to link them directly. However, in some cases it is possible to retrieve the PSD from a finite number of moments for a given point in time. The k^{th} moment $\left(M_k\right)$ of the PSD function is defined as:

$$M_k(t)=\int_0^\infty x^k f(x,t)dx, \quad k=0,1,2,\ldots \tag{3.15}$$

with $M_0 = \int_0^{\infty} f(x,t)dx = 1.$ Since the particle number concentration is equivalent to the PSD function when the total mass is normalized, the coagulation equation can be applied to the PSD function in the same manner. The only difference that occurs when applying the equivalency between the PSD and the number concentration is in the moments. In the case of the particle size distribution M_1 represents the average size particle and in the case of the number concentration it represents the total mass of particles per unit volume. This equivalency can easily be understood since the rate of aggregation and fragmentation must be the same when either the PSD $f(x,t)$ or the number concentration $n(x,t)$ is used. By replacing $n(x,t)$ in (3.2) with the PSD function $f(x,t)$ and calculating the corresponding k^{th} moment, we obtain:

$$\begin{aligned}\frac{dM_k}{dt} &= \frac{1}{2}\int_0^{\infty}\int_0^{x} x^k K(x, y-x) f(y-x,t) f(y,t)dydx \\ &\int_0^{\infty}\int_0^{\infty} x^k K(x,y) f(x,t) f(y,t)dydx.\end{aligned} \tag{3.16}$$

Introducing the step function $H(x-y)$ defined as:

$H(x-y) = 0$ if $x < y,$

$H(x) = 1$ otherwise then (3.16) is equivalent to:

$$\frac{dM_k}{dt} = \frac{1}{2}\int_0^\infty\int_0^\infty x^k H(x-y)K(x,y-x)f(y-x)f(y,t)dydx$$
$$\int_0^\infty\int_0^\infty x^k K(x,y)f(x,t)f(y,t)dydx \tag{3.17}$$

Switching the order of integration and substituting $x' = y - x$ on the first integral of the right-hand side of (3.17) yields the general form of the moments' equation [3.9] for the case of aggregation only:

$$\frac{dM_k}{dt} = \int_0^\infty\int_0^\infty \left[\frac{1}{2}(x+y)^k - x^k\right] K(x,y)f(x,t)f(y,t)dxdy \tag{3.18}$$

The variable x' was replaced by *y* in (3.18) without loss of generality.

In a similar fashion for the first fragmentation equation, (3.3) is multiplied by x^k and integrated between 0 and infinity, the moment equation for aggregation and fragmentation is obtained:

$$\frac{dM_k}{dt} = \int_0^\infty\int_0^\infty \left[\frac{1}{2}(x+y)^k - x^k\right] K(x,y)f(x,t)f(y,t)dxdy +$$
$$\int_x^\infty\int_0^\infty x^k f(y,t)\phi(y,x)dxdy - \int_0^x\int_0^\infty x^{k-1} f(x,t)y\phi(y,x)dxdy \tag{3.19}$$

Let G_1 and G_2 be the second and third double integrals of the right-hand side of (3.19), respectively:

$$G_1 = \int_x^\infty \int_0^\infty x^k f(y,t)\phi(y,x)dxdy, \text{ and } G_2 = \int_0^x \int_0^\infty x^{k-1} f(x,t)\phi(y,x)dxc$$

Introducing the step function $H(x)$ into G_1 where:

$H(x) = 0$ if $x < 0$,

$H(x) = 1$ otherwise, then:

$$G_1 = \int_0^\infty \int_0^\infty x^k H(y-x) f(y,t)\phi(y,x)dydx. \tag{3.20}$$

By substituting *u* by $y - x$ and switching the order of integration we obtain:

$$G_1 = \int_0^\infty \int_0^\infty x^k f(x+y,t)\phi(x,x+y)dxdy, \tag{3.21}$$

where *u* was replaced by *y* without loss of generality. Similarly, the third double integral of the right-hand side of (3.19) results in:

$$G_2 = \int_0^\infty \int_0^\infty (x+y)^{k-1} y f(x+y,t)\phi(x+y,y)dxc \tag{3.22}$$

The final general form of the moment equation results from combining (3.21) and (3.22) with (3.19):

$$\begin{aligned}\frac{dM_k}{dt} = &\int_0^\infty\int_0^\infty\left[\frac{1}{2}(x+y)^k - x^k\right]K(x,y)\,f(x,t)\,f(y,t)dxdy \\ &\int_0^\infty\int_0^\infty x^k f(x+y,t)\phi(x,x+y)dxdy - \\ &\int_0^\infty\int_0^\infty (x+y)^{k-1}\,y\,f(x+y,t)\phi(x+y,y)dxdy.\end{aligned} \tag{3.23}$$

Finally, adopting the alternate representation provided in (3.4), the general moment equation for aggregation and fragmentation can also be written as:

$$\begin{aligned}\frac{dM_k}{dt} = &\int_0^\infty\int_0^\infty\left[\frac{1}{2}(x+y)^k - x^k\right]K(x,y)\,f(x,t)\,f(y,t)dxdy + \\ &\int_x^\infty\int_0^\infty x^k f(y,t)\varphi(y)P(x,y)dxdy - \int_0^\infty x^k f(x,t)\varphi(x)d\end{aligned} \tag{3.24}$$

Both (3.23) and (3.24) are valid representations of the general moments' equation for aggregation and fragmentation.

3.3 Model Development

Sediment PSD can be modeled using a multimodal Gamma distribution. The moment method presented in Section 3.2.5 will be used to solve the coagulation equation when the PSD function is a multimodal Gamma probability density function. Four different cases will be analyzed:

1) coagulation with initial unimodal Gamma PSD function,

2) coagulation with initial multimodal Gamma PSD function

3) coagulation and fragmentation with initial unimodal Gamma PSD function

4) coagulation and fragmentation with initial multimodal Gamma PSD function.

3.3.1 Case 1: Coagulation with Initial Unimodal Gamma PSD Function

Replacing the general aggregation kernel for turbulent flow regime (3.8) and the Gamma unimodal probability function (3.11) into (3.18) yields:

$$\begin{aligned}\frac{dM_k}{dt} &= \frac{A}{\beta^{2\alpha}\Gamma^2(\alpha)}\int_0^\infty\int_0^\infty\left[\frac{1}{2}(x+y)^k + x^k\right]\left(x^r + y^r\right)^p x^{\alpha-1}y^{\alpha-1}e^{-x/\beta}e^{-y/\beta}dx \\ &= I_{1,k} - I_{2k}, \quad k = 1,2\ldots\end{aligned} \qquad (3.25)$$

where:

$$I_{1,k} = \frac{A}{2\beta^{2\alpha}\Gamma^2(\alpha)}\int_0^\infty\int_0^\infty (x+y)^k\left(x^r + y^r\right)^p x^{\alpha-1}y^{\alpha-1}e^{-x/\beta}e^{-y/\beta}dx \qquad (3.26)$$

and

$$I_{2,k} = \frac{A}{\beta^{2\alpha}\Gamma^{2}(\alpha)} \int_{0}^{\infty}\int_{0}^{\infty} x^{k}\left(x^{r}+y^{r}\right)^{p} x^{\alpha-1} y^{\alpha-1} e^{-x/\beta} e^{-y/\beta} dxc \tag{3.27}$$

Using the binomial theorem to calculate $\left(x^{r}+y^{r}\right)^{p}$, and adopting the variables substitutions $u = x/\beta$ and $v = y/\beta$, (3.26) results in:

$$I_{1,k} = \frac{A\beta^{k+rp}}{2\Gamma^{2}(\alpha)} \int_{0}^{\infty}\int_{0}^{\infty} \sum_{l=0}^{k}\binom{k}{l}\sum_{m=0}^{p}\binom{p}{m} u^{k-l+r(p-m)+\alpha-1} e^{-u} v^{l+rm+\alpha-1} e^{-v} duc \tag{3.28}$$

In addition, the Gamma function is defined by:

$$\Gamma(\theta) = \int_{0}^{\infty} \tau^{\theta-1} e^{-\tau} d\tau, \quad \theta > 0. \tag{3.29}$$

Moving the integrals inside the double sum and applying (3.29) yields:

$$I_{1,k} = \frac{A\beta^{k+rp}}{2\Gamma^{2}(\alpha)} \sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m} \Gamma(k-l+r(p-m)+\alpha)\Gamma(l+rm+c \tag{3.30}$$

The second integral $I_{2,k}$ is obtained in similar fashion:

$$I_{2,k} = \frac{A\beta^{k+rp}}{\Gamma^2(\alpha)} \sum_{m=0}^{p} \binom{p}{m} \Gamma\left(k + r(p-m) + \alpha\right)\Gamma\left(rm + c\right. \tag{3.31}$$

Note that in the two previous equations the parameters of the Gamma are a function of time due to the aggregation process: $\alpha \equiv \alpha(t)$ and $\beta \equiv \beta(t)$. A more detailed description of above equations derivation is provided in Appendix F.

Using the definition of the k^{th} moment (3.15), the left-hand side of (3.18) can be expanded to obtain a system of ODEs that only depend on the Gamma parameters $\alpha(t)$ and $\beta(t)$:

$$\frac{dM_k}{dt} = \frac{d}{dt}\int_0^{\infty} x^{k+\alpha-1} \frac{e^{-x/\beta}}{\beta^{\alpha}\Gamma(\alpha)} dx. \tag{3.32}$$

Substituting u by x/β and using the definition of the Gamma distribution (3.29), the above equation results in:

$$\frac{dM_k}{dt} = \frac{d}{dt}\left(\frac{\beta^k \Gamma(k+\alpha)}{\Gamma(\alpha)}\right) = \frac{d}{dt}\left(\beta^k \prod_{j=1}^{k} \left(k - j + \alpha\right.\right. \tag{3.33}$$

where the relationship $\Gamma(z+1) = z\Gamma(z)$ was used. Applying the chain rule on the right-hand side of (3.33):

$$\frac{dM_k}{dt} = \frac{\partial M_k}{\partial \alpha}\frac{d\alpha}{dt} + \frac{\partial M_k}{\partial \beta}\frac{d\beta}{dt} \tag{3.34}$$

then the moments' equations results in:

$$\begin{aligned}\frac{dM_k}{dt} &= \beta^k \sum_{q=1}^{k}\left(\prod_{j\neq q}^{k}(k-j+\alpha)\right)\frac{d\alpha}{dt} + k\beta^{k-1}\prod_{j=1}^{k}(k-j+\alpha) \\ &= I_{1,k} - I_{2,k},\end{aligned} \tag{3.35}$$

where $I_{1,k}$ and $I_{2,k}$ are given by (3.30) and (3.31), respectively. Obtaining $\alpha(t)$ and $\beta(t)$ enables to determining the PSD function and its cumulative distribution assuming that the PSD remains unimodal over time. In reality sediments are multimodal in nature and thus it is necessary to expand the previous analysis to PSD functions having more than one mode.

3.3.2 Case 2: Coagulation with initial Multimodal Gamma PSD Function

The unimodal case derived in the previous section was expanded to include the multimodal form of the PSD function (3.10). Replacing (3.10) into the general moment equation (3.18) results in:

$$\frac{dM_k}{dt} = \sum_{i=1}^{n}\sum_{j=1}^{n}\omega_i\omega_j\left(I_{1,ijk} - I_{2,ijk}\right) \tag{3.36}$$

with

$$\begin{aligned} I_{1,ijk} &= \frac{A}{2\beta_i^{\alpha_i}\beta_j^{\alpha_j}\Gamma(\alpha_i)\Gamma(\alpha_j)}\int_0^{\infty}\int_0^{\infty}\left(x+y\right)^k\left(x^r + y^r\right)^p x^{\alpha_i - 1}e^{-x/\beta_i}y^{\alpha_i - 1}e^{-x/\beta_i}dxdy \\ &= \frac{A}{2\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\Big(\beta_i^{k-l+r(p-m)}\beta_j^{l+rm}\ \cdot \\ &\qquad \cdot\Gamma(k-l+r(p-m)+\alpha_i)\Gamma(l+rm+\alpha_j)\Big) \end{aligned} \tag{3.37}$$

and

$$\begin{aligned} I_{2,ijk} &= \frac{A}{\beta_i^{\alpha_i}\beta_j^{\alpha_j}\Gamma(\alpha_i)\Gamma(\alpha_j)}\int_0^{\infty}x^k\left(x^r + y^r\right)^p x^{\alpha_i - 1}e^{-x/\beta_i}y^{\alpha_i - 1}e^{-x/\beta_i}dxdy \\ &= \frac{A}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{m=0}^{p}\binom{p}{m}\beta_i^{k+r(p-m)}\beta_j^{rm}\Gamma(k+r(p-m)+\alpha_i)\Gamma(rm+\alpha \end{aligned} \tag{3.38}$$

The moments in multimodal form are given by:

$$M_k = \sum_{i=1}^{n}\omega_i\int_0^{\infty}xf_i(x)dx = \sum_{i=1}^{n}\omega_i\beta_i^k\prod_{j=1}^{k}(k-j+\alpha \tag{3.39}$$

and by applying the chain rule and differentiating each term yields:

$$\begin{aligned}\frac{dM_k}{dt} &= \sum_{i=i}^{n}\left\{\beta_i^{k-1}\left[\beta_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\omega_i}{dt}+\right.\right.\\ &\quad +\omega_i\beta_i\sum_{q=1}^{k}\prod_{j\neq q}^{k}(k-j+\alpha_i)\frac{d\alpha_i}{dt}+k\omega_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\beta_i}{dt}\\ &=\sum_{i=1}^{n}\sum_{j=1}^{n}\omega_i\omega_j\left(I_{1,ijk}-I_{2,ijk}\right).\end{aligned}\tag{3.40}$$

Finally, the above equation provides a system of ODEs of the functions $\alpha_i(t)$, $\beta_i(t)$ and $\omega_i(t)$, for i = 1...n. For each mode a set of three ODEs is therefore needed and hence the total required number of ODEs to solve the problem is $3n = k$. Note that if n = 1 we obtain the unimodal equation (35) with $\omega_1 = 1$, and $d\omega_1/dt = 0$.

3.3.3 Case 3: Coagulation and Fragmentation with Initial Unimodal Gamma PSD Function

The aggregation-fragmentation model for both the unimodal and multimodal cases will be developed using the equation given by (3.24). Since the first double integral of the right-hand side of (3.24) (namely the aggregation term) has already been solved in Section 3.3.1, it is only necessary to consider the solution of the second and third double integrals (namely the fragmentation term) in this analysis. Denote by $J_{1,k}$ and $J_{2,k}$ the two fragmentation double integrals:

$$J_{1,k} = \int_{x}^{\infty}\int_{0}^{\infty} x^k f(y,t)\varphi(y)P(x,y)dxdy, \tag{3.41}$$

and

$$J_{2,k} = \int_0^\infty x^k f(x,t)\varphi(x)dx. \tag{3.42}$$

The overall fragmentation rate $\varphi(x)$ is given in (3.9) where the coefficients $B, \lambda \in \mathbf{R}^+$. There is no defined expression for the fragmentation distribution $P(x,y)$. Since the Gamma distribution model provide an excellent fit of the PSD of sediment particles, it is reasonable to assume that the PSD of the fragments should also follow a Gamma distribution however with different parameters (e.g. a and b instead of α and β, respectively). Therefore the fragments' PSD is given by:

$$P(x,y) = f(y/x) = \frac{(y/x)^{a-1}\exp(-y/bx}{b^a\Gamma(a)} \tag{3.43}$$

Clearly $P(x,y) \geq 0$ and $\int_0^\infty P(x,y)dy = 1$. Combining (3.43) and (3.9) into (3.24) yields:

$$J_{1,k} = \frac{Bb^{\lambda+\alpha-1}\beta^{\lambda+a-1}}{\Gamma(a)\Gamma(\alpha)}\int_0^\infty \frac{x^{k+\lambda+\alpha}}{(bx+\beta)^{\lambda+a+\alpha-1}}\Gamma(\lambda+a+\alpha-1,x)c \tag{3.44}$$

where $\Gamma(a,x)=\int_{x}^{\infty} z^{a-1}e^{-z}dz$ is the *incomplete Gamma function* and

$$J_{2,k}=\frac{B\beta^{k+\lambda}}{\Gamma(\alpha)}\Gamma(k+\lambda+\alpha). \tag{3.45}$$

The first term of the fragmentation moment equation J_1 can't be solved analytically and thus a numerical method is needed. There are several numerical integration methods available to solve the indefinite integral in (3.44). Finally the moment equation for aggregation and fragmentation considering a unimodal Gamma initial PSD is given by:

$$\frac{dM_k}{dt}=I_{1,k}-I_{2,k}+J_{1,k}-J_{2,k}, \tag{3.46}$$

where $I_{1,k}$ and $I_{2,k}$ are given by (3.30) and (3.31) respectively and $J_{1,k}$ and $J_{2,k}$ are given by (3.44) and (3.45), respectively. The derivative of the Moments' equation with respect to time, $\frac{dM_k}{dt}$, is obtained by the chain rule:

$$\frac{dM_k}{dt}=\beta^k\sum_{q=1}^{k}\left(\prod_{j\neq q}^{k}(k-j+\alpha)\right)\frac{d\alpha}{dt}+k\beta^{k-1}\prod_{j=1}^{k}(k-j+\alpha)\frac{d\beta}{dt}.$$

3.3.4 Case 4: Coagulation and Fragmentation with Initial Multimodal Gamma PSD Function

Following the approach from Section 3.3.2 let $J_{1,ik} - J_{2,ik}$ be the fragmentation rate of the moment equation for the multimodal case:

$$J_{1,k} - J_{2,k} = \sum_{i=1}^{n} \omega_i \left(J_{1,ik} - J_{2,ik} \right). \tag{3.47}$$

where:

$$J_{1,ijk} = \frac{Bb^{\lambda+\alpha_i-1}\beta_i^{\lambda+a-1}}{\Gamma(a)\Gamma(\alpha_i)} \int_0^{\infty} \frac{x^{k+\lambda+\alpha_i}}{\left(bx+\beta_i\right)^{\lambda+a+\alpha_i-1}} \Gamma(\lambda + a + \alpha_i - 1, x)c \tag{3.48}$$

and

$$J_{2,ik} = \frac{B\beta_i^{k+\lambda}}{\Gamma(\alpha_i)} \Gamma(k + \lambda + \alpha_i). \tag{3.49}$$

The moments' change rate $\frac{dM_k}{dt}$ in its multimodal form is given by (3.40). The complete development of the fragmentation terms is presented in Appendix G.

3.4 Summary and Concluding Remarks

The Smoluchowski's continuous coagulation equation was introduced to model the binary rate of aggregation between sediment particles of different sizes. Attempting to find an exact solution for this integro-differential equation for a specific problem can be arduous. Analytical solutions only exist for a reduced set of expressions for the aggregation kernels which don't usually have a practical realistic application. Furthermore, these exact solutions assume an initial PSD which is uniform (small differences in particle sizes). Due to these limitations it is necessary to apply other techniques to solve the coagulation equation: these include numerical methods, power series, and the method of moments.

The method of moments was chosen because it provides an alternate approach to the coagulation problem instead of solving directly for the PSD function. To interpret the solutions of the moments' equation to the PSD as a function of time, it was first necessary to develop a 'general' moment equation in which the limits of the integrals remained between $[0,\infty)$ (which is consistent with a time variable). This new expression for the moments' equation combined with the assumption that the initial PSD follows a Gamma distribution and using a general expression for the coagulation kernel enabled obtaining an analytical expression of the integrals and therefore reducing the moments' equations to a simple system of ODE's.

In Chapter 2 the multimodal Gamma distribution was used to model the PSD of sediment particles with excellent results. Using this mathematical representation of the initial PSD (before coagulation) it was possible to analytically solve the integrals of the right-hand side of the general moments' equation for the unimodal case. The multimodal case was a simple extension of the unimodal case due to the linearity of the problem.

The next step in this investigation was to include the fragmentation of aggregates. Two equivalent models were described and only one was chosen to be included in the moment equation. The approach mentioned in [3.9] was used to obtain an expression for the general moment equation when considering fragmentation. A general expression for the overall fragmentation rate was developed based on the fragmentation kernels in the presence of turbulent shear forced obtained from the literature.

It is extremely difficult to accurately determine the joint probability of two random variables representing the size of the particle that has been created by the fragmentation of the second particle. Since this joint probability function is needed to solve the general moments' equation, it was necessary to assume a simplifying hypothesis (namely that both random variables are independent from each other). The same analysis conducted for the cases that considered only coagulation was applied for the case where the fragmentation was added.

The results of the moment equation considering aggregation and fragmentation for both the unimodal and multimodal cases are presented in Table 3 - 1.

Table 3 - 1: Solutions to the integrals of the moments' equation

Case 1: Coagulation with initial unimodal Gamma PSD function$^{(a)}$
$\beta^k \sum_{q=1}^{k}\left(\prod_{j\neq q}^{k}(k-j+\alpha)\right)\frac{d\alpha}{dt}+k\beta^{k-1}\prod_{j=1}^{k}(k-j+\alpha)\frac{d\beta}{dt}=I_{1,k}-I_{2,k}$ with $I_{1,k}=\frac{A\beta^{k+rp}}{2\Gamma^2(\alpha)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\Gamma(k-l+r(p-m)+\alpha)\Gamma(l+rm+\alpha),$ and $I_{2,k}=\frac{A\beta^{k+rp}}{\Gamma^2(\alpha)}\sum_{m=0}^{p}\binom{p}{m}\Gamma(k+r(p-m)+\alpha)\Gamma(rm+\alpha).$
Case 2: Coagulation with initial multimodal Gamma PSD function$^{(a)}$
$\sum_{i=i}^{n}\Bigg\{\beta_i^{k-1}\Bigg[\beta_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\omega_i}{dt}+$ $+\omega_i\beta_i\sum_{q=1}^{k}\prod_{j\neq q}^{k}(k-j+\alpha_i)\frac{d\alpha_i}{dt}+k\omega_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\beta_i}{dt}\Bigg]\Bigg\}=\sum_{i=1}^{n}\sum_{j=1}^{n}\omega_i\omega_j\left(I_{1,ijk}-I_{2,ijk}\right)$ with $I_{1,ijk}=\frac{A}{2\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\beta_i^{k-l+r(p-m)}\beta_j^{l+rm}\Gamma(k-l+r(p-m)+\alpha_i)\Gamma(l+rm+$ and $I_{2,ijk}=\frac{A}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{m=0}^{p}\binom{p}{m}\beta_i^{k+r(p-m)}\beta_j^{rm}\Gamma(k+r(p-m)+\alpha_i)\Gamma(rm+\alpha_j).$
Case 3: Coagulation and Fragmentation with initial unimodal Gamma PSD function$^{(b)}$
$\beta^k \sum_{q=1}^{k}\left(\prod_{j\neq q}^{k}(k-j+\alpha)\right)\frac{d\alpha}{dt}+k\beta^{k-1}\prod_{j=1}^{k}(k-j+\alpha)\frac{d\beta}{dt}=I_1^k-I_2^k+J_1^k-J_2^k$ with $I_1^k=\frac{A\beta^{k+rp}}{2\Gamma^2(\alpha)}\sum_{l=0}^{k}\binom{k}{l}\sum_{m=0}^{p}\binom{p}{m}\Gamma(k-l+r(p-m)+\alpha)\Gamma(l+rm+\alpha),$ $I_2^k=\frac{A\beta^{k+rp}}{\Gamma^2(\alpha)}\sum_{m=0}^{p}\binom{p}{m}\Gamma(k+r(p-m)+\alpha)\Gamma(rm+\alpha),$ and $J_{1,k}=\frac{Bb^{\lambda+\alpha-1}\beta^{\lambda+a-1}}{\Gamma(a)\Gamma(\alpha)}\int_0^{\infty}\frac{x^{k+\lambda+\alpha}}{(bx+\beta)^{\lambda+a+\alpha-1}}\Gamma(\lambda+a+\alpha-1,x)dx,$ $J_{2,k}=\frac{B\beta^{k+\lambda}}{\Gamma(\alpha)}\Gamma(k+\lambda+\alpha).$

Table 3 - 1 (cont.):

Case 4: Coagulation and Fragmentation with initial multimodal Gamma PSD function[b]
$\sum_{i=i}^{n}\left\{\beta_i^{k-1}\left[\beta_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\omega_i}{dt}+\omega_i\beta_i\sum_{q=1}^{k}\prod_{j\neq q}^{k}(k-j+\alpha_i)\frac{d\alpha_i}{dt}+\right.\right.$ $\left.\left.+k\omega_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\beta_i}{dt}\right]\right\}=\sum_{i}^{n}\sum_{j}^{n}\omega_i\omega_j\left(I_{1,ijk}-I_{2,ijk}\right)+\sum_{i=1}^{n}\omega_i\left(J_{1,ik}-J_{2,ik}\right),$ with $I_{1,ijk}=\frac{A}{2\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\beta_i^{k-l+r(p-m)}\beta_j^{l+rm}\Gamma(k-l+r(p-m)+\alpha_i)\Gamma(l+rm+\alpha_j),$ $I_{2,ijk}=\frac{A}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{m=0}^{p}\binom{p}{m}\beta_i^{k+r(p-m)}\beta_j^{rm}\Gamma(k+r(p-m)+\alpha_i)\Gamma(rm+\alpha_j),$ and $J_{1,ik}=\frac{Bb^{\lambda+\alpha_i-1}\beta_i^{\lambda+a-1}}{\Gamma(a)\Gamma(\alpha_i)}\int_0^{\infty}\frac{x^{k+\lambda+\alpha_i}}{\left(bx+\beta_i\right)^{\lambda+a+\alpha_i-1}}\Gamma(\lambda+a+\alpha_i-1,x)dx,$ $J_{2,ik}=\frac{B\beta_i^{k+\lambda}}{\Gamma(\alpha_i)}\Gamma(k+\lambda+\alpha_i).$

Notes: [a] See Appendix F for derivation details.

[b] See Appendix G for derivation details.

CHAPTER 4: Modeling the Flocculation of Fine Sediments using the Method of Moments to Solve the Coagulation Equation – Numerical Results

Abstract

The use of a Gamma distribution as a mathematical model to represent the particle size distribution (PSD) of fine sediment particles enables representing the changes in PSD due to flocculation through a system of ordinary differential equations (ODE). This simple system of ODEs was solved numerically for both aggregation and fragmentation processes in an effort to model the rapid flocculation of fine sediment particles using different anionic polyacrylamide (APAM) flocculants. For this purpose, a set of 4 different sediment samples were used to obtain the initial PSD data (no APAM was added) and for each sediment sample 4 different APAM flocculants were used to obtain the flocculated PSDs after applying 2 minutes of turbulent mixing. Both the initial (non-flocculated) and flocculated PSD data points were obtained through the use of an X-Ray particle size analyzer that utilizes the sedimentation technique. The initial PSD was modeled using a bimodal Gamma distribution. A simple genetic algorithm was developed to calibrate the aggregation-fragmentation parameters to the observed flocculated PSDs for the bimodal case and the results show excellent correlation between predicted and observed PSDs ($R^2 > 0.9878$).

4.1 Introduction

Induced aggregation of fine solid particles through the use of chemical flocculation is of major importance in the treatment of water for human consumption (e.g. water treatment plants) and it is gradually being considered for the treatment of surface water systems (e.g. runoff from construction/mining activities). Mining and construction activities require that a sizeable area be stripped from its natural vegetative cover enabling surface water runoff to erode and transport the fine sediment particles towards natural water bodies. If sediment is not treated it can cause detrimental effects to the aquatic bio-environment.

The most common sediment control system encountered in construction and mining activities is the use of a sediment pond where flow velocities through the structure are considerably reduced enabling for the coarser particles to settle and be trapped. The trapping efficiency of a sediment pond depends mainly on two factors: flow velocity through the structure and particle size distribution (PSD) of the incoming sediment. To improve the trapping efficiency, it is therefore necessary to alter one or both of these factors.

The anionic version of polyacrylamides (PAM) is used in natural environments since it is considered to be environmentally safe compared to its cationic counterpart. Anionic polyacrylamides (APAM) are long chain polymers used for flocculation of fine solid particles. Nowadays, APAM flocculants are mainly used in large agricultural areas to reduce erosion by scattering water-soluble APAM across the surface through irrigation sprinklers. When runoff starts eroding the upper soil layer (topsoil) the APAM flocculant intervenes forming larger particles (flocs) that remain on-site. Recommended by the United States Natural Resources Conservation System [4.14] as a soil conservation practice, the application of APAM over the land surface not only reduces soil migration from the agricultural land, but also reduces the sediment concentrations in runoff; thus protecting the surrounding natural aquatic environment. Roa-Espinosa et al. [4.16] conducted a series of experiments where a PAM flocculant was applied on the surface of small soil plots and found that sediment yield was considerably reduced. Even though the land application of APAM flocculants has seen an increase in the past recent years in soil conservation and erosion control practices, its use is still minimal.

Another important aspect in flocculation processes is fragmentation or floc breakup. A floc will continue growing (adhering to other particles or flocs) up to a point where the resistant forces of the floc structure become inferior to the shear forces in the fluid. Therefore, excessively increasing the turbulent mixing energy could be detrimental to the flocculation process even though the probability of particle-to-particle collision increases.

4.2 Theory

In a turbulent mixing zone the aggregation of particles induced by fluid shear forces is the dominant factor progressing the PSD to the coarser side and the average particle diameter increases.

When particle-to-particle coagulation is enhanced by the use of an APAM flocculant the aggregation process is significantly accelerated: flocs are formed in a matter of a few minutes time. The rate of change of the particle number concentration, *n*, is given by the continuous form of the Smoluchowski coagulation equation [4.6]:

$$\begin{aligned}&\frac{\partial n(x,t)}{\partial t}=\frac{1}{2}\int_0^x K(y,x-y)n(y,t)n(x-y,t)dy-n(x,t)\int_0^\infty K(x,y)m(y,t)d\\&s.t.\quad n(x,0)=n_0(x),\end{aligned} \tag{4.1}$$

where x and y denote the continuous variables representing the mass of the particles, $K(x,y)$ is the aggregation kernel, and $n_0(x)$ is a continuous function that represents the initial particle number distribution. The first integral on the right-hand side of (4.1) represents the increase in the particle number concentration of particles of mass *x* when a particle of mass $x-y$ (with $y < x$) comes in contact with a particle of mass *y*. The constant ½ accounts for the symmetry of the aggregation kernel (i.e. $K(x,y)=K(y,x)$) and the second integral represents the decrease in the number concentration of particles of mass *x* when they aggregate with other particles.

A floc is formed when two particles bind together. In turn, this floc can also grow, since this binding process can repeat itself many times over; however, a point in time is reached where the floc is no longer stable due to its attained size and fragmentation commences. To correctly model the fragmentation process, a new term describing the changes in the PSD due to floc breakup must be inserted to (4.1). The fragmentation process is represented by the following resulting expression:

$$\begin{aligned} &\frac{\partial n(x,t)}{\partial t} = \int_x^\infty n(y,t)\varphi(y)P(x,y)dy - \varphi(x)n(x, \\ &s.t. \quad n(x,0) = n_0(x). \end{aligned} \tag{4.2}$$

where the first integral on the right-hand side of (4.2) represents the gain in number concentration when a particle larger than x (in mass) breaks up into fragments of mass x and the second term represents the loss in number concentration when a particle x segments into smaller fragments. The fragmentation kernel $\varphi(x)$ accounts for the rate of segmentation of particles of mass x and $P(x,y)$ represents the distribution of the fragmented particles.

The general aggregation-fragmentation equation results when both (4.1) and (4.2) are combined. If the number concentration is normalized, the obtained expression is analogous to the PSD in mass $f(x)$, and therefore the general aggregation-fragmentation equation can be written in terms of the PSD as:

$$\frac{\partial f(x,t)}{\partial t} = \frac{1}{2}\int_0^x K(y,x-y) f(y,t) f(x-y,t)dy - f(x,t)\int_0^\infty K(x,y) f(y,t)dy + \int_x^\infty f(y,t)\varphi(y)P(x,y)dy - \varphi(x) f(x,t), \tag{4.3}$$

$$s.t. \quad f(x,0) = f_0(x).$$

Solving the above equation is no easy task and depends on the chosen expressions for the aggregation and fragmentation Kernels. The most common technique is to use a numerical method (e.g. [4.12]). Another approach to solve (4.3) is to use the moments' representation of the PSD where the k^{th} moment of the PSD is given by:

$$M_k = \int_0^\infty x^k f(x)dx. \tag{4.4}$$

For k = 1, $M_1 = 1$ since the PSD function is analogous to a probability distribution density function. For k = 2, the second moment M_2 represents the average particle size.

Multiplying (4.3) by x^k on both sides and integrating between 0 and $+\infty$ the general moments' equation for both aggregation and fragmentation processes is obtained (Chapter 3):

$$\frac{dM_k}{dt} = \int_0^\infty \int_0^\infty \left[\frac{1}{2}(x+y)^k - x^k \right] K(x,y) f(x,t) f(y,t) dxdy + \\ \int_x^\infty \int_0^\infty x^k f(y,t)\varphi(y) P(x,y) dxdy - \int_0^\infty x^k f(x,t)\varphi(x) \tag{4.5}$$

When collision between two particles is driven by homogeneous turbulent shear forces in the fluid the aggregation Kernel takes the following form [4.17]:

$$K(x,y) = A\left(x^{1/3} + y^{1/3}\right)^3, \tag{4.6}$$

where A is a coefficient that depends on the physical problem. In Chapter 3 the use of a more generalized aggregation kernel was proposed and is given by:

$$K(x,y) = A\left(x^r + y^r\right)^p, \tag{4.7}$$

where $r > 0$ and p =1,2… In addition, imposing that the aggregation kernel is a homogeneous function of degree κ:

$$K(cx,cy) = c^\kappa K(x,y), \tag{4.8}$$

yields $\kappa = rp.$

The fragmentation rate or kernel takes the following general expression (e.g. [3.11]):

$$\varphi(x) = Bx^{\lambda}, \tag{4.9}$$

where B and λ are coefficients that depend on the physics of the problem. There is no defined expression for the distribution of fragmented particles based on the original aggregate size; however most authors agree that it should be a function of the particle size distribution of the particles defined by the quotient between the original aggregate and the size of the fragments:

$$P(x, y) \propto f(x/y). \tag{4.10}$$

4.3 Solutions to the General Moments' Equation

The general moments' equation (4.5) was solved in Chapter 3 assuming that the PSD follows a multimodal Gamma distribution. This assumption is founded on results obtained from comparing several different sediment PSD data with unimodal, bimodal and trimodal Gamma distributions (Chapter 2). These results showed that the use of a bimodal gamma distribution was an accurate model to fit sediment PSD. The multimodal Gamma distribution is given by:

$$f(x) = \sum_{i=1}^{n} \omega_i \frac{x^{\alpha_i - 1} e^{-y/\beta_i}}{\beta_i^{\alpha_i} \Gamma(\alpha_i)}, \tag{4.11}$$

where: n is the number of modes,

x is the particle's hydraulic diameter,

$\alpha_i > 0$ is the shape parameter of the Gamma distribution of mode i,

$\beta_i > 0$ is the scale parameter of the Gamma distribution of mode i,

$\Gamma(s) = \int_0^\infty x^{s-1} e^{-x} dx$ is the Gamma function, and

$\omega_i > 0$ is the mass proportion of mode i, with $\sum_{i=1}^{n} \omega_i = 1$.

Note that since the PSD changes over time due to flocculation, the multimodal Gamma distribution presented in (4.11) and therefore all of the parameters $(\omega_i, \alpha_i, \beta_i, i = 1,2,\ldots n)$ are time-dependent.

4.3.1 Aggregation

The aggregation term is given by the first double integral on the right-hand side of the general moments' equation (4.5). Using the aggregation kernel described in (4.7) and assuming that a multimodal Gamma PSD represents the initial condition of the aggregation process, the following system of ordinary differential equation (ODE) is obtained:

$$\begin{aligned}
&\sum_{i=i}^{n}\left\{\beta_i^{k-1}\left[\beta_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\omega_i}{dt}+\right.\right.\\
&\qquad\omega_i\beta_i\sum_{q=1}^{k}\prod_{j\neq q}^{k}(k-j+\alpha_i)\frac{d\alpha_i}{dt}+k\omega_i\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\beta_i}{dt}\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}\omega_i\omega_j\left(I_{1,ijk}-I_{2,ijk}\right),\quad k=2,3,\ldots 3n
\end{aligned}\tag{4.12}$$

where n is the number of modes in the PSD,

$$\begin{aligned}
I_{1,ijk}=\frac{A}{2\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\beta_i^{k-l+r(p-m)}\beta_j^{l+rm}\Gamma(k-l+r(p-m)+\alpha_i)\cdot\\
\Gamma(l+rm+\alpha_j),
\end{aligned}\tag{4.13}$$

and

$$I_{2,ijk}=\frac{A}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{m=0}^{p}\binom{p}{m}\beta_i^{k+r(p-m)}\beta_j^{rm}\Gamma(k+r(p-m)+\alpha_i)\Gamma(rm+\alpha_j).\tag{4.14}$$

The right-hand side of (4.12) is a positive monotonic increasing function, which represents the rate of aggregation changes of the moments of the multimodal Gamma distribution function. These moments have been previously expressed in function of the parameters $(\omega_i, \alpha_i, \beta_i)$ transforming the moments' equation into a system of ODEs. The solution of this resulting system provides an expression for the evolution over time of the parameters that define the PSD of the aggregated particles.

4.3.2 Aggregation and Fragmentation

The second and third terms of the right-hand side of (4.5) define the fragmentation terms of the general moments' equation. Introducing a step function:

$H(y-x)=1$ if $x>y$

and $H(y-x)=0$ otherwise, the first fragmentation term results in:

$$\int_{x}^{\infty}\int_{0}^{\infty} x^k \varphi(y) P(x,y) f(y,t) dxdy = \int_{0}^{\infty}\int_{0}^{\infty} H(y-x) x^k \varphi(y) P(x,y) f(y,t) dxc \tag{4.15}$$

Substituting $y-x$ by v yields:

$$\begin{aligned}\int_{x}^{\infty}\int_{0}^{\infty} x^k \varphi(y) P(x,y) f(y,t) dxdy &= \int_{-x}^{\infty}\int_{0}^{\infty} H(v) x^k \varphi(x+v) P(x,x+v) f(x+v,t) dx \\ &= \int_{0}^{\infty}\int_{0}^{\infty} x^k \varphi(x+y) P(x,x+y) f(x+y,t) dxdy,\end{aligned} \tag{4.16}$$

where *v* was replaced by *y* without any loss of generality.

An expression for the distribution of the fragments was provided in Chapter 3; however the first fragmentation term (4.16) resulted in an integro-differential equation, the solution of which requires to numerically solving the integral. In other to avoid this problem the following simplifying expression for $P(x, x+y)$ is introduced:

$$P(x, x+y) = f(x)\left(\frac{y}{x+y}\right)^{\tau} = \frac{x^{a-1}e^{-x/b}}{b^{a}\Gamma(a)}\left(\frac{y}{x+y}\right) \qquad (4.17)$$

where *a* and *b* are the average scale and shape parameters, respectively and thus:

$$a = \sum_{i=1}^{n}\omega_i\alpha_i,\ \ b = \sum_{i=1}^{n}\omega_i\beta_i. \qquad (4.18)$$

Introducing (4.18) into (4.16) and replacing the exponent τ by $\lambda+\alpha_i-1$ yields:

$$\begin{aligned} J_{1,ik} &= \int_0^{\infty}\int_0^{\infty} x^k\varphi(x+y)P(x,x+y)f(x+y,t)dxdy = \\ &= \frac{Bb^k\Gamma(k+a-1)}{\Gamma(a)}\sum_{i=1}^{n}\omega_i\frac{\beta_i^{\lambda+k+a}}{(\beta_i+b)^{k+a}\Gamma(\alpha_i)}\Gamma(\lambda+\alpha_i - \end{aligned} \qquad (4.19)$$

The second and last fragmentation term is given by:

$$J_{2,ik} = \int_0^\infty x^k f(x,t)\varphi(x)dx = B\sum_{i=}^{n}\frac{\beta_i^{k+\lambda}}{\Gamma(\alpha_i)}\Gamma(k+\lambda+\alpha \tag{4.20}$$

Finally, the multimodal aggregation-fragmentation moments' equation is given by the following system of ODEs:

$$\begin{aligned}\sum_{i=i}^{n}\Bigg[&\beta_i^{k-1}\prod_{j=2}^{k}(k-j+\alpha_i)\frac{d\omega_i}{dt}+\omega_i\beta_i^{k-1}\sum_{q=1}^{k}\prod_{\substack{j\neq q\\ j\geq 2}}^{k}(k-j+\alpha_i)\frac{d\alpha_i}{dt}+\\ &+k\omega_i\beta_i^{k-2}\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\beta_i}{dt}\Bigg]=\sum_{i}^{n}\sum_{j}^{n}\omega_i\omega_j\left(I_{1,ijk}-I_{2,ijk}\right)+\sum_{i=1}^{n}\omega_i\left(J_{1,ik}-J_{2,il}\right.\end{aligned} \tag{4.21}$$

where $I_{1,ijk}$ and $I_{2,ijk}$ are given by (4.13) and (4.14), respectively and $J_{1,ik}$ and $J_{2,ik}$ are given by (4.19) and (4.20), respectively. Note finally that the fragmentation function $\sum_{i=1}^{n}\omega_i\left(J_{1,ik}-J_{2,ik}\right), k=1\ldots 3n$ is a monotonous decreasing function of *k*.

4.4 Time Evolution of the General Moments' Equation

The general moments' equation (4.21) represents a system of ODEs, which is difficult to solve analytically due to complex expressions of the considered functions; therefore the use of a numerical method suggests itself as a practical approach. The most widespread numerical methods for solving ODEs are perhaps the family of Runge-Kutta methods, with explicit and implicit forms depending on the desired accuracy. The common forth order Runge-Kutta method (RK4) will be used to obtain the expression for the evolution over time of the general moments' equation.

4.4.1 Multivariate Fourth Order Runge-Kutta (RK4) Algorithm

The motivation is to numerically solve the system of ODEs (4.21) that represent the general moments' equation of the aggregation and fragmentation rates involved in the flocculation of sediment particles.

Assume that the following system of ODEs needs to be solved numerically:

$$\dot{\mathbf{v}} = \mathbf{f}(\mathbf{v}), \quad \mathbf{v}(t = 0) = \mathbf{v}_0 \tag{4.22}$$

where $\mathbf{v} = (v_1, v_2, \ldots v_m)^{\mathrm{T}}$, with $v_i : \mathbf{R}^+ \to \mathbf{R}$ a function of time,

$\mathbf{f}(\mathbf{v}) = (f_1(\mathbf{v}), f_2(\mathbf{v}), \ldots f_m(\mathbf{v}))^{\mathrm{T}}$, with $f_i : \mathbf{R}^m \to \mathbf{R}$, and

m denotes the number of ODEs in the system ($m = 3n$).

The over-dot denotes the time derivative: $\dot{x} = dx/dt$. The time evolution of the RK4 algorithm for the multivariate case is:

$$
\begin{aligned}
&t_{i+1} = t_i + h \\
&\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \\
&\text{with :} \\
&\mathbf{k}_1 = \mathbf{f}(\mathbf{v}_i), \\
&\mathbf{k}_2 = \mathbf{f}(\mathbf{v}_i + \frac{h}{2}\mathbf{k}_1), \\
&\mathbf{k}_3 = \mathbf{f}(\mathbf{v}_i + \frac{h}{2}\mathbf{k}_2), \\
&\mathbf{k}_4 = \mathbf{f}(\mathbf{v}_i + h\mathbf{k}_3).
\end{aligned}
\tag{4.23}
$$

where h denotes the time increment, i.e. $t_{i+1} = t_i + h$.

Let $\mathbf{v} = (v_1, v_2, \ldots v_n)^{\mathrm{T}}$ be the vector containing the multimodal Gamma parameters and mass proportions which are time-dependent, and are needed to obtain the flocculated PSD. The general moments' equation (4.21) can then be written as:

$$
\begin{aligned}
&\mathbf{M}\dot{\mathbf{v}} = \mathbf{I}(\mathbf{v}) + \mathbf{J}(\mathbf{v}) \\
&\Rightarrow \dot{\mathbf{v}} = \mathbf{M}^{-1}\left(\mathbf{I}(\mathbf{v}) + \mathbf{J}(\mathbf{v})\right),
\end{aligned}
\tag{4.24}
$$

where $\mathbf{M}$ is a square matrix containing all the derivative coefficients shown in the left-hand side of (4.21), thus

$$\mathbf{M} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} & a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{k,1} & w_{k,2} & \cdots & w_{k,n} & a_{k,1} & a_{k,2} & \cdots & a_{k,n} & b_{k,1} & b_{k,2} & \cdots & b_{k,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{3n,1} & w_{3n,2} & \cdots & w_{3n,n} & a_{3n,1} & a_{3n,2} & \cdots & a_{3n,n} & b_{3n,1} & b_{3n,2} & \cdots & b_{3n,n} \end{bmatrix} \tag{4.25}$$

where: $w_{k,i} = \beta_i^{k-1} \prod_{j=2}^{k} (k - j + \alpha_i),$

$$a_{k,i} = \omega_i \beta_i^{k-1} \sum_{q=2}^{k} \prod_{\substack{j \neq q \\ j \geq 2}}^{k} (k - j + \alpha_i),$$

$$b_{k,i} = k\omega_i \beta_i^{k-2} \prod_{j=2}^{k} (k - j + \alpha_i),$$

$k = 1 \ldots 3n, i = 1 \ldots n,$

and: $\dot{\mathbf{v}} = \begin{pmatrix} \dot{\omega}_1 & \cdots & \dot{\omega}_n & \dot{\alpha}_1 & \cdots & \dot{\alpha}_n & \dot{\beta}_1 & \cdots & \dot{\beta}_n \end{pmatrix}_{3n\times 1}^T,$

$\mathbf{I}(\mathbf{v}) = \left(\sum_i^n \sum_j^n \omega_i \omega_j \left(I_{1,ijk} - I_{2,ijk} \right) \right)_{3n\times 1}^T$ is the vector containing the aggregation rate elements, and

$\mathbf{J}(\mathbf{v}) = \left(\sum_{i=1}^n \omega_i \left(J_{1,ik} - J_{2,ik} \right) \right)_{3n\times 1}^T$ is the vector containing the fragmentation rate elements.

Applying the multivariate RK4 method (4.23) to the system of ODEs (4.24) results in the time evolution of the multimodal Gamma parameters and mass proportions that define the PSD changes throughout the flocculation process.

4.4.2 An Example of the Time Evolution of the General Moments' Equation with Bimodal Gamma Initial PSD

To illustrate the multivariate RK4 method presented in the previous section, the time evolution of the bimodal Gamma parameters and mass proportions will be obtained using an initial PSD calibrated with a given sediment sample (see Figure 4 - 1). The initial PSD was obtained using an X-Ray particle size analyzer that utilizes the sedimentation theory to obtain the mass percentages finer than a defined set of values (each value provides a data point of the entire PSD). The optimum bimodal initial Gamma parameters as well as the mass proportions were estimated using the least squares method. The time evolution of the flocculated PSD was then obtained by solving the system of ODEs (4.24) using the multivariate RK4 method explained in Section 4.4.1. Two different time periods are illustrated (t = 60 and t = 120) in Figure 4 - 2 where the effects are clearly evident of the flocculation that shifts the PSD curve to the coarser side as time increases.

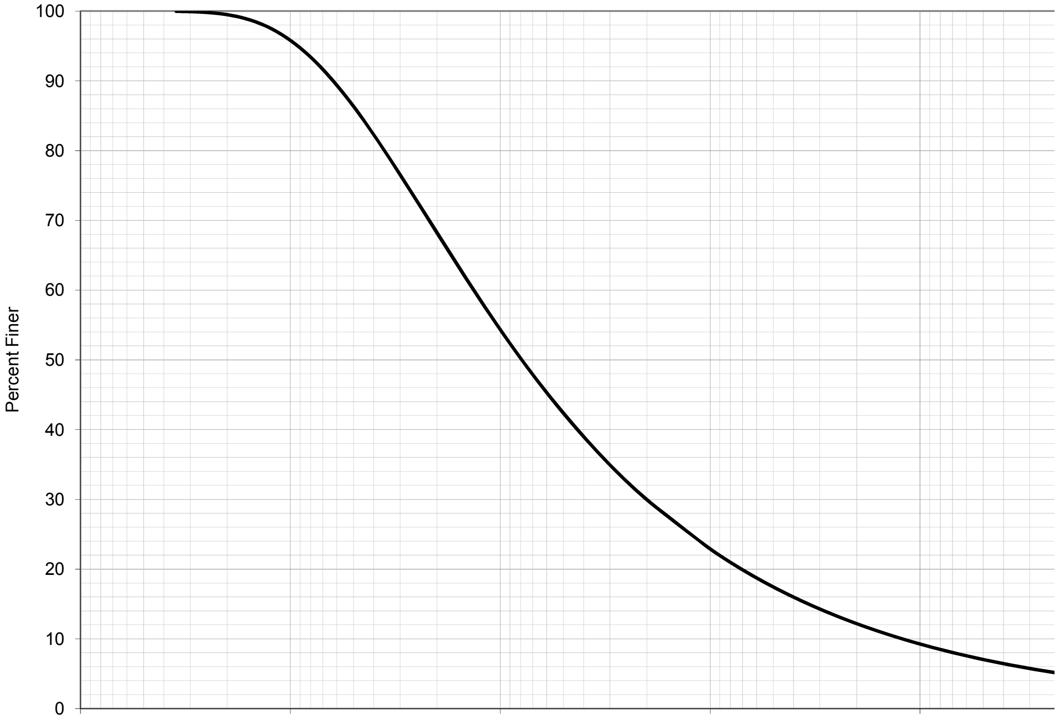

Figure 4 - 1: Initial PSD curve used to obtain the time evolution of flocculation

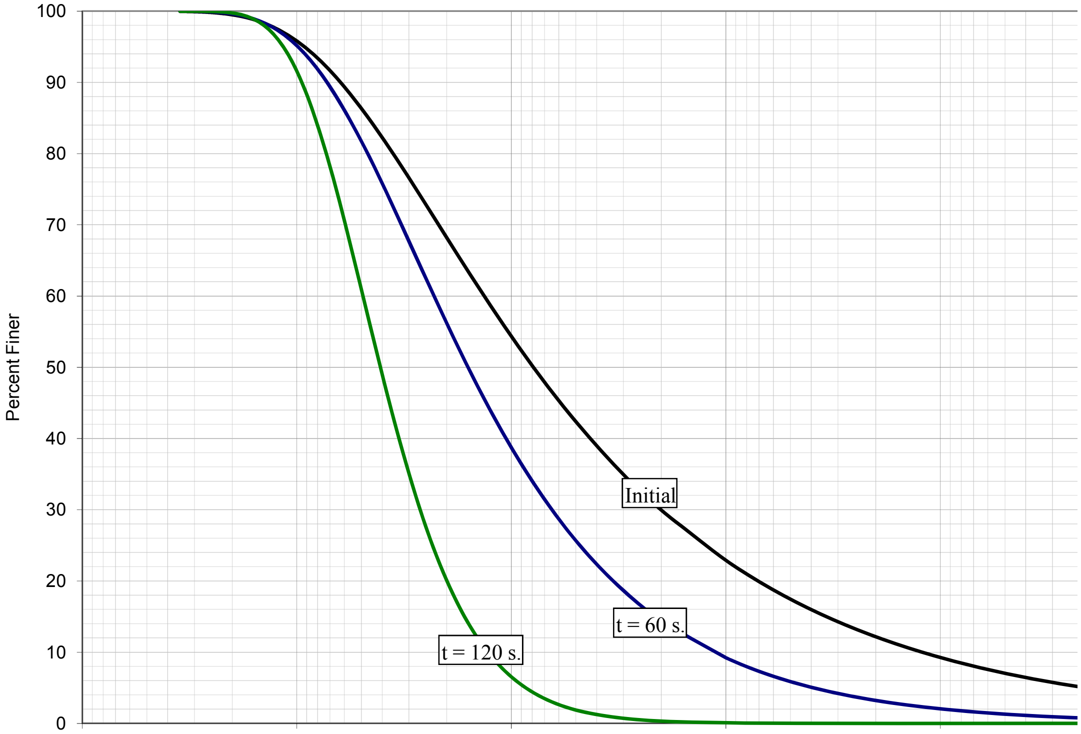

Figure 4 - 2: Example of the time evolution results of the PSD using a bimodal Gamma PSD

4.5 Model Calibration

The flocculation model presented in Section 4.3 was calibrated using experimental PSD data obtained from real sediment samples for the case of several different APAM flocculants. The model calibration was conducted using a simple genetic algorithm that enables finding the optimum aggregation and fragmentation parameters based on a set of assumptions that will be detailed further on.

4.5.1 Experimental Setup

Four different composite sediment samples, labeled A1 to A4 were chosen for the experimental analysis. These composite samples were assembled using sediment samples having particles smaller than 300 microns and gathered from different land uses and watersheds that drain to a sediment pond. Four APAM flocculants were obtained from different manufacturers labeled MF, FP, TF and HF. The optimum dosages of the flocculants for a sediment concentration of approximately 40,000 mg/L (required to run the particle size analyzer) were provided by the respective manufacturers. The PSD data for both the initial (control) and flocculated cases were obtained using a Micromeritics X-Ray particle size analyzer that uses the sedimentation technique to obtain the percent finer in mass of 130 particle diameters ranging from 0.2 to 300 microns. A total of 4 initial/control and 16 flocculated PSDs (4 for each initial sediment PSD) were obtained. In addition, the Micromeritics X-Ray particle size analyzer utilizes a propeller mixer producing a vortex that flocculates the sediment particles through turbulent shear forces. The rapid mixing apparatus was set for 120 seconds.

The bimodal Gamma parameters and mass proportions were calibrated using the least square method (Table 4 – 1). The average error, last column of Table 4 – 1, is calculated as:

$$\bar{\varepsilon} = \frac{1}{N}\sum_{i=1}^{N}\left|F_{i,\text{observed}} - F_{i,\text{predicted}}\right|, \tag{4.26}$$

where: N is the number of data points ($N = 130$),

$F_{i,\text{observed}}$ are the PSD data points obtained from the X-Ray particle size analyzer,

$F_{i,\text{predicted}}$ are the PSD data points modeled using a bimodal Gamma distribution obtained from (4.11)

Finally, the initial and flocculated PSD graphs are illustrated in Appendix H.

Table 4 - 1: Calibrated bimodal Gamma parameters and mass proportions for the PSD data

Soil	Flocculant	ω_1	α_1	β_1	ω_2	α_2	β_2	Average Error [a] (%)
A1	Control	0.600	2.718	9.594	0.400	0.273	6.233	0.150
	MF	0.701	3.910	7.160	0.299	0.533	16.378	0.710
	FP	0.723	3.584	7.953	0.277	0.140	24.384	0.613
	TF	0.736	5.684	5.848	0.264	1.366	8.526	0.120
	HF	0.878	5.113	7.314	0.122	3.048	2.480	0.102
A2	Control	0.266	1.239	1.680	0.734	0.343	9.280	0.067
	MF	0.542	14.257	1.881	0.458	2.084	12.730	0.518
	FP	0.894	13.768	1.578	0.106	0.119	58.414	0.320
	TF	0.716	12.845	1.822	0.284	1.348	14.736	0.264
	HF	0.880	14.690	1.561	0.120	1.245	17.003	0.254
A3	Control	0.148	1.478	1.680	0.852	0.369	9.280	1.307
	MF	0.587	13.828	2.227	0.413	2.106	13.230	0.356
	FP	0.853	11.146	2.458	0.147	0.166	43.257	0.628
	TF	0.759	13.095	2.102	0.241	1.178	21.457	0.253
	HF	0.760	14.013	1.948	0.240	1.403	18.001	0.271
A4	Control	0.452	2.496	10.886	0.548	0.467	4.366	0.244
	MF	0.557	10.727	2.766	0.443	1.567	13.491	0.402
	FP	0.614	2.966	8.236	0.386	0.274	28.329	0.597
	TF	0.809	5.328	5.982	0.191	1.970	4.439	0.389
	HF	0.870	6.348	4.840	0.130	3.729	1.476	0.176

Notes: [a] The average error is given by (4.26)

4.5.2 Calibration of the Model Parameters using a Simple Genetic Algorithm

The model parameters are divided in two groups: the aggregation and fragmentation parameters. The aggregation parameters consist of the kernel's parameters *A*, *r* and *p*, and the fragmentation parameters consist of the parameters B and λ included in the fragmentation kernel. Through a sensitivity analysis it was concluded that the parameters A and B are the most sensitive parameters for the aggregation and fragmentation processes, respectively. Therefore, in order to simplify the calibration process, only these two parameters were considered.

Additionally, it is necessary to assume certain values for the remaining parameters. The aggregation kernel (4.7) used in the present analysis shows a weak influence on the finite maximum particle diameter when the product given by rp is less or equal than 1 [4.12]. The analytical results described in Section 4.3 were based on an infinite maximum particle diameter (the integrals on the right-hand of the moments' equation are form 0 to $+\infty$); however the asymptotic behavior of the Gamma distribution makes this difference almost negligible: therefore the choice of $rp < 1$ is suitable for the present analysis. Moreover, when $p >$ 1, the difference of the aggregation rate with respect to the fragmentation counterpart is much larger, and unrealistic results start arising when (4.21) is used based on a non-asymptotic behavior. Setting the value of *p* to *p* = 1 forces setting the value of *r* < 1: for the present analysis, a value of 0.1 was deemed suitable. Finally, the parameter λ needs to satisfy the condition: $\lambda + \alpha_i - 1 > 0$, since from (4.17) the exponent κ needs to be positive (otherwise the distribution of the fragmented particles $P(x, y)$ could experience a non-asymptotic behavior): therefore, a suitable value for λ is set to $\lambda = 1$.

The system of ODEs that describe the rate of the PSD's moment variations over time can therefore be written as:

$$\dot{\mathbf{v}} = \mathbf{M}^{-1}\left(A \cdot \mathbf{I}'(\mathbf{v}) + B \cdot \mathbf{J}'(\mathbf{v})\right) \tag{4.27}$$

where **I'** and **J'** are the aggregation and fragmentation rate vectors, which elements have been normalized dividing by the parameters *A* and *B*, respectively.

To provide more flexibility for the calibration process and also better results, the parameters *A* and *B* are replaced by diagonal matrices **A** and **B**; respectively, having the dimension of the number of ODEs in the system, i.e. 3*n*. To calibrate all the values of the diagonal **A** and **B** matrices a simple genetic algorithm was used. The objective is to perform a least squares method on the observed flocculated PSD, i.e. to minimize the sum of squares error between the observed PSD data obtained from the experiments (see Section 4.5.1) and the predicted PSD values obtained through the flocculation model described in Section 4.3.

A genetic algorithm (GA) is a guided search algorithm used to solve non-linear optimization problems that was developed from genetic science. This analogy resides in the selection mechanism that nature implements where only the fittest and stronger species survive and thus they need to change and improve in order to endure. The main difference of this technique compared to other non-linear optimization methods is that it considers the use of a set of potential solutions instead of a single evaluation per iteration. This larger set of potential solutions enables the GA method to deviate away from a local minima or maxima and continue the search towards the global minimum/maximum if such a global optimum exists.

The set of potential solutions is called a population and each member of the population is denoted a chromosome. The selection process of the best sub-set of chromosomes within a population is conducted via evaluating a fitness function that provides a measure of error or deviation with respect to the objective the method is seeking. The fitness measure that was selected for the calibration process is the sum of square errors:

$$SEE = \sum_{i=1}^{m} \varepsilon_i^2 = \sum_{i=1}^{m} \left(F_{i,\text{observed}} - F_{i,\text{predicted}}\right)^2 \tag{4.28}$$

where: *N* is the number of data points in the PSD curve (*N* = 130),

$F_{i,\text{observed}}$ are the empirically obtained values of the flocculated PSD, and

$F_{i,\text{predicted}}$ are the predicted (modeled) values of the flocculated PSD.

The simple genetic algorithm is as follows:

1. A sub-set of the population is selected having the maximum value of the fitness measure (4.28),
2. The sub-set of the population that was not selected is eliminated and replaced with ‘improved’ chromosomes that have been ‘bred’ using the previously selected chromosomes.,
3. The breeding process of the new chromosomes is conducted through a sequence of mutation and cross-over techniques,
4. The mutation is a random modification of the elements of a chromosome and
5. The cross-over is the exchange within the elements of two ‘parent’ chromosomes to form a son (new chromosome).
6. Other improvement techniques include averaging between parents.
7. A new population is created (improved population) and the process continues until a stop criterion has been met.

The flow chart of this algorithm is illustrated in Figure 4 - 3. The goodness of fit between the observed flocculated PSD data points and the predicted values was conducted using the coefficient of determination (R^2), given by:

$$R^2 = \left\{ \frac{N\sum Y_o Y_p - \sum Y_o \sum Y_p}{\sqrt{N\sum Y_o^2 - \left(\sum Y_o\right)^2}\sqrt{N\sum Y_p^2 - \left(\sum Y_p\right)^2}} \right. \qquad (4.29)$$

where: N is the number of data points,

Y_o are the observed values, and

Y_p are the predicted values.

The R^2 values obtained from the genetic algorithm on the flocculated PSDs of samples A1 to A4 along with the calibrated (optimum) aggregation and fragmentation parameters (**A***i* and **B***i*) are presented in Tables 4- 2 to 4 - 5; respectively.

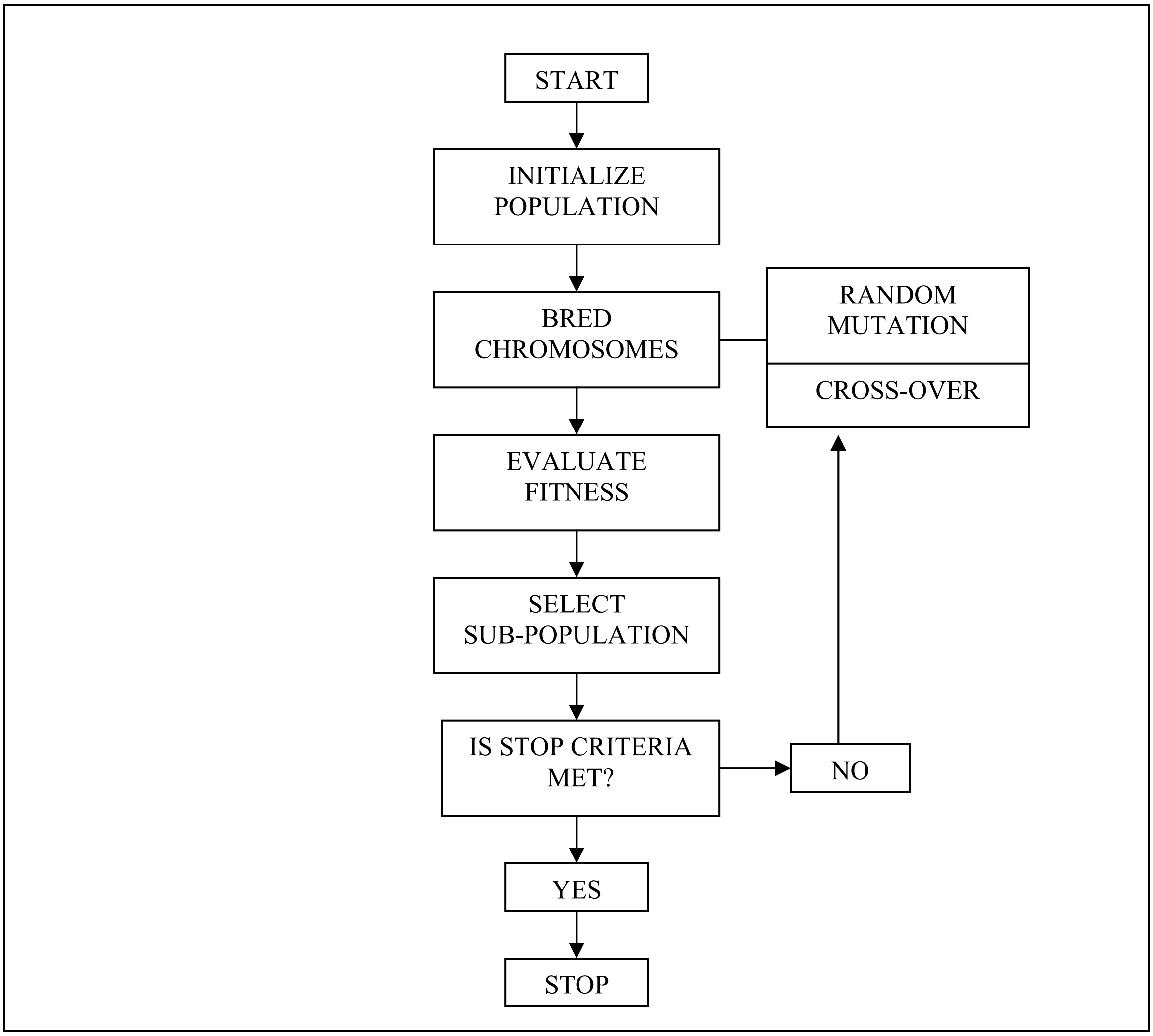

Figure 4 - 3: Flow chart of the genetic algorithm

The genetic algorithm was programmed using the C# computer language and the resulting code is included in Appendix I.

The predicted versus the observed flocculated PSD for the four different sediment samples (A1 to A4) using the four different flocculants are illustrated in Appendix J.

Table 4 - 2: Calibrated Aggregation and Fragmentation Parameters (**A** and **B**) for Sediment Sample A1

Flocculant		ω_1	α_1	β_1	ω_2	α_2	β_2	R^2
MF	A_i (x 10^{-10})	1.721	1.721	1.721	4.289	4.289	4.289	0.9973
	B_i (x 10^{-10})	1.623	1.623	1.623	4.044	4.044	4.044	
FP	A_i (x 10^{-9})	1.157	13.910	13.910	13.910	13.910	13.910	0.9955
	B_i (x 10^{-9})	115.578	115.578	115.578	115.578	115.578	115.578	
TF	A_i (x 10^{-10})	107.489	107.489	107.489	436.386	436.386	436.386	0.9969
	B_i (x 10^{-10})	3.132	3.132	3.132	12.715	12.715	12.715	
HF	A_i (x 10^{-11})	350.331	892.909	892.909	302.558	302.558	302.558	0.9878
	B_i (x 10^{-11})	26.016	26.016	26.016	8.816	8.816	8.816	

Table 4 - 3: Calibrated Aggregation and Fragmentation Parameters (A and B) for Sediment Sample A2

Flocculant		ω_1	α_1	β_1	ω_2	α_2	β_2	R^2
MF	A_i (x 10^{-9})	328.664	77.777	77.777	2.504	2.504	2.504	0.9976
	B_i (x 10^{-9})	69.993	69.993	69.993	2.253	2.253	2.253	
FP	A_i (x 10^{-9})	2295.921	79.070	79.070	3.221	3.221	3.221	0.9935
	B_i (x 10^{-9})	71.157	71.157	71.157	2.899	2.899	2.899	
TF	A_i (x 10^{-10})	247.977	9.727	9.727	7.221	722.127	722.127	0.9968
	B_i (x 10^{-10})	8.753	8.753	8.753	649.857	649.857	649.857	
HF	A_i (x 10^{-9})	5513.940	82.456	82.456	5.658	5.658	5.658	0.9952
	B_i (x 10^{-9})	74.204	74.204	74.204	5.091	5.091	5.091	

Table 4 - 4: Calibrated Aggregation and Fragmentation Parameters (A and B) for Sediment Sample A3

Flocculant		ω_1	α_1	β_1	ω_2	α_2	β_2	R^2
MF	Ai (x 10-8)	3.649	3.649	3.649	2.161	2.161	2.161	0.9992
	Bi (x 10-8)	3.284	3.284	3.284	1.945	1.945	1.945	
FP	Ai (x 10-9)	1.698	1.698	1.698	82.514	82.514	82.514	0.9956
	Bi (x 10-9)	1.528	1.528	1.528	74.256	74.256	74.256	
TF	Ai (x 10-9)	21.234	21.234	21.234	6.418	6.418	6.418	0.9968
	Bi (x 10-9)	19.109	19.109	19.109	5.776	5.776	5.776	
HF	Ai (x 10-9)	21.234	21.234	21.234	6.418	6.418	6.418	0.9962
	Bi (x 10-9)	19.109	19.109	19.109	5.776	5.776	5.776	

Table 4 - 5: Calibrated Aggregation and Fragmentation Parameters (A and B) for Sediment Sample A4

Flocculant		ω_1	α_1	β_1	ω_2	α_2	β_2	R^2
MF	A_i (x 10^{-10})	18.181	18.181	18.181	1.320	1.320	1.320	0.9966
	B_i (x 10^{-10})	16.361	16.361	16.361	1.188	1.188	1.188	
FP	A_i (x 10^{-9})	2.642	31.762	31.762	31.762	31.762	31.762	0.9958
	B_i (x 10^{-9})	263.909	263.909	263.909	263.909	263.909	263.909	
TF	A_i (x 10^{-10})	18.261	18.261	18.261	1.199	1.199	1.199	0.9985
	B_i (x 10^{-10})	16.434	16.434	16.434	1.079	1.079	1.079	
HF	A_i (x 10^{-11})	29.118	242.825	242.825	9.480	9.480	9.480	0.9967
	B_i (x 10^{-11})	218.523	218.523	218.523	8.532	8.532	8.532	

4.6 Conclusions

A new simple method to model the changes in PSD of fine sediment particles caused by aggregation and fragmentation processes when APAM flocculants are employed was presented. Expressing the initial PSD as a multimodal Gamma distribution enabled representing the aggregation and fragmentation general moments' equation as a system of ODEs that can easily be solved by a numerical method such as the Runge-Kutta method. The bimodal Gamma distribution was used to model the PSD of four different composite sediment samples without flocculation yielding excellent results (average error less than 1.4%). Four different APAM flocculants were used on each of the four different sediment samples and the resulting flocculated PSDs were compared with the calibrated PSDs obtained from the model. Calibration of the flocculated PSD data was conducted using a genetic algorithm. Results show that the average error for the flocculated PSDs were less than 1.7% for flocculants MF, TF and HF and less than 5.8% for flocculant FP. Interestingly, it can be inferred that this difference is due to the fact that FP flocculant failed to completely aggregate the fine clay fraction (< 2 microns).

From the results it can be inferred that the A_i and B_i values are identical for each mode except for a few tests and thus it may only be necessary to calibrate *n* pairs (A_i, B_i) instead of calibrating 3*n* pairs, where *n* is the number of modes. The values A_i represent the calibration parameters for the aggregation part of the flocculation model. Conversely, the B_i parameters account for the fragmentation contribution of the model. It is expected that flocculants presenting higher A_i with lower B_i values would perform better; however it is difficult to conclude this since the values of the other parameters have been assumed to simplify the problem and thus more work is needed in this regard.

Finally, the results show that the proposed approach can be used to accurately model the changes in PSD of fine sediment particles when flocculated using an APAM flocculant. The main advantage of this approach is that it does not induce high computational costs to numerically solve the continuous aggregation-fragmentation equation.

CHAPTER 5: Continuous One Dimensional Sedimentation Model of a Poly-dispersed Suspension of Particles having a Continuous Particle Size Distribution

Abstract

The Kynch kinematic theory of sedimentation can be extended to mathematically model the one-dimensional settling of a poly-disperse suspension of non-compressible particles in a liquid medium. This results in a system of hyperbolic conservation laws that require extensive computational time when a large number of particle sizes are considered (e.g. naturally occurring sediment particles). A continuous one dimensional sedimentation model that resulted in a scalar hyperbolic conservation law was derived from the well-known Kynch kinematic sedimentation model. This new continuous model tracks the changes in depth and time of the total concentration of suspended particles in a fluid settling in a one dimensional space where gravity is the only external force and where no wall effects are present. In addition, a continuous representation of the Batchelor, modified Batchelor and Richardson and Zaki settling velocity models were derived. The developed continuous one dimensional sedimentation model was calibrated using column tests results with glass micro-spheres particles to limit flocculation effects during sedimentation and compression near the bottom layer. Two different glass micro-spheres particle size distributions (PSDs) were used: a small and a large average size particle with average diameters of 7 and 37 microns; respectively. The Batchelor settling velocity

model was selected for calibration due to its simplicity and since it theoretically applies to diluted poly-dispersed suspensions such that exists in sedimentation ponds. Parameter estimation was achieved through the implementation of a genetic algorithm that employed a non-oscillatory, shock-absorbing numerical scheme to solve the continuous sedimentation model. Excellent values of coefficient of determination ($R^2 > 0.89$, except for one test replicate) were obtained for both the small and large glass micro-spheres PSDs.

5.1 Introduction

Mathematical modeling of the settling process of fine particles of different sizes (poly-dispersed suspension) inside a vertical column is of special interest in several diverse areas of science: separation of solids in waste water treatment plants, polymer science, thickening in mineralogy, sediment pond design in environmental engineering, etc. The well-known kinematic sedimentation model, based on the Kynch theory of sedimentation [5.6], is a discrete model that tracks the changes in concentration of a mono-dispersed (one single size particle) suspension in a one dimensional fluid with no wall effects. The extension of the mono-dispersed Kynch model to the poly-dispersed case result in a system of hyperbolic conservation laws [5.3]. The solution to this system of hyperbolic conservation laws requires the use of a numerical method since no analytical solution is available (the flux function is dependent on the concentration of particles). Furthermore, discontinuities also called shocks (from the collision of the characteristic lines) arise in the solution of such type of partial differential equations [5.4] and thus a numerical scheme that can capture these shocks is needed. There are many different numerical schemes that have been developed for this matter [5.8]; however, the most cost-effective (balancing computational efficiency with accuracy) numerical scheme should be selected. This efficiency consideration is especially critical when extending the sedimentation model to more than one dimension, which is the case for modeling sedimentation ponds. An alternative option to decrease the numerical method computational time would be to reduce the number of discrete particle sizes chosen from the PSD. However, reducing the number of particle sizes that are modeled may lead to significant accuracy problems. The proposed model enhances analysis efficiency by simplifying the system of conservation laws to a scalar problem by grouping all the particles into a single continuous model that can track the concentration changes of the total particle spectrum instead of a finite number of particles.

The findings presented in this Chapter can be considered as the first step in developing a theoretical sediment pond design tool. The proposed continuous model can easily be expanded to two or three dimensions at no extensive additional computational cost. Note however when applying an expanded two or three dimensional model to a sedimentation pond several other factor need to be considered, such as: turbulent zones, dead zones, resuspension and density currents.

5.2 Background

5.2.1 Kynch Model of Sedimentation (KMS) of a Poly-disperse Suspension

The KMS is obtained through the mass conservation of sediment particles settling through a control volume of fluid [5.11]. Assume that a set of solid spheres of *N* different size classes having the same density, are settling in a column filled with a fluid of known properties, Figure 5 - 1. The only external force applied to the sediment suspension is gravity and no wall effects are considered. Furthermore, no compression effects are present near the bottom layer and thus the particles are considered non-compressible. Let *v_i* be the settling velocity of spherical particles of size class (species) *i* and let ξ_i be the mass concentration of such particles as a function of time and depth, i.e. $\xi_i \equiv \xi_i(t,z), i = 1,\ldots N$, then the mass balances of the solid particles can be written as:

$$\begin{aligned} &\frac{\partial \xi_i}{\partial t} + \frac{\partial f_i}{\partial z} = 0, \quad i = 1,2,\ldots N, \\ &s.t. \quad \xi_i(0,z) = \xi_i^0(z), \quad 0 \le z \le L, \end{aligned} \tag{5.1}$$

where: $f_i = \xi_i \cdot v_i$ is the mass flux of particles through a depth *z* at time *t* and has units of mass per area per time,

z is the depth measured from the top, Figure 5 - 1, and

$\xi_i^0(z)$ is the initial mass concentration profile.

The system of partial differential equations (PDEs) described by (5.1) is the discrete representation of the kinematic model of sedimentation of a poly-dispersed suspension of non-compressible particles based on the Kynch theory of sedimentation. The KSM is totally defined by the settling velocity v_i, which depends on the type of settling the particles are experiencing.

5.2.2 Types of Settling

Inside a settling column a combination of different types of settlings occurs based on initial and changes in sediment concentration and depth approaching the bottom of the column. Based on the mass concentration of particles inside the column, four different types of settling can occur simultaneously or separately. The simplest is the discrete particle settling where particles settle independently of each other, i.e. no interaction between particles occurs. This type of settling only happens at extremely low concentrations. It is more probable to find this type of settling at the top portion of the column after considerable settling time has elapsed. The second type of settling is the transition settling where particle-to-particle interaction start occurring and where contact between particles could increase the settling velocity by creating flocs (aggregation) or reduce the settling velocity by repelling forces. As the concentration of particles further increases, particle-to-particle interaction increases and settling of particles becomes hindered. The hindered settling or third type of settling refers to settling in which concentration is high enough that interaction between particles interferes with the settling of the above particles. The final type of settling is the compression settling that occurs after the particles have been deposited at the bottom of the column. The weight of additional particles compresses the deposited sediment decreasing the voids therefore increasing their concentration.

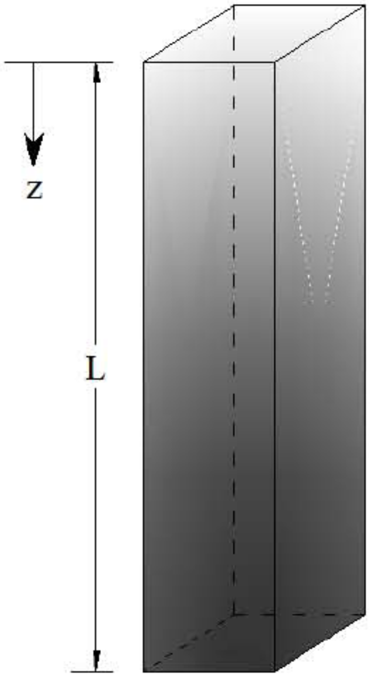

Figure 5 - 1: 1-D Sedimentation through a Vertical Column

In a sediment pond all four types of settling can occur. A comprehensive model would then take into consideration all of these types of settling; however the transition type is the most important inside a sediment pond since concentrations are large enough that discrete settling is rare to occur and small enough that hindered settling is unlikely to happen except when the bottom layer is approached.

5.2.3 Settling Velocity Models

The flux function associated to settling particles passing through a unit sectional area depends on the settling velocities. In a suspension of sediment particles it is expected that particles interacting with each other influences the settling velocity. As the total mass concentration increases it becomes more difficult for particles to settle since higher degree of particle-to-particle interaction takes place and thus smaller velocities are encountered. In general the settling velocity of a particle will be a function of the total solids mass concentration $\xi = \sum_{i=1}^{N} \xi_i$ of particles and the terminal velocity $v_{\infty,i}$:

$$v_i = v_{i,\infty} V(\xi), \tag{5.2}$$

where: v_i is the settling velocity of particle species *i*,

$v_{i,\infty}$ is the terminal settling velocity of particle species *i*, and

$V(\xi)$ is the settling function that depends on the total particle concentration, with $V(0) = 1$.

The terminal settling velocity represents the settling velocity of a single particle ($\xi = 0$) settling in an infinite fluid medium and is given by Stokes Law:

$$v_{i,\infty} = \frac{g \Delta \rho \, x_i^2}{18 \mu_f}, \tag{5.3}$$

where: g is the acceleration of gravity,

$\Delta\rho = \rho_s - \rho_f$, is the difference in density between the suspended solids (ρ_s) and the fluid (ρ_f),

μ_f is the fluid dynamic viscosity, and

x_i is the equivalent hydraulic diameter of particle specie *i*.

Batchelor Settling Velocity Model

Batchelor [5.1] derived a theoretical model for the settling velocity of a particle settling in a diluted polydisperse solution of *N* class species, which is given by:

$$v_i^B = v_{i,\infty}\left(1+\sum_{j=1}^{N} S_{ij}\xi_j\right) \tag{5.4}$$

where: v_i^B is the Batchelor settling velocity of particle specie *i*,

$v_{i,\infty}$ is the terminal settling velocity given by Stokes Law (5.3).

S_{ij} are sedimentation coefficients of particle species *i* and *j* that are defined as:

$$S_{ij} = S_{ij}^{(G)} + S_{ij}^{(I)} + S_{ij}^{(B)}, \quad i, j = 1 \ldots N \tag{5.5}$$

where $S_{ij}^{(G)}$ accounts for the contribution of gravity, $S_{ij}^{(I)}$ represents the particle-to-particle interaction and $S_{ij}^{(B)}$ accounts for the Brownian motion contribution. These three terms make use of the pair-distribution function $p_{ij}(r)$ that returns the probability that a particle of specie *j* is within a distance *r* of the particle of specie *i*.

Batchelor's settling velocity equation in theory applies to a dilute dispersion of particles where particle-to-particle interaction occurs; however, the range of particles mass concentration is not defined. Discrete settling can also be modeled using Batchelor's settling velocity equation: simply set the sedimentation coefficients to 0 such that the settling velocity equals the terminal velocity. The definition of the sedimentation coefficients used in the model does not consider the case where flocs are formed. Another constrain of the Batchelor's equation is that when the concentration reaches its maximum value ($\xi = \xi_{\max}$, usually at the bottom of the column) the settling velocity doesn't vanish. To this end Hofler and Schwarzer [5.5] modified Batchelor's equation resulting in the following expression:

$$v_{i,\infty}^{B\,\mathrm{mod}} = u_{i,\infty} \exp\left(\sum_{j=1}^{N} S_{ij} \xi_j + 2 \frac{\xi}{\xi_{\max}} \right) \left(1 - \frac{\xi}{\xi_{\max}} \right) \tag{5.6}$$

which vanishes for $\xi = \xi_{\max}$ and has the same partial derivatives for $\xi = 0$.

The main difficulty of the Batchelor model is to determine the coefficients S_{ij} for a specific polydisperse solution. These coefficients depend on the pair-distribution function that contains a high degree of solution complexity. Batchelor and Wen [5.2] successfully numerically solved the integrals that define the S_{ij} parameters. They concluded that the simple empirical relation:

$$S_{ij} = \frac{-2.5}{1+0.6\lambda} - \gamma\left(\lambda^2 + 3\lambda + 1\right) \tag{5.7}$$

where: $\lambda = x_j / x_i$ and $\gamma = (\rho_j - \rho_f)/(\rho_i - \rho_f)$ is accurate to the first decimal place. In general terms, the above equation can be written as:

$$S_{ij} = \sum_{k=0}^{2} c_k \left(\frac{x_j}{x_i}\right)^k, \tag{5.8}$$

where: c_k are coefficients that can be obtained directly from (5.7).

Richardson and Zaki Settling Velocity Model

The well-known Richardson and Zaki [5.10] formula for the settling velocity is given by the following expression:

$$v_i^{RZ} = v_{i,\infty}\left(1-\xi\right)^{\eta}, \eta \geq 1, \tag{5.9}$$

where: v_i^{RZ} is the Richardson and Zaki settling velocity of particle specie *i*,

$v_{i,\infty}$ is the terminal settling velocity given by Stokes Law (5.3), and

η is an exponent that depends on the physical properties of the problem.

If a particle is settling under conditions where the Stokes' Law is valid (Reynold's number < 0.2) the exponent η takes the value of 4.65. The Richardson and Zaki model was developed under hindered settling conditions where mass concentrations are very high and thus it can be expected that this formula does not provide accurate results when dealing with dilute or transitional sediment concentrations. Therefore, this model can only be applied to conditions where very high sediment concentrations are present, which occur near the bottom layer.

5.3 Continuous Representation of the KMS

The KMS presented in Section 5.2.1 only applies to a discrete representation of the PSD. Sediment particles entering a sediment pond usually embody a wide spectrum of size species ranging from fine clays to coarse sands. There is no analytical solution to the KMS when using the settling velocity models described previously and thus a numerical method is needed to obtain an approximate solution. The accuracy of this approximate solution will depend on the number of particle size species chosen from the entire PSD spectrum. The higher the number of particle size species the more accurate the solution becomes; however, the more computational time is required. In other words, the selection of the number of particle species from the PSD to be modeled needs to be large enough to obtain a desired degree of accuracy yet small enough to have an acceptable computational efficiency. The appropriate number of particle species to achieve this balance is currently unknown. Another approach taken is this research, that resolves this problem, was to develop a continuous representation of the KMS that is derived from the discrete form simplifying the sedimentation model to a scalar hyperbolic conservation law.

If the discrete size of particle species i is replaced by the continuous size variable x then a continuous representation of the KMS can be obtained. Writing $\xi_x = \xi(t, x, z)$ the mass concentration of a sediment particle of size x at time t and depth z then (5.1) becomes:

$$\begin{aligned}&\frac{\partial \xi_x}{\partial t} + \frac{\partial f(\xi_x)}{\partial z}, \quad x \in [0, +\infty) \\ &s.t. \quad \xi_x(0,z) = \xi_x^0(z), \quad 0 \le z \le L.\end{aligned} \tag{5.10}$$

The PSD of sediment particles can be modeled using a continuous distribution function $p(x)$, such as for example a Gamma probability distribution (Chapter 2), and thus for the multimodal case:

$$p(x) = \sum_{i=1}^{n} \omega_i p_i(x), \tag{5.11}$$

where: n is the number of modes,

ω_i is the mass proportion of mode i, and

$p_i(x)$ is the continuous (unimodal) distribution function used to model the PSD.

In addition, the concentration of the particle size x can be defined by the following expression:

$$\xi_x = \xi \cdot p(x), \tag{5.12}$$

where: $\xi = \int_0^\infty \xi_x dx$ is the total mass concentration at time t and at depth z, and $p(x)$ is the multimodal PSD function given by (3.10). If the Gamma probability distribution is used then:

$$p(x) = \sum_{i=1}^{n} \omega_i \frac{x^{\alpha_i - 1} e^{-x/\beta_i}}{\beta_i^{\alpha_i} \Gamma(\alpha_i)}, \tag{5.13}$$

where: α_i is the shape parameter of the Gamma distribution for mode i, and

β_i is the scale parameter of the Gamma distribution for mode i.

To extend the discrete results to a continuous model of a well-mixed suspension when considering fine sediment particles settling under gravitational forces, a settling velocity function is required.

5.3.1 Continuous Representation of the Settling Velocity Models

There is no known continuous form of the settling velocity presented in Section 5.2.3. Continuing with the derivation of the CKSM introduced in the previous section, a continuous mathematical representation of the settling velocity is needed. This continuous representation will be developed for the three different settling velocity models presented in Section 5.2.3: Batchelor, modified Batchelor and Richardson and Zaki.

If the equivalent hydraulic diameter of a particle is represented by the continuous variable x then the following equivalent continuous expression for the settling velocity is obtained:

$$v(\xi, x) = v_\infty(x) V(\xi)\,, \tag{5.14}$$

where: the settling function $V(\xi)$ can also be a function of the particle size *x*.

Continuous Representation of the Batchelor Settling Velocity Model

Let $S(x, y)$ be Batchelor's sedimentation coefficients represented by a continuous function that accounts for interactions between particles of size *x* and particles of size *y*, then (5.4) results in:

$$v_x^B \equiv v^B(\xi, x) = u_\infty(x)\left(1 + \int_0^\infty S(x, y)\xi_y dy\right). \tag{5.15}$$

Using the polynomial approximation of Batchelor's settling function (5.8), i.e.

$$S(x, y) = \sum_{k=0}^{m} c_k \left(\frac{y}{x}\right)^k, \quad \text{with } m \geq 2, \tag{5.16}$$

and using (5.3), (5.11) and (5.12) into (5.15) yields:

$$v_x^B = \mu_\infty x^2 \left(1 + \xi \int_0^\infty \sum_{k=0}^{m} c_k \left(\frac{y}{x}\right)^k \cdot \sum_{i=1}^{n} \omega_i p_i(y) dy \right. \tag{5.17}$$

where $\mu_\infty = \dfrac{g\Delta\rho}{18\mu_f}$. Regrouping terms in the right-hand side of the above equation results in:

$$v_x^B = \mu_\infty \left(x^2 + \xi \sum_{k=0}^{m} c_k x^{2-k} \sum_{i=1}^{n} \int_0^\infty \omega_i y^k p_i(y) dy \right. \tag{5.18}$$

The k^{th} moment of the multimodal PSD function is given by:

$$M_k = \int_0^\infty y^k p(y) dy = \sum_{i=1}^{n} \int_0^\infty \omega_i y^k p_i(y) dy, \tag{5.19}$$

and thus the continuous representation of the Batchelor's settling velocity model is finally given by:

$$v_x^B = \mu_\infty \left(x^2 + \xi \sum_{k=0}^{m} c_k x^{2-k} M_k \right). \tag{5.20}$$

Continuous Representation of the Modified Batchelor Settling Velocity Model

Following along the lines of the previous analysis, the modified continuous Batchelor velocity results in:

$$v_x^{B\,\mathrm{mod}} = \mu_\infty x^2 \exp\left(\xi\sum_{k=0}^{\infty} c_k x^{-k} M_k + 2\frac{\xi}{\xi_{\max}}\right)\left(1-\frac{\xi}{\xi_{\max}}\right) \tag{5.21}$$

Continuous Representation of the Richardson and Zaki Settling Velocity

Similarly, by replacing the Stokes terminal velocity by its continuous form, the Richardson and Zaki continuous representation of the settling velocity is obtained:

$$v_x^{RZ} \equiv v^{RZ}(\xi, x) = \mu_\infty x^2 (1-\xi)^\eta . \tag{5.22}$$

5.3.2 Continuous Kinematic Sedimentation Model (CKSM)

In the effort of trying to use the discrete KMS to model the sedimentation of fine sediment particles it is necessary to divide the PSD curve in intervals and for each interval an average particle size is chosen. The degree of accuracy of the model will depend on the number of intervals selected. Let Δx_i, i = 1…N, be the interval around the particle size x_i then (5.1) can be written as:

$$\begin{aligned}&\sum_{i=1}^{N}\left(\frac{\partial \xi_i}{\partial t}+\frac{\partial f_i}{\partial z}\right)\Delta x_i = 0\\&\Rightarrow \frac{\partial}{\partial t}\left(\sum_{i=1}^{N}\xi_i \Delta x_i\right)+\frac{\partial}{\partial z}\left(\sum_{i=1}^{N} f_i \Delta x_i\right)=0.\end{aligned} \tag{5.23}$$

Taking the limit $N \to \infty$ ($\Rightarrow \Delta x_i \to 0$) yields:

$$\frac{\partial}{\partial t}\int_0^{\infty}\xi_x dx + \frac{\partial}{\partial z}\int_0^{\infty} f(\xi_x)dx = 0, \tag{5.24}$$

and defining $F(\xi) \equiv \int_0^{\infty} f(\xi_x)dx$ as the *total flux function*, the CKMS is finally obtained:

$$\begin{aligned}&\frac{\partial \xi}{\partial t}+\frac{\partial F(\xi)}{\partial z}=0\\&s.t. \quad \xi(0,z)=\xi^0(z)\\&\qquad 0 \le \xi \le \xi_{\max},\end{aligned} \tag{5.25}$$

where: $\xi^0(z)$ is the initial total concentration profile and $\xi_{\max}$ is the maximum possible mass concentration inside the settling column. Next the expressions for $F(\xi)$ will be provided for each of the different settling velocity models described previously.

Total Flux Function using the Batchelor's Settling Velocity Model

The continuous Batchelor settling velocity was derived in the previous section and is given by (5.20). Using (5.12) and (5.20) the total flux function then becomes:

$$\begin{aligned} F^{B}(\xi) &= \int_0^\infty \xi_x v_x^B \, dx \\ &= \xi\mu_\infty \int_0^\infty p(x)\Bigg(x^2 + \xi\sum_{k=0}^{m} c_k x^{2-k} M_k \Bigg) \end{aligned} \tag{5.26}$$

By further developing the right-hand side of the above equation the following result is obtained:

$$\begin{aligned} F^{B}(\xi) &= \xi\mu_\infty \Bigg(\int_0^\infty x^2 p(x)dx + \xi\sum_{k=0}^{m} c_k M_k \int_0^\infty x^{2-k} p(x)d \\ &= \xi\mu_\infty \Bigg(M_2 + \xi\sum_{k=0}^{m} c_k M_k M_{2-k} \Bigg). \end{aligned} \tag{5.27}$$

It is easy to infer from the above result that the maximum possible value that the index *m* can take is 2, which is the same value for the discrete case (see (5.8)). By replacing the value of *m* = 2 in the above equation, the Batchelor total flux equation is obtained:

$$F^{B}(\xi) = \xi\mu_\infty [M_2 + \xi(c_0 M_2 + c_1 M_1^2 + c_2 M_2 \tag{5.28}$$

Finally, replacing $d_0 = c_1$ and $d_1 = c_0 + c_2$ results in:

$$F^{B}(\xi) = \xi\mu_{\infty}\left[M_2 + \xi\left(d_0 M_1^2 + d_1 M_2\right)\right]. \tag{5.29}$$

Total Flux Function using the Modified Batchelor's Settling Velocity Model

In the case of the modified Batchelor model, the continuous settling velocity is given by (5.21) and thus multiplying by ξ_x and integrating over the entire size spectrum results in:

$$F^{B\,\mathrm{mod}}(\xi) = \xi\mu_{\infty}\exp\left(2\frac{\xi}{\xi_{\max}}\right)\left(1-\frac{\xi}{\xi_{\max}}\right)^{2}\int_{0}^{\infty} p(x)\exp\left(\xi\sum_{k=0}^{m} c_k x^{-k} M_k\right)c \tag{5.30}$$

.

Expanding the right-hand side of the above equation by replacing the exponential function by its series expansion ($\exp(z) = \sum_{k=0}^{\infty}\frac{z^k}{k!}$) results in:

$$\int_{0}^{\infty} p(x)\exp\left(\xi\sum_{k=0}^{2} c_k x^{-k} M_k\right)dx = \exp\left(c_0\xi\right)\sum_{j=0}^{\infty}\sum_{k=0}^{\infty}\frac{c_1^j M_1^j}{j!}\frac{c_2^k M_2^k}{k!}M_{2(1-k)} \tag{5.31}$$

The above equation is valid only when $2(1-k)-j \geq 0$; which results in the set of values defined by k = 0 and $j = \{0,1,2\}$. Finally, the final total flux function for the modified Batchelor case results in:

$$F^{B\,\mathrm{mod}}(\xi) = \xi\mu_{\infty} \exp\left(2\frac{\xi}{\xi_{\max}}\right)\left(1-\frac{\xi}{\xi_{\max}}\right)^{2} \exp(\xi)\left(d_0 M_1^{2} + d_1 M \right. \tag{5.32}$$

<u>Richardson and Zaki</u>

The continuous expression of the settling velocity for the Richardson and Zaki model is given by (5.22). Multiplying the settling velocity by ξ_x and integrating within the entire possible set of particle sizes *x* results in the total flux function for the Richardson and Zaki model and is given by:

$$F^{RZ}(\xi) = \xi\mu_{\infty}(1-\xi)^{\eta} M_2 \tag{5.33}$$

5.3.3 Flocculation

During the settling process inside the column particles can be attracted and aggregate to form flocs. When this happens the settling velocity increases, which also increases the total flux function. To consider this effect the PSD need to be defined as a function of time. In Chapter 3 a new continuous aggregation and fragmentation model was derived that tracks the changes of the PSD over time due to flocculation when a multimodal Gamma distribution is used to model the particles' initial PSD. The aggregation and fragmentation model is given by:

$$\begin{aligned}\sum_{i=i}^{n}\Bigg[&\beta_i^{k-1}\prod_{j=2}^{k}(k-j+\alpha_i)\frac{d\omega_i}{dt}+\omega_i\beta_i^{k-1}\sum_{q=1}^{k}\prod_{\substack{j\neq q\\ j\geq 2}}^{k}(k-j+\alpha_i)\frac{d\alpha_i}{dt}+\\ &+k\omega_i\beta_i^{k-2}\prod_{j=1}^{k}(k-j+\alpha_i)\frac{d\beta_i}{dt}\Bigg]=\sum_{i}^{n}\sum_{j}^{n}\omega_i\omega_j\left(I_{1,ijk}-I_{2,ijk}\right)+\sum_{i=1}^{n}\omega_i\left(J_{1,ik}-J_{2,ik}\right),\end{aligned}\tag{5.34}$$

where the multimodal Gamma distribution representing the PSD function is given by (5.13). The aggregation terms are given by:

$$\begin{aligned}I_{1,ijk}=\frac{A}{2\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\beta_i^{k-l+r(p-m)}\beta_j^{l+rm}\Gamma(k-l+r(p-m)+\alpha_i)\\ \cdot\Gamma(l+rm+\alpha_j),\end{aligned}\tag{5.35}$$

and

$$I_{2,ijk}=\frac{A}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{m=0}^{p}\binom{p}{m}\beta_i^{k+r(p-m)}\beta_j^{rm}\Gamma(k+r(p-m)+\alpha_i)\Gamma(rm+\alpha\tag{5.36}$$

where: A is a constant that depend on the aggregation conditions. The fragmentation terms are given by:

$$J_{1,ik} = \frac{Bb^{k}\Gamma(k+a-1)}{\Gamma(a)}\sum_{i=1}^{n}\omega_i \frac{\beta_i^{\lambda+k+a}}{(\beta_i+b)^{k+a}\Gamma(\alpha_i)}\Gamma(\lambda+\alpha_i - \qquad (5.37)$$

and

$$J_{2,ik} = B\sum_{i=}^{n}\frac{\beta_i^{k+\lambda}}{\Gamma(\alpha_i)}\Gamma(k+\lambda+\alpha_i). \qquad (5.38)$$

where: B is a constant that depend on the fragmentation conditions.

Combining the scalar conservation law given by (5.25) with one of the total flux functions, previously derived with the above aggregation-fragmentation equations, yields the continuous kinematic sedimentation model that considers flocculation. Since no analytical solution exist for this type of conservation law a numerical method needs to be used. For each time step in the numerical method, the multimodal Gamma parameters that define the PSD of the sediment particles needs to be updated using the aggregation-fragmentation equations presented above. With the updated parameters the new moments M_1 and M_2 are obtained for the total flux function.

If flocculation occurs before the settling process commences (e.g. when flocculation is induced through a chemical flocculant prior to entering the sediment pond) the aggregation-fragmentation model described previously need to be implemented before solving the conservation law. The only difference with the non-flocculated case is that the initial PSD has been updated prior to the sedimentation process to account for the induced flocculation.

5.4 Settling Column Tests

5.4.1 Description

A column test uses a vertical settling column as a surrogate to mimic sediment-settling processes within a sediment pond. The settling column used in the experimental design consists of a vertical cylindrical PVC pipe fitted with six sampling ports located at 30 cm intervals along the column.

The column was filled with various concentrations of particles in water where pH and temperature were controlled. After mixing and achieving a relatively homogenous suspension, the sedimentation process begins. At pre-designated time intervals samples were withdrawn from each port and subsequently tested for total solids concentration. The concentration profiles along the column were then obtained as a function of time and initial sediment concentration. Figure 5 - 2 shows a sketch of the type of column used in the experiments.

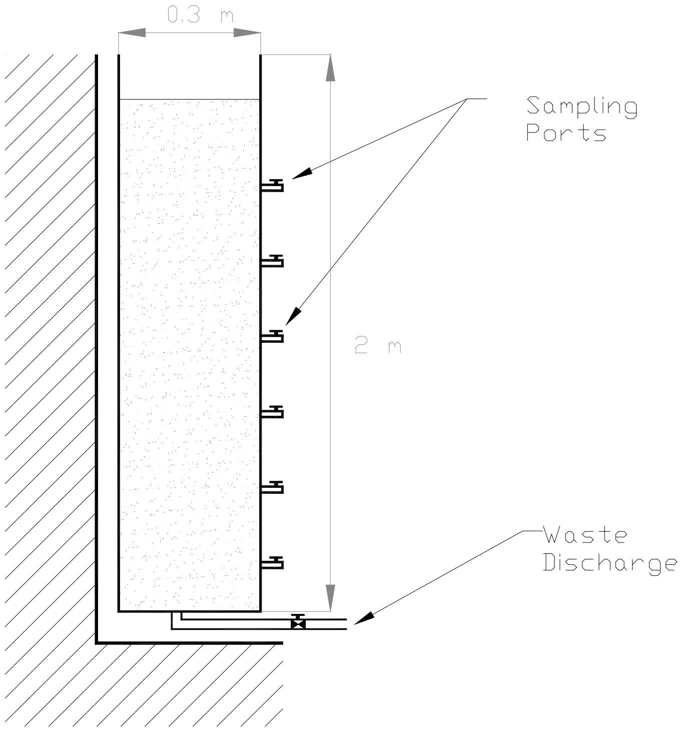

Figure 5 - 2: Cross Section of Sedimentation Column

Three cylindrical columns with 6 sampling ports each were fabricated for this research. Column dimensions are 2 m in height and an internal diameter of 25 cm. A drainage line, controlled by a valve, was inserted at the bottom of each column to drain and clean the columns after completing each experiment. The columns are equipped with solenoid valves located at each port that open at the touch of a button. The use of the solenoid valves provides withdrawal of consistent sample volumes since the entire set of samples (6) are initiated at the same time with slight differences in sampling durations based on the head of water above the respective valves.

5.4.2 Particle Size Distributions

The sediment samples were obtained using glass micro-spheres to limit flocculation effects during the sedimentation process and to avoid compression effects at the bottom layer (non-compressible particles). Two PSDs of glass micro-spheres were used having average diameters of 7 μm (small) and 35 μm (large) and their PSD curves are listed in Chapter 2, Section 2.3. The PSD of the small and large glass micro-spheres were obtained using an X-Ray particle size analyzer that used the sedimentation theory (Stokes' Law).

A bimodal Gamma distribution was shown in Chapter 2 to provide the most accurate representation and therefore was used to fit the PSD curves of the small and large distributions. The obtained Bimodal Gamma parameters are presented in Table 5 - 1.

Table 5 - 1: Bimodal Gamma Parameters, fist and second moments of the two Glass Micro-spheres' PSDs.

Glass Micro-Sphere	ω_1	ω_2	α_1	α_2	β_1	β_2	M_1 [(a)]	M_2 [(a)]
Small	0.028	0.972	0.429	5.102	62.987	1.235	6.878	113.922
Large	0.046	0.954	0.316	8.685	326.122	3.930	37.280	3260.523

Notes: [(a)] M_1 and M_2 are obtained analytically from (5.19).

For each tested PSD three different total solids concentrations were used: small, medium and high concentrations with target values of 5,000 mg/L, 15,000 mg/L and

45,000 mg/L; respectively. The mass of glass micro-spheres that needed to be introduced in the columns for each test were derived from the targeted total solids concentrations since the initial volume of water was known. Three replicates of each column test were conducted giving a total of (2 PSDs x 3 concentrations x 3 replicates) 18 column tests.

5.4.3 Procedure

Calculated amounts of glass micro-spheres and tap water were placed inside the column to obtain the desired initial concentrations. The solution was then thoroughly mixed using compressed air delivered at the bottom of the column through dispersal heads to distribute the air flow throughout the entire column height and cross-sectional area. The mixing of the sediment slurry was conducted for 5 minutes and the first set of samples was taken while the mixing apparatus continued to operate. These samples represent the initial concentration profile. After initial sampling was completed mixing stops and the sedimentation process begins. The clock is set to 0 and sampling at each port was conducted at 5, 10, 15, 20, 30, 45, 60 and 90 minutes. Sampling occurred concurrently from all ports with an average sampling time of 2.0 seconds. The initial and final water heights were recorded for each sampling time interval.

The initial column water height (L) was set at 1.75 m to avoid any spillage during the mixing process. Ph was set at a neutral level (pH between 6.5 and 7.5) and the water temperature was set at 20 ± 2 °C. Every time a set of samples was taken the total depth decreases and the relative depth between ports is adjusted accordingly. Consequently an overall correction was made to each depth such that the final calculated depth matched the final recorded depth. Finally the total solids concentration was obtained for each sample and the concentration profile along the column was determined for each sampling time.

5.5 Model Calibration

The CKSM was calibrated considering a time frame that is representative of the retention time of a typical sediment pond. A time interval of 60 minutes was selected to conduct the calibration. It is important to note however that if induced flocculation had been applied to the sediment-laden flow entering the sediment pond then smaller settling times are applicable and thus calibration should be conducted for time intervals of approximately 10 to 30 minutes or smaller. Note that this case only applies when flocculation has been induced prior to sedimentation and that the sediment pond's retention time is small.

Typically, in a sediment pond sediment concentrations vary in the range where particle-to-particle interactions (transitional settling) are expected and thus the Batchelor settling velocity model is more applicable than the Richardson and Zaki model. This is especially true where effluent withdrawals occur throughout a vertical portion of the spillway such as when utilizing a perforated riser or when withdrawals occur from a single elevation such as a fixed siphon. No hindered settling is modeled in the Batchelor's model. Furthermore, for simplification purposes the non-modified Batchelor's model was used for calibration. The parameters that need to be estimated are therefore the d_0 and d_1 sedimentation coefficients.

The CKSM described by (5.25) in combination with the total flux function (5.29) derived from the Batchelor's settling velocity model needs to be solved using a numerical method. Based on the nature of hyperbolic conservation laws, discontinuities in the solution (shocks) are expected to occur and therefore a *shock-absorbing* numerical scheme is required. In addition, oscillations may also arise with *high-viscosity* schemes and thus it also needs to be treated accordingly. The selection of the numerical method should rely on a combination of accuracy and computational cost while addressing the shock-absorbing and non-oscillatory conditions. The second order central scheme developed by Nessyahu and Tadmor [5.9] has been chosen as the numerical method to calibrate the CKSM mainly due to its operational efficiency. Moreover, the non-staggered grid version presented in [5.6] was used. Note that the calibration process being an optimization based problem requires a thorough iteration process to find an accurate solution that may take extensive amount of time and thus a less computational expensive numerical method is desired. Finally, the CKSM is a scalar hyperbolic conservation law and thus the scalar version of the numerical method will be presented next.

5.5.1 High-Resolution Non-Oscillatory Central Scheme with Non-Staggered Grid

A brief description of the numerical method that was used to calibrate the model will be presented in this section. For further details refer to [5.9] and [5.6].

The *z-t* space and time plane is represented by discrete pairs labeled (z_j, t^n), for *j* = 0….*J* and *t* = 0…*T*. The space and time finite differences Δz and Δt; respectively, are defined such that $j\Delta z = z_j$ and $n\Delta t = t^n$. The boundary conditions are defined by: $J\Delta z = L$ and $T\Delta t = t_{\max}$.

The purpose of using a numerical method is to find an approximate solution of the total solids concentration function $\xi(z,t)$ at the grid points (z_j, t^n). This approximate solution is depicted by $\zeta_j^n \approx \xi(z_j, t^n)$. The objective of the central scheme is to find an approximate solution of the average of the function at the associated cell or *z* interval of the grid at a given time. The central scheme approach limits the formation of shocks or discontinuities that usually arise with these types of partial differential equations.

The Nessyahu and Tadmor numerical scheme (NT) is based on the well celebrated Lax-Friedrich 1st order scheme (LxF) that is given by:

$$\overline{\zeta}_{j+\frac{1}{2}}^{n+1} = \frac{1}{2}\left(\overline{\zeta}_j^n + \overline{\zeta}_{j+1}^n\right) - \lambda\left[F\left(\zeta_{j+1}^n\right) - F\left(\zeta_j^n\right)\right] \tag{5.39}$$

where $\overline{\zeta}_j^n$ is the approximate average solution of the associated cell (interval) $I_j = \left\{\left|z - z_j\right| \le \frac{\Delta z}{2}\right\}$ at the time step $t^n = n\Delta t$, and $\lambda = \frac{\Delta t}{\Delta z}$. The LxF besides being a 1st-order numerical method, it has the inconvenient that it suffers from excessive numerical viscosity and thus high level of oscillations arises at the neighborhood of the discontinuities. This is due mainly by the discrete representation of the flux derivative in

(5.39). To solve this problem Nessyahu and Tadmor suggested the use of the *MinMod* delimiter to numerically calculate the derivatives. The MinMod (*MM*) delimiter is given by:

$$MM\{x_1, x_2, \ldots\} = \begin{cases} \min_j \{x_j\} & \text{if } x_j > 0 \ \forall \\ \max_j \{x_j\} & \text{if } x_j < 0 \ \forall \\ 0 & \text{otherwise.} \end{cases} \tag{5.40}$$

The non-oscillatory numerical derivatives are given by:

$$\begin{gathered} \zeta'_j = MM\left(\theta \Delta \zeta_{j+\frac{1}{2}}, \frac{1}{2}\left(\Delta \zeta_{j-\frac{1}{2}} + \Delta \zeta_{j+\frac{1}{2}} \right), \theta \Delta \zeta_{j-\frac{1}{2}} \right), \quad 1 \le \theta \le \\ \Delta \zeta_{j+\frac{1}{2}} = \zeta_{j+1} - \zeta_j, \quad \Delta \zeta_{j-\frac{1}{2}} = \zeta_j - \zeta_{j-1}. \end{gathered} \tag{5.41}$$

The first step in the method is to calculate a predictor step in the time evolution process,

$$\zeta_j^{n+\frac{1}{2}} = \zeta_j^n - \frac{\lambda}{2} F'_j , \tag{5.42}$$

with

$$F'_j = \frac{\partial F(\overline{\zeta}^n_j)}{\partial z} = \frac{\partial F(\overline{\zeta}^n_j)}{\partial \xi} \zeta'_j , \tag{5.43}$$

where $\frac{\partial F}{\partial \xi}$ is obtained analytically and ζ'_j is given by (5.41).

Finally, the NT scheme is given by the corrector step:

$$\overline{\zeta}^{n+1}_{j+\frac{1}{2}} = \frac{1}{2}\left(\overline{\zeta}^{n}_{j} + \overline{\zeta}^{n}_{j+1}\right) + \frac{1}{8}\left(\zeta'_{j} - \zeta'_{j+1}\right) - \lambda\left[F\left(\zeta^{n+\frac{1}{2}}_{j+\mathbf{1}}\right) - F\left(\zeta^{n+\frac{1}{2}}_{j}\right.\right. \tag{5.44}$$

The non-staggered scheme is obtained by first calculating the staggered cell-averages at time t^{n+1} using a piece-wise linear interpolant (to maintain the second-order accuracy) and second, the non-staggered cell-averages are obtained by averaging the linear interpolant at the cell-averages for the next time step. The resulting non-staggered NT scheme is given by:

$$\begin{aligned}\overline{\zeta}^{n+1}_{j} &= \frac{1}{4}\left(\overline{\zeta}^{n}_{j-1} + 2\overline{\zeta}^{n}_{j} + \overline{\zeta}^{n}_{j+1}\right) - \frac{1}{16}\left(\zeta'_{j+1} - \zeta'_{j-1}\right) \\ &\quad - \frac{\lambda}{2}\left[F\left(\zeta^{n+\frac{1}{2}}_{\mathbf{j+1}}\right) - F\left(\zeta^{n+\frac{1}{2}}_{\mathbf{j}-1}\right)\right] - \frac{1}{8}\left(\zeta'_{j+\frac{1}{2}} - \zeta'_{j-\frac{1}{2}}\right),\end{aligned} \tag{5.45}$$

where:

$$\zeta'_{j-\frac{1}{2}} = MM\left(\Delta\overline{\zeta}^{n+1}_{j}, \Delta\overline{\zeta}^{n+1}_{j-1}\right), \quad \zeta'_{j+\frac{1}{2}} = MM\left(\Delta\overline{\zeta}^{n+1}_{j+1}, \Delta\overline{\zeta}^{n+1}_{j}\right), \tag{5.46}$$

and

$$
\begin{aligned}
\Delta\overline{\zeta}_{i}^{\,n+1} &= \frac{1}{2}\left(\overline{\zeta}_{i+1}^{\,n}-\overline{\zeta}_{i-1}^{\,n}\right)-\frac{1}{8}\left(\zeta'_{i-1}-2\zeta'_{i}+\zeta'_{i+1}\right)\\
&\quad-\lambda\left[F\left(\zeta_{i-1}^{n+\frac{1}{2}}\right)-2F\left(\zeta_{i}^{n+\frac{1}{2}}\right)+F\left(\zeta_{i+1}^{n+\frac{1}{2}}\right)\right],\quad i=j-1,j,j+1.
\end{aligned}
\tag{5.47}
$$

The code for the above numerical scheme was written in C# programming language and is presented in Appendix M.

5.5.2 Initial Condition

The CKSM is an initial value problem and thus the accuracy of its results is highly dependent on the accuracy of the mathematical representation (interpolation) of the initial concentration profile (at time t = 0). The column tests provide 6 observed points for the initial concentration profile since the column only has 6 ports to withdraw samples. Evidently that the grid used in the numerical method described in Section 5.5.1 requires a larger amount of points along the z-axis (i.e. J > 6) to obtain accurate results.

To populate the grid with an accurate initial concentration profile $\zeta(z,t=0)$ it is necessary to interpolate between the six different initial condition points obtained from the columns tests. A simple piece-wise linear interpolation between the observed points is not adequate since discontinuities are induced directly at each of the six initial condition points that could cause instability in the solution when a coarse grid configuration is used. To avoid potential oscillations while keeping the number of grid points small, a cubic Spline interpolation method was used (see Appendix N for the C# code implementation). The cubic Spline method has the advantage of providing a continuous interface and continuous derivatives between the interpolated segments. The segments at the boundaries were extrapolated also using the cubic Spline method.

The columns tests however did not provide changes in PSD along the depth of the column for the initial concentration profile. This is due to the fact that the amount of sediment withdrawn from the column's ports was not sufficient to run a valid PSD through the X-Ray particle size analyzer. Withdrawing larger samples for the initial condition would have depleted the total volume of water inside the column to a point that the level of water is too close or pass the first (top) sampling port. Therefore, a constant PSD along the depth of the column was assumed with the Spline-adjusted sediment concentrations. This assumption however may lead to important errors that will be discussed later.

5.5.3 Boundary Conditions

The boundary conditions at the top and bottom of the column need to be defined. The initial concentrations at the boundaries are not known and can't be assumed 0 and $\xi_{\max}$ at the top and bottom layers; respectively. The only boundary conditions that can be imposed are with respect to the flux function. Since the column remains closed during the duration of the test, no particles enter (at the top) or leave (at the bottom) the column at any time. This can be interpreted as no flux enters or leaves the column and thus:

$$F(\xi)\Big|_{z=0} = 0, \quad F(\xi)\Big|_{z=L} = 0, \quad \forall t. \tag{5.48}$$

5.5.4 Parameter Estimation (Inverse Value Problem)

Once a continuous approximation of the initial condition concentration profile has been created, the NT numerical method can be used to obtain an approximate solution of the second order. To obtain an estimate of the optimum d_0 and d_1 sedimentation coefficients, it is necessary to use an optimization tool. Given that the NT numerical scheme needs to be implemented for each iteration of the optimization process, a random search algorithm is necessary to solve the inverse value problem. The genetic algorithm presented in Chapter 4, Section 4.5.2 will be used for parameter estimation.

5.6 Results

The genetic algorithm was used to obtain an approximation of the optimum d_0 and d_1 sedimentation coefficients for each of the column tests listed in Appendices 5.A and 5.B, based on the time interval of 60 minutes. The goodness of fit is given by the coefficient of determination (R^2):

$$R^2 = \left[\frac{N\sum \xi_o \xi_p - \sum \xi_o \sum \xi_p}{\sqrt{N\sum \xi_o^2 - \left(\sum \xi_o\right)^2}\sqrt{N\sum \xi_p^2 - \left(\sum \xi_p\right)^2}} \right] \tag{5.49}$$

where N is the number of observed data points ($N = 6$ in this case), ξ_o are the observed concentration data points obtained directly from the column tests and ξ_p are the predicted concentration data points obtained from the calibration process.

Nine column tests were conducted using the small and large micro-spheres PSDs: three initial concentrations with three replicates, resulting in 18 total experiments. Data was incomplete for one of the replicates (replicate 1) of the large micro-spheres for the low initial concentration. The resulting R^2 values for both particles size distributions (small and large), Table 5 - 1 and the observed and predicted concentration profiles obtained after 60 minutes of settling time has elapsed for small and large micro-spheres are illustrated in Appendix O and Appendix P; respectively.

Table 5 - 2: R^2 values obtained from the calibration process

Glass Micro-Sphere	Low Initial Concentration (5,000mg/L)		
	Rep. 1	Rep. 2	Rep. 3
Small	0.972	0.929	0.915
Large	NA	0.556	0.957
Glass Micro-Sphere	Medium Initial Concentration (15,000 mg/L)		
	Rep. 1	Rep. 2	Rep. 3
Small	0.966	0.968	0.960
Large	0.971	0.914	0.927
Glass Micro-Sphere	High Initial Concentration (45,000 mg/L)		
	Rep. 1	Rep. 2	Rep. 3
Small	0.896	0.973	0.968
Large	0.927	0.915	0.923

The sedimentation coefficients d_0 and d_1 that resulted from the calibration process are listed in Tables 5 - 3 and 5 - 4 for the small and large glass micro-spheres, respectively.

Table 5 - 3: Calibrated sedimentation coefficients for the small glass micro-spheres

Initial Concentration	Replicate 1		Replicate 2		Replicate 3	
	$\mu_\infty d_0$	$\mu_\infty d_1$	$\mu_\infty d_0$	$\mu_\infty d_1$	$\mu_\infty d_0$	$\mu_\infty d_1$
Low (5,000 mg/L)	0.0215	2.939E-09	0.0765	8.283E-09	0.00141	1.932E-10
Medium (15,000 mg/L)	0.00738	1.714E-09	0.0105	1.448E-09	0.0502	3.254E-09
High (45,000 mg/L)	0.000711	2.605E-11	0.000335	6.562E-11	0.000458	2.684E-10

Table 5 - 4: Calibrated sedimentation coefficients for the large glass micro-spheres

Initial Concentration	Replicate 1		Replicate 2		Replicate 3	
	$\mu_\infty d_0$	$\mu_\infty d_1$	$\mu_\infty d_0$	$\mu_\infty d_1$	$\mu_\infty d_0$	$\mu_\infty d_1$
Low (5,000 mg/L)	N/A	N/A	-0.000766	-2.959E-11	-0.00136	-6.574E-10
Medium (15,000 mg/L)	-0.000493	-4.431E-11	-0.000285	-2.604E-11	-0.000289	-1.398E-11
High (45,000 mg/L)	-0.000178	-3.702E-10	-0.000112	-9.712E-11	-0.000133	-1.162E-10

For the small glass micro-spheres the resulting sedimentation coefficients values were all positive, which implies that the settling velocities of the particles settling inside the column were larger than the respective terminal velocities of particles without particle-to-particle interactions. This result was maintained even for high concentration; however with smaller coefficient values. One possible explanation is that the Batchelor's settling velocity equation may not be adequate to model the sedimentation process of fine particles (7 microns in diameter) when using the CKSM. Conversely, for the large micro-spheres calibrated sedimentation coefficients yielded the expected behavior where the influence of particle-to-particle interactions on the settling velocity is smaller that the terminal velocity.

The observed 60 minutes concentration profile depicted in Figure 5 - 26 in Appendix L (replicate 1 with small glass micro-spheres and high initial concentration) shows an atypical behavior where the concentration near the bottom of the column is much smaller than the up-gradient concentration. This atypical behavior may be due to the fact that the PSD is not known at the different depth in the column and thus it is expected that the initial concentration profile may have had larger size particles near the bottom of the column before settling started.

5.7 Summary and Concluding Remarks

A new one dimensional scalar continuous sedimentation model was developed from the well-known discrete kinematic sedimentation model based on the Kynch theory of sedimentation. A total flux function was defined as the integral of the flux for the entire particle size spectrum. Using this definition, an analytical continuous representation of the total flux function was derived for the three different settling velocity models presented in this chapter: Batchelor, Modified Batchelor and Richardson and Zaki. Table 5 - 5 summarizes the total flux functions for the different settling velocity models.

Table 5 - 5: Total flux functions for the three different settling velocity models

Settling Velocity Model	Total Flux Function $F(\xi)$
Batchelor	$F^{B}(\xi) = \xi\mu_{\infty}\left[M_2 + \xi\left(d_0 M_1^2 + d_1 M_2\right)\right]$
Modified Batchelor	$F^{B\,\mathrm{mod}}(\xi) = \xi\mu_{\infty}\exp\left(2\frac{\xi}{\xi_{\max}}\right)\left(1-\frac{\xi}{\xi_{\max}}\right)^2 \exp(\xi)\left(d_0 M_1^2 + d_1 M_2\right)$
Richardson and Zaki	$F^{RZ}(\xi) = \xi\mu_{\infty}\left(1-\xi\right)^{\eta} M_2$

The PSD of natural occurring sediment particles covers a large spectrum of sizes, ranging from large sand to fine clay particles. This imposes an important limitation when using the discrete KSM since the number of particle size species that need to be used to have a representative discrete PSD to obtain accurate results is extremely difficult to estimate. The larger the number of size species used the larger is the computational cost embedded in trying to obtain an accurate solution through a numerical method since there is no analytical solution to these types of systems of hyperbolic conservation laws. The continuous representation of the sedimentation model presented in this Chapter simplifies the model to a scalar (single) conservation law and thus the computational cost is considerably reduced. Furthermore, calibration of a scalar model is quite less complex than calibration of a system of conservation laws since there are only two parameters to calibrate.

The continuous sedimentation model provides a prediction of the total concentration as a function of depth and time. Most sediment regulations are based on the effluent sediment concentration. The prediction of total sediment concentration, for depth and time, is sufficient to estimate effluent concentration and thus determine if a defined effluent sediment concentration is achieved for a single sediment pond. At most construction sites (approximately 90%) only a single pond is used within a watershed for sediment control. A disadvantage of this model is that it does not predict the concentration of each particle species separately. Not predicting the specific concentration of each particle size, as a function of depth and time, restricts the use of this model to a single sediment pond since for a system of ponds in series the PSD information as a function of depth of the outlet flow of the previous sediment pond will influence the performance of the next. An expanded model needs to be developed that considers shifts of the PSD due to the sedimentation process, Chapter 6.

Calibration of the CKSM was conducted employing two primary procedures. The first procedure was to select a numerical method that is both non-oscillatory and shock-absorbing while having a small computational cost (computationally efficient). Under these circumstances the non-staggered version of the NT central scheme was selected to solve the CKSM. The second step is to obtain an approximation of the optimum sedimentation coefficients (parameters) using the genetic algorithm that was presented in Chapter 4, Section 4.5.2. The CKSM is an initial value problem and thus the predictive capability is driven by an accurate representation of the initial concentration profile. The initial concentration profile for every point in the grid was estimated from the 6 available observed points using the cubic Spline interpolation method to avoid discontinuities within the initial concentration profile that can otherwise lead to instabilities.

The calibration process revealed that the proposed continuous model fits the glass micro-sphere data well ($R^2 > 0.89$ except for one test replicate) and therefore it represents an excellent mathematical tool to model the sedimentation process. It is important however to note that not having the initial variation of the PSD along the depth (z-axis) of the column may result in significant error levels when larger particles are present in the PSD (silts and sands). Larger particles are usually at the bottom layers of the column since it requires a higher mixing energy to re-suspend them to the top. Conversely, smaller particles are usually distributed homogenously along the depth, which explains the excellent results of this study.

In conclusion, the CKSM is a theoretically rigorous, yet computationally efficient sedimentation model that is also readily applicable and expandable to two or three dimensions. However, this model requires additional development specifically addressing the defined initial concentration profile that incorporates variations in depth of the PSD to obtain accurate results when modeling sediment PSDs.

CHAPTER 6: Future Research

The research presented in the previous chapters provided new developments in the area of flocculation and sedimentation modeling. These new developments are considered as a first step to obtaining a comprehensive physically based sedimentation model that can be applied to a sediment pond. Logically, this leads to further research suggestions.

A new continuous aggregation and fragmentation model derived from the well-known Smoluchowski coagulation equation using the Gamma distribution was developed and presented in Chapter 3. The proposed flocculation model was calibrated using limited sediment and flocculent data. Results are very promising. It is recommended that this modeling approach be extended through testing with additional soils and flocculants to statistically establish calibrated parameters with associated soil type and flocculent products.

The continuous kinematic sedimentation model (CKSM) presented in Chapter 5 is a completely new approach to modeling the settling process of particles in one dimension; however it is only the first step in obtaining a comprehensive sediment pond design model. The CKSM therefore opens new opportunities for research. Results in Chapter 5 showed that the Batchelor's equation needs to be modified in order to improve modeling the settling velocities of fine particles (7 microns in diameter) when using the CKSM; however it showed promising results with larger particles (25 microns in diameter). Thus it may be necessary to explore a new settling velocity model that will better accommodate the smaller size fraction of sediments. Additionally, the CKSM was calibrated using glass micro-spheres only and thus calibration and validation of the model using column test data obtained using sediment is recommended. Calibration using sediment will also need to be conducted considering the use of flocculation (combining the CKSM with the flocculation model proposed in Chapter 3).

Obtaining the PSD of samples withdrawn from the column tests was extremely difficult since the amount of volume and the concentration were too small to test using the X-Ray particle size analyzer. No PSD data was available from the column tests and thus to calibrate the CKSM it was necessary to assume that the initial concentration profile had a constant PSD along the entire depth of the column. This is obviously a restriction to the CKSM. To withdraw larger samples from the column a larger diameter column would need to be used to reduce the water level drop inside the column during the test. A larger column however would require larger mixing energy to obtain a homogeneous initial concentration profile and sampling procedures may need to be modified to ensure a representative sample being drawn.

The settling velocity expressions associated with the CKSM utilize the first two moments of the PSD. Assuming the PSD is available for the entire set of concentration profiles, a relationship that relates the moments of the initial PSD (before settling occurs, i.e. from the original PSD) to the changes in PSD as a function of depth and time can be obtained. Using this relationship it could also be possible in conjunction with the CKSM to obtain the PSD as a function of time and depth. This relationship however needs to obey the following two conditions:

1) initial condition: $M_k(0,z) = M_k^0(z)$ is known, and

2) terminal condition: $\lim_{t\to\infty} M_k(z=0) = 0$, and $\lim_{t\to\infty} M_k(z=L) = M_{k,\max}$, $k = 1,2$.

The two moments posses minimum and maximum values that depend on the physics of the problem. The minimum value is attained at the top of the column when a large amount of time has elapsed. A first order decay model can be used for example to describe the decrease in the average size diameter at the top of the column:

$$\begin{aligned} M_{1,\max} &= M_1\left(2 - e^{-\delta_1 t}\right), \\ M_{1,\min} &= 2M_1 - M_{1,\max} = M_1 e^{-\delta_1 t} \end{aligned} \tag{6.1}$$

Finally, to provide more flexibility to the CKSM the sedimentation coefficients can be set as a function of depth, i.e. $d_0 = d_0(z)$ and $d_1 = d_1(z)$. The numerical method used to solve the CKSM needs to update the values of M_1, M_2, d_0 and d_1 at each time step.

APPENDIX A: Multimodal Form – proof by induction

The multimodal approach states that the PSD of a sediment resulting from completely mixing a finite number of sediments with known PSDs can be written as the superposition (linear combination) of the known PSDs functions (2.8), assuming that no particle-to-particle interaction occur. This will proven by induction.

For n = 1 the result is trivially obtained by replacing $\omega_1 = 1$. Therefore assume that the result holds for n-1 and then it will be proven it holds for n.

Let $M^{n\text{-}1}$ be the mass of sediment of the n-1 original combined sediment samples, thus $M^{n-1} = \sum_{i=1}^{n-1} M_i$ and let M_n be the mass of the added sediment material that complete the n different portions of the final sediment sample. The total mass of the final sample is then $M^{n-1} + M_n = \sum_{i=1}^{n} M_i$. The cumulative PSD functions for the n-1 composite sediment and the n sediment portion are $F^{n-1}(x) = \sum_{i=1}^{n-1} \omega_i \cdot F_i(x)$ and $F_n(x)$ respectively. Let $x' \in \Omega$ be any diameter inside the domain of definition of all the particle size classes $\left(\Omega = \bigcup_{i=1}^{n} \Omega_i\right)$ and let m_i be the mass of sediment sample i finer than x', $i = 1 \dots n$ .Then

$$F^{n-1}(x') = \sum_{i=1}^{n-1} \omega_i \cdot F_i(x') = \frac{m^{n-1}}{M^{n-1}} = \frac{\sum_{i=1}^{n-1} m_i}{\sum_{i=1}^{n-1} M_i},$$

$$w_i = \frac{M_i}{M^{n-1}}, \quad i = 1 \ldots n-1$$

$$F_n(x') = \frac{m_n}{M_n}$$

Similarly, the percent finer than x' of the final combined sediment sample is given by

$F(x') = \dfrac{m^{n+1} + m_n}{M^{n+1} + M_n} = \dfrac{\sum_{i=1}^{n} m_i}{\sum_{i=1}^{n} M_i}$. Now, it is easy to prove that

$$\begin{aligned} F(x') - F^{n-1}(x') - F_n(x') &= \frac{m^{n-1} + m_n}{M^{n-1} + M_n} - \frac{m^{n-1}}{M^{n-1}} - \frac{m_n}{M_n} = -\frac{m^{n-1} M_n}{\left(M^{n-1} + M_n\right) M^{n-1}} - \frac{m_n M^{n-1}}{\left(M^{n-1} + M_n\right) M_n} \\ &= -F^{n-1}(x') \cdot \frac{M_n}{M^{n-1} + M_n} - F_n(x') \cdot \frac{M^{n-1}}{M^{n-1} + M_n} \end{aligned}$$

Thus,

$$F(x') = F^{n-1}(x') + F_n(x') - F^{n-1}(x') \cdot \frac{M_n}{M^{n-1} + M_n} - F_n(x') \cdot \frac{M^{n-1}}{M^{n-1} + M_n}$$

$$= F^{n-1}(x') \cdot \frac{M^{n-1}}{M^{n-1} + M_n} + F_n(x') \frac{M_n}{M^{n-1} + M_n}$$

$$= \sum_{i=1}^{n-1} \omega_i \cdot F_i(x') \cdot \frac{M^{n-1}}{M^{n-1} + M_n} + F_n(x') \frac{M_n}{M^{n-1} + M_n}$$

Let $\omega_i' = \omega_i \cdot \frac{M^{n-1}}{M^{n-1} + M_n}, \quad i = 1 \ldots n-1,$ and let $\omega_n' = \frac{M_n}{M^{n-1} + M_n}$, then

$$F(x') = \sum_{i=1}^{n} \omega_i' \cdot F_i(x')$$

The above result is true for all $x' \in \Omega$, thus concluding the result (2.8). Finally, only $\sum_{i=1}^{n} w_i' = 1$ remains to be proven, indeed

$$\sum_{i=1}^{n} w_i' = \frac{M^{n-1}}{M^{n-1} + M_n} \sum_{i=1}^{n-1} w_i + \frac{M_n}{M^{n-1} + M_n} = \frac{M^{n-1}}{M^{n-1} + M_n} + \frac{M_n}{M^{n-1} + M_n} = 1$$

End of proof

APPENDIX B: Glass Micro-spheres PSD Data Curves

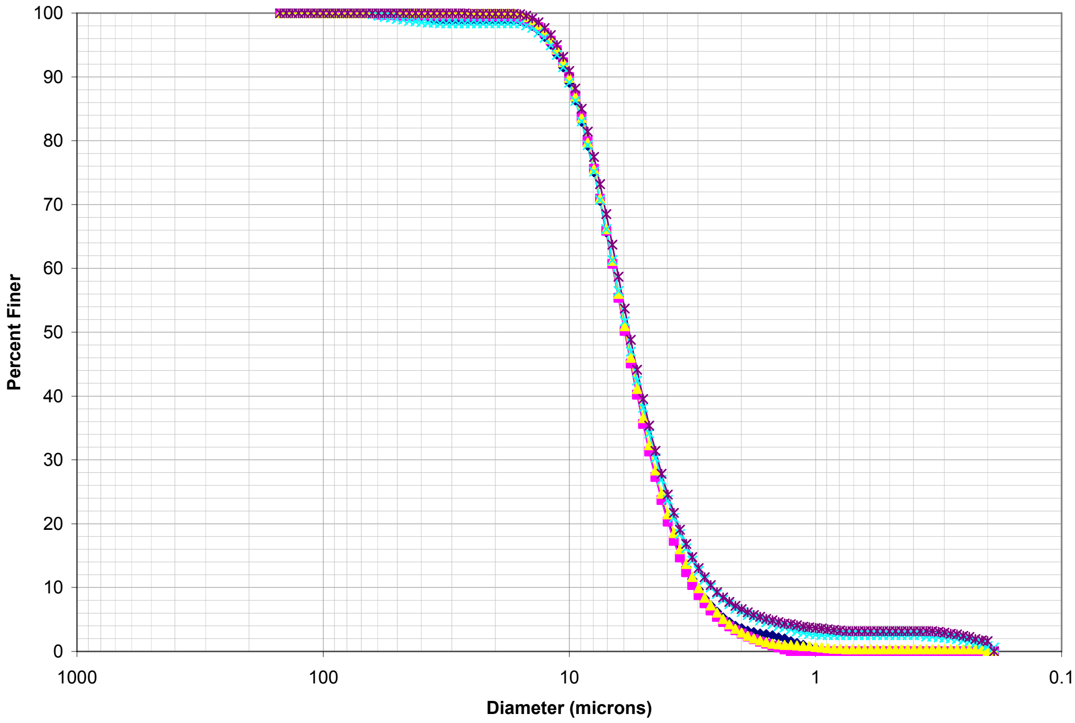

Figure 2 - 5: Small (7 μm) glass micro-spheres PSD curves obtained using the X-Ray particle size analyzer.

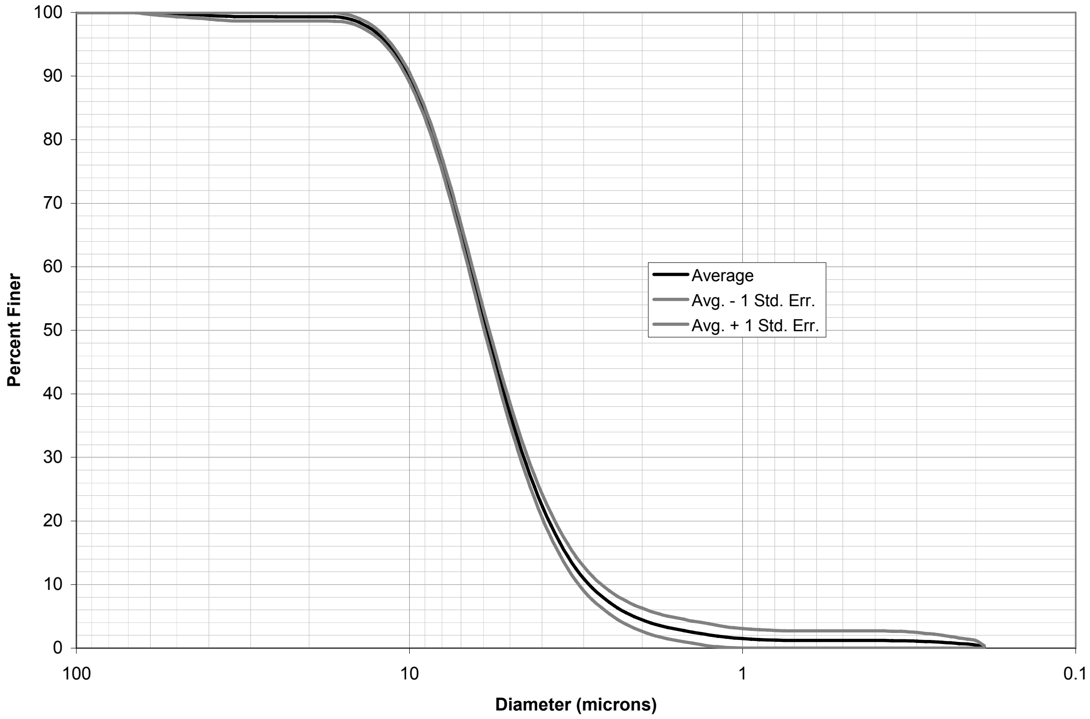

Figure 2 - 6: Average PSD curve for small (7 μm) glass micro-spheres.

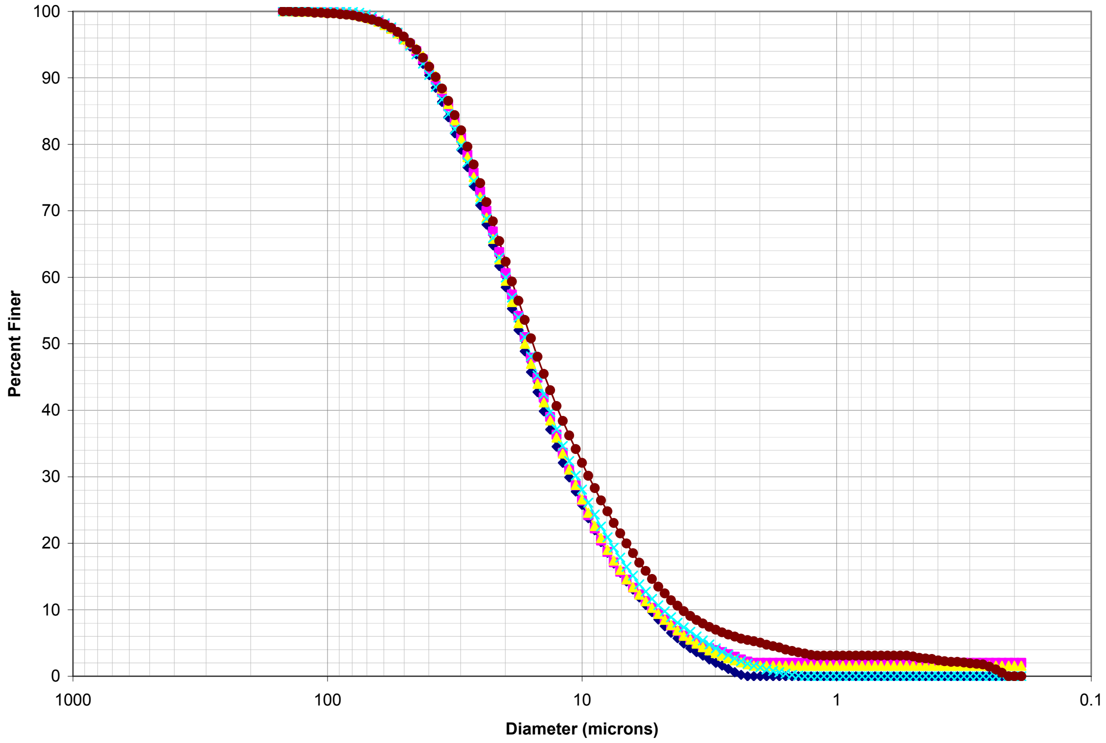

Figure 2 - 7: Medium (25 μm) glass micro-spheres PSD curves obtained using the X-Ray particle size analyzer.

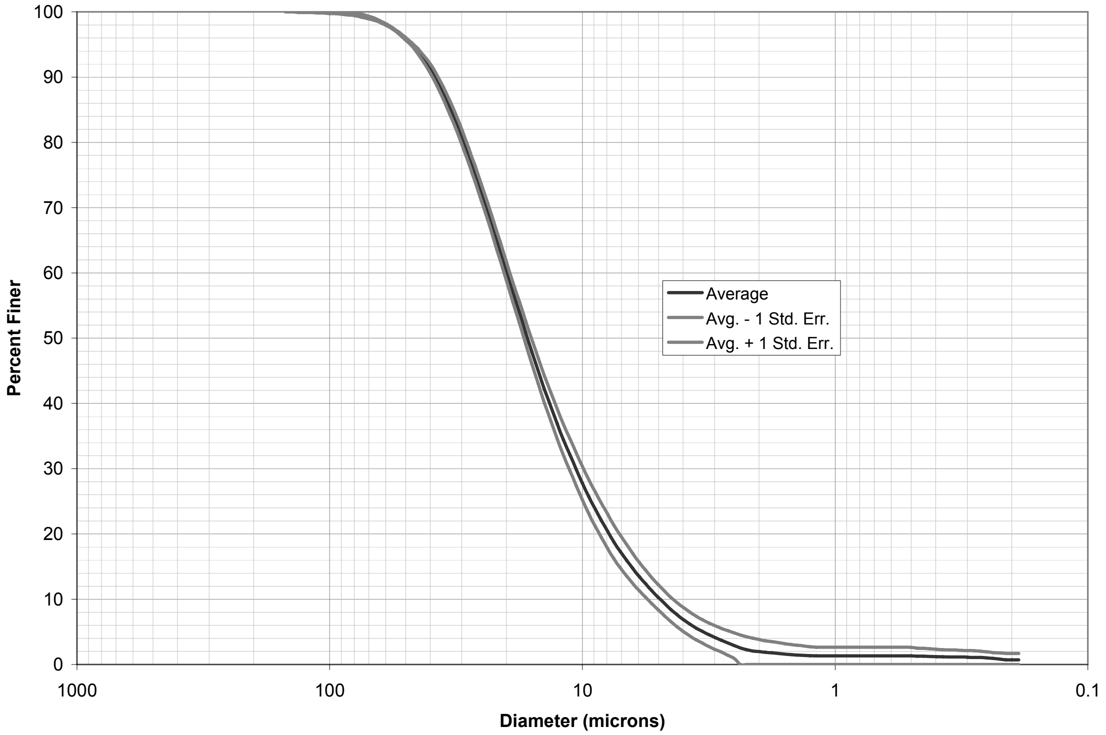

Figure 2 - 8: Average PSD curve for medium (25 μm) glass micro-spheres.

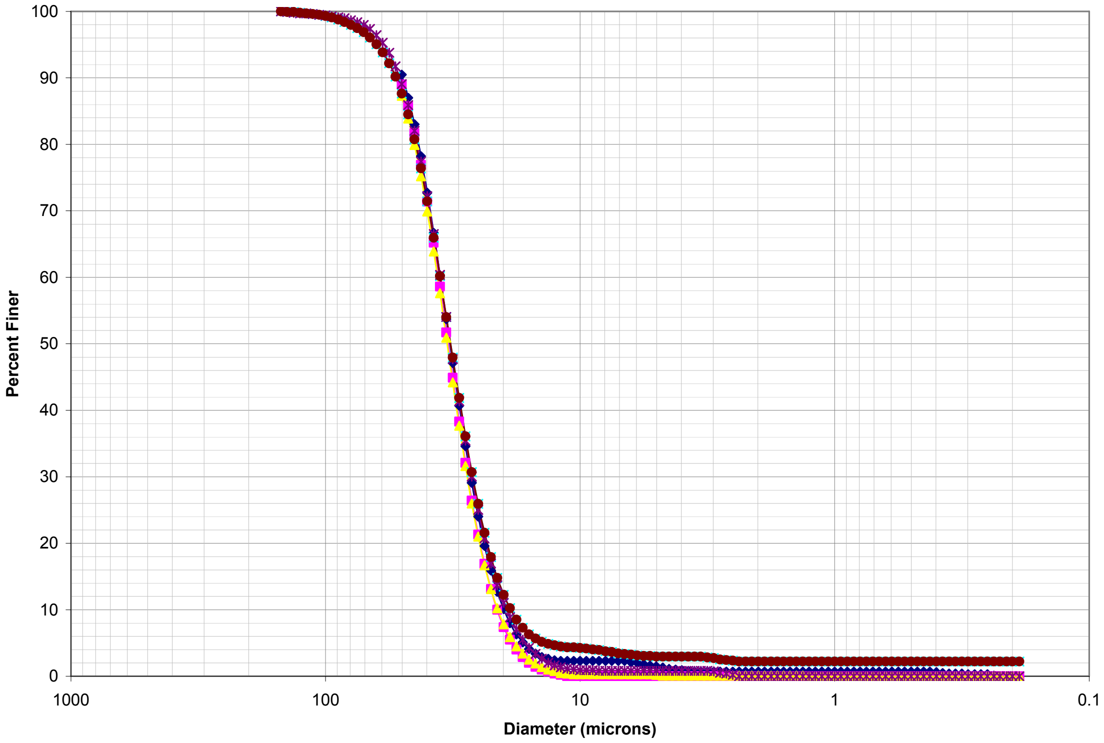

Figure 2 - 9: Large (35 μm) glass micro-spheres PSD curves obtained using the X-Ray particle size analyzer.

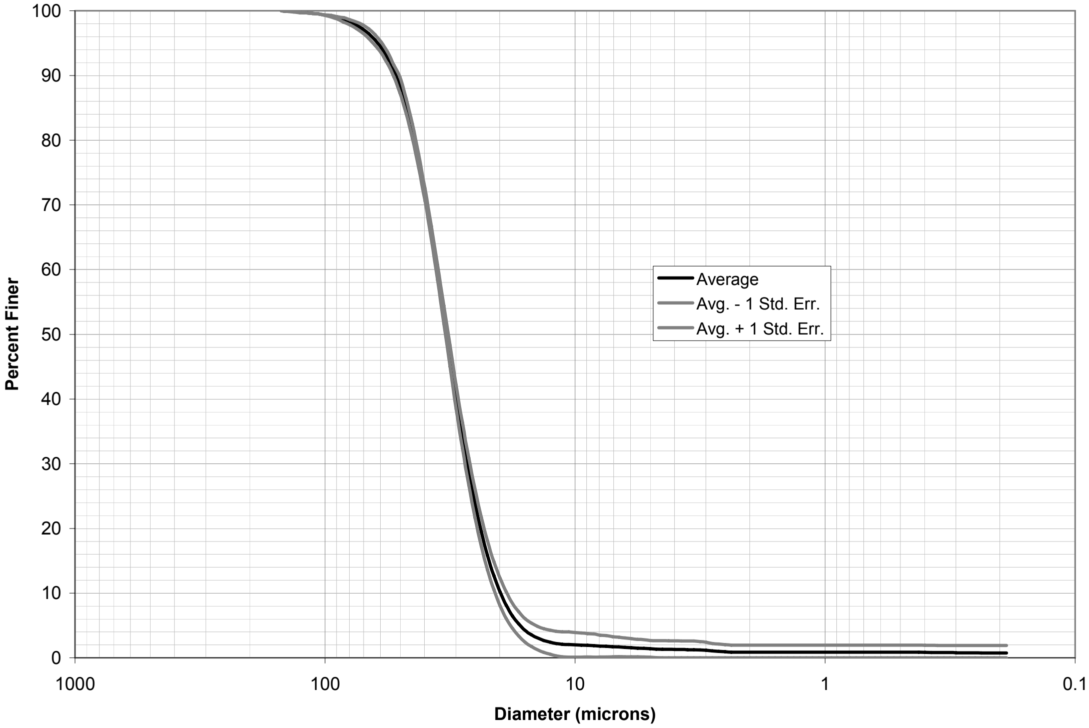

Figure 2 - 10: Average PSD curve for large (35 μm) glass micro-spheres.

APPENDIX C: Bimodal PSD Test Results

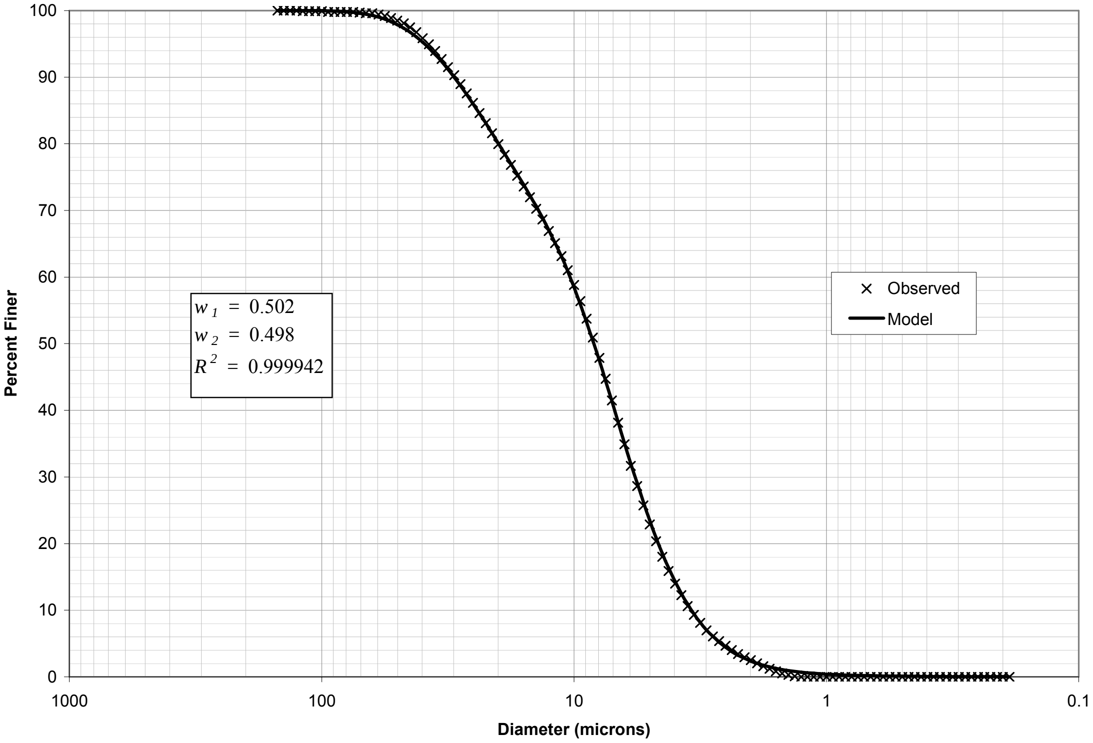

Figure 2 - 11: Bimodal PSD test # 1, replicate # 1.

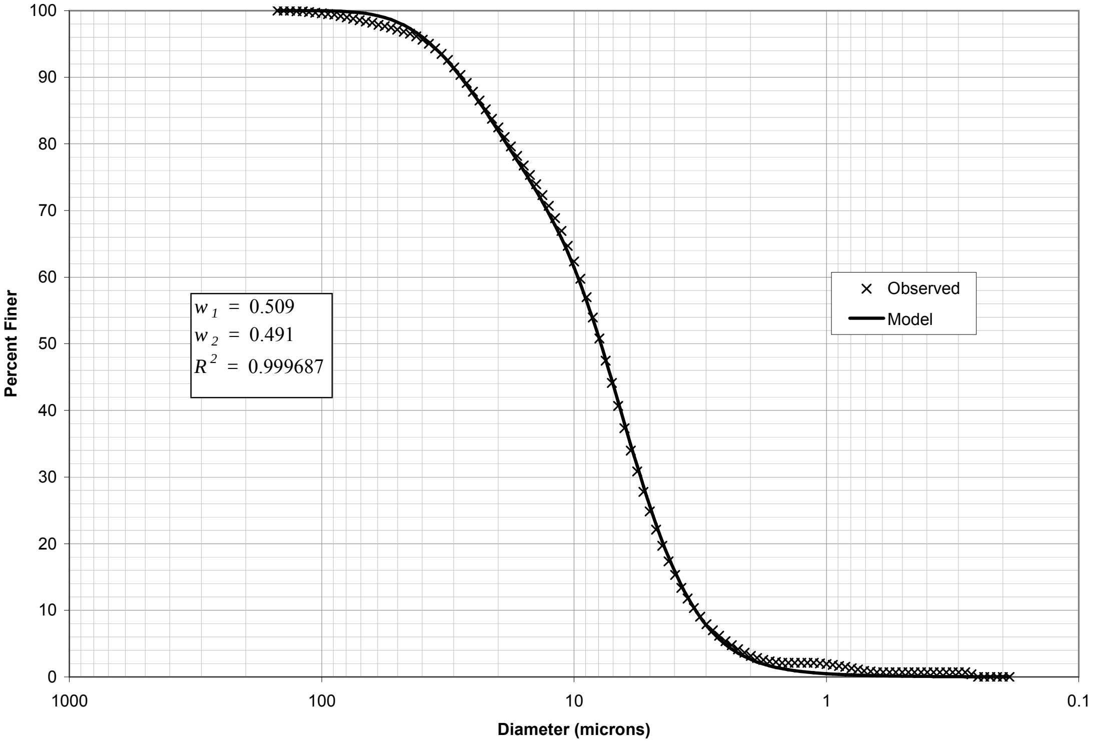

Figure 2 - 12: Bimodal PSD test # 1, replicate # 2.

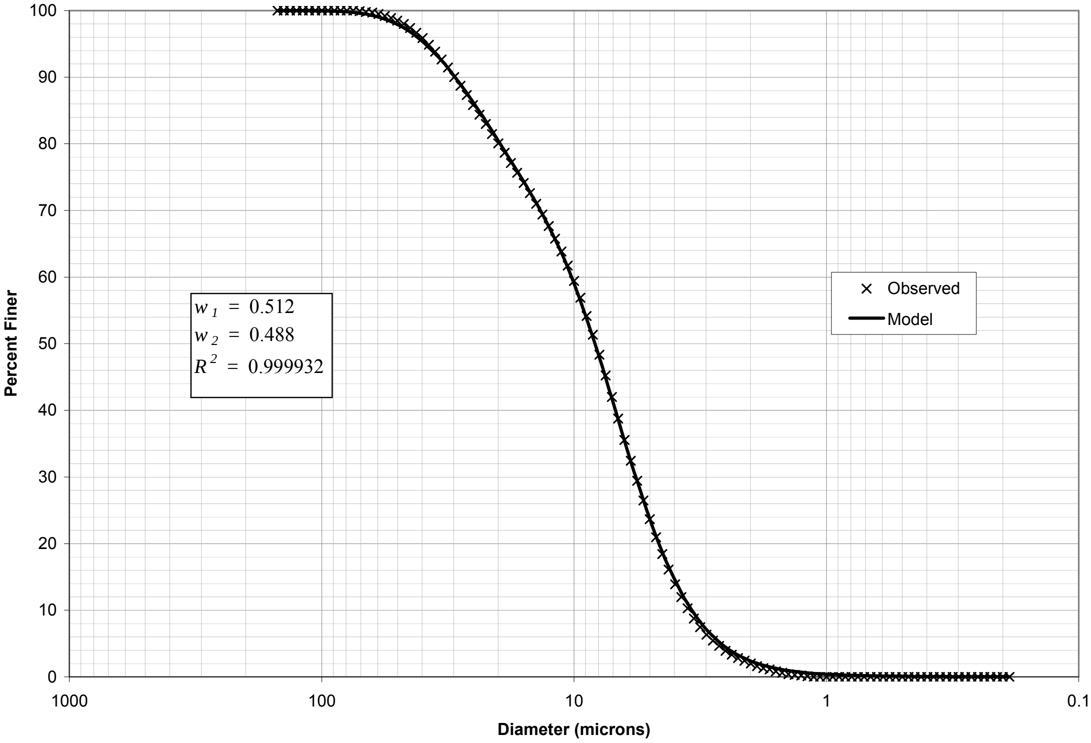

Figure 2 - 13: Bimodal PSD test # 1, replicate # 3.

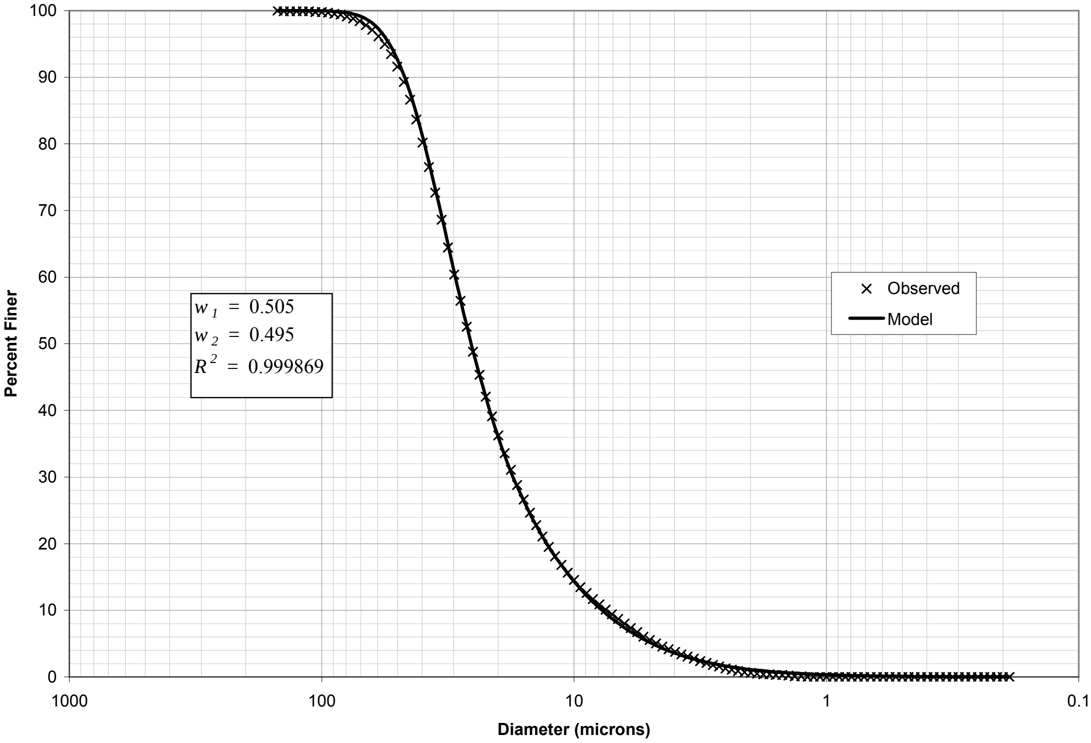

Figure 2 - 14: Bimodal PSD test # 2, replicate # 1.

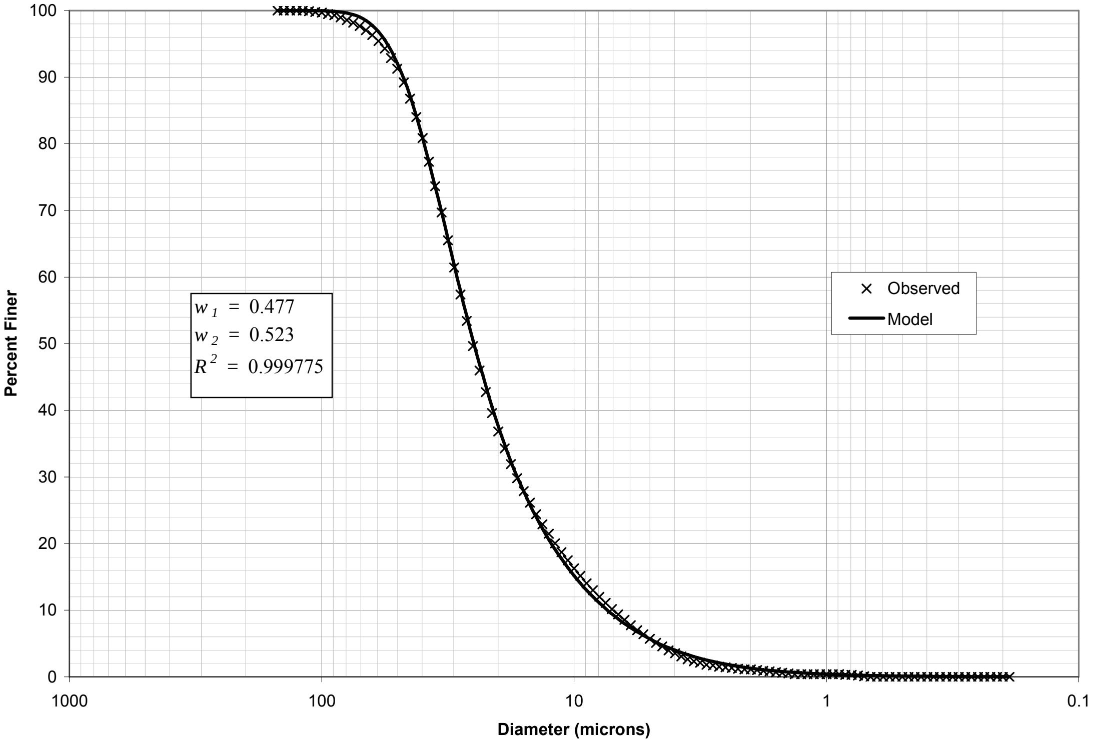

Figure 2 - 15: Bimodal PSD test # 2, replicate # 2.

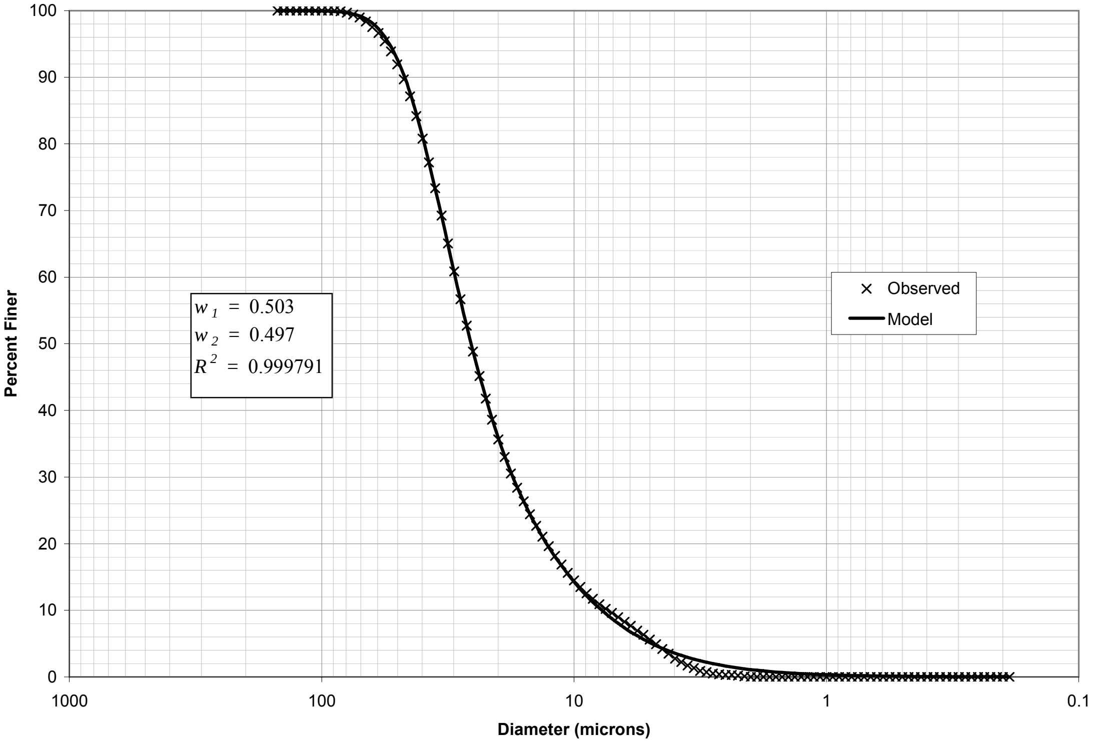

Figure 2 - 16: Bimodal PSD test # 2, replicate # 3.

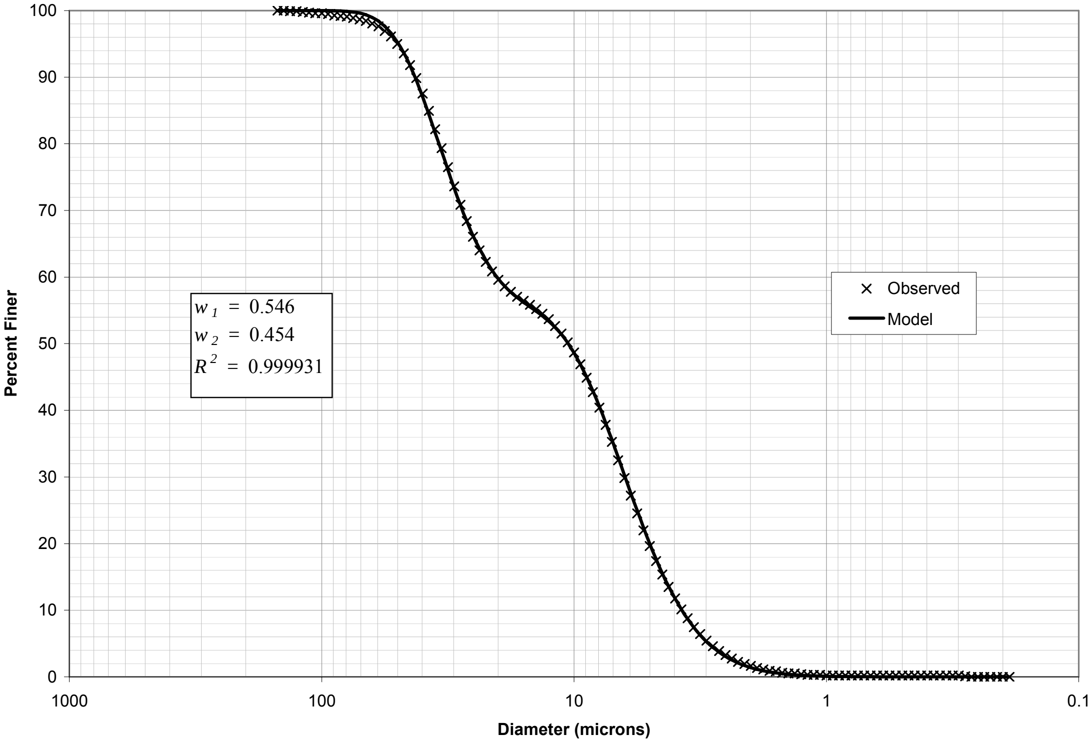

Figure 2 - 17: Bimodal PSD test # 3, replicate # 1.

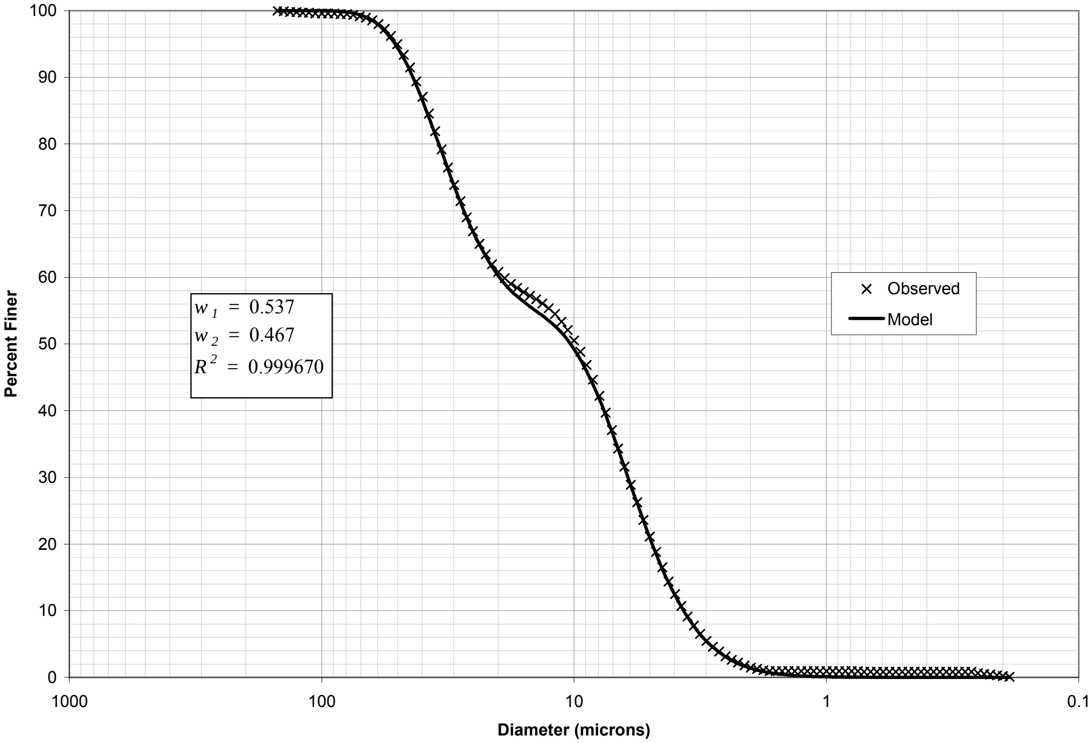

Figure 2 - 18: Bimodal PSD test # 3, replicate # 2.

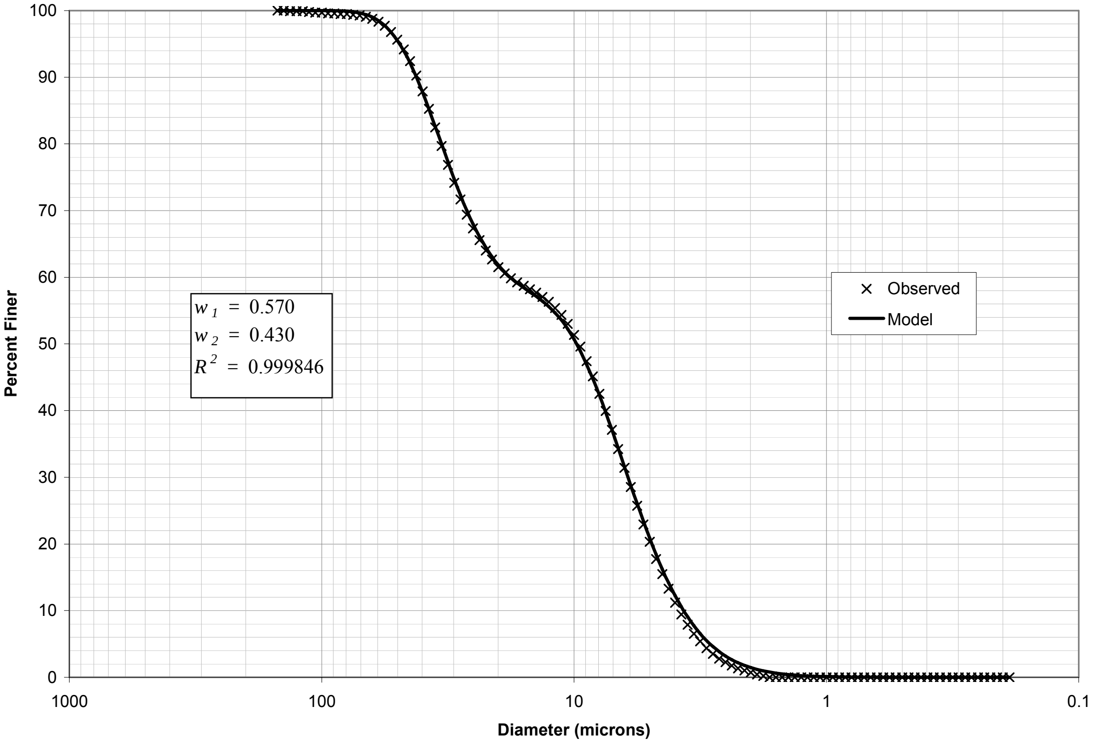

Figure 2 - 19: Bimodal PSD test # 3, replicate # 3.

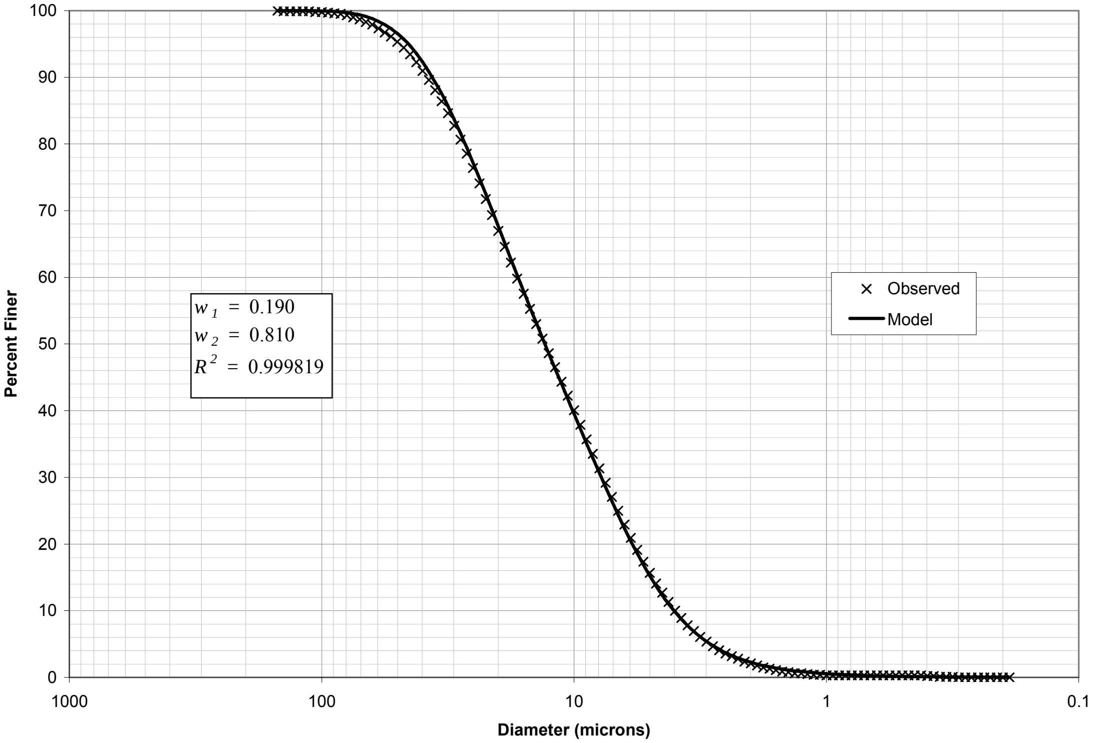

Figure 2 - 20: Bimodal PSD test # 4, replicate # 1.

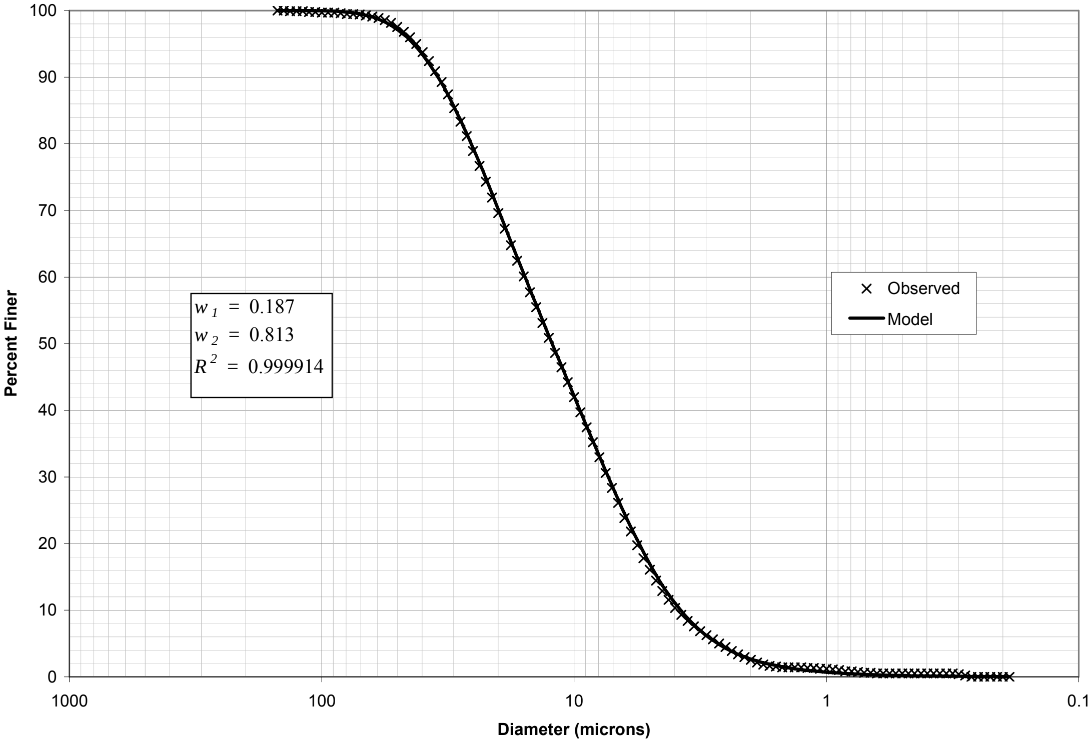

Figure 2 - 21: Bimodal PSD test # 4, replicate # 2.

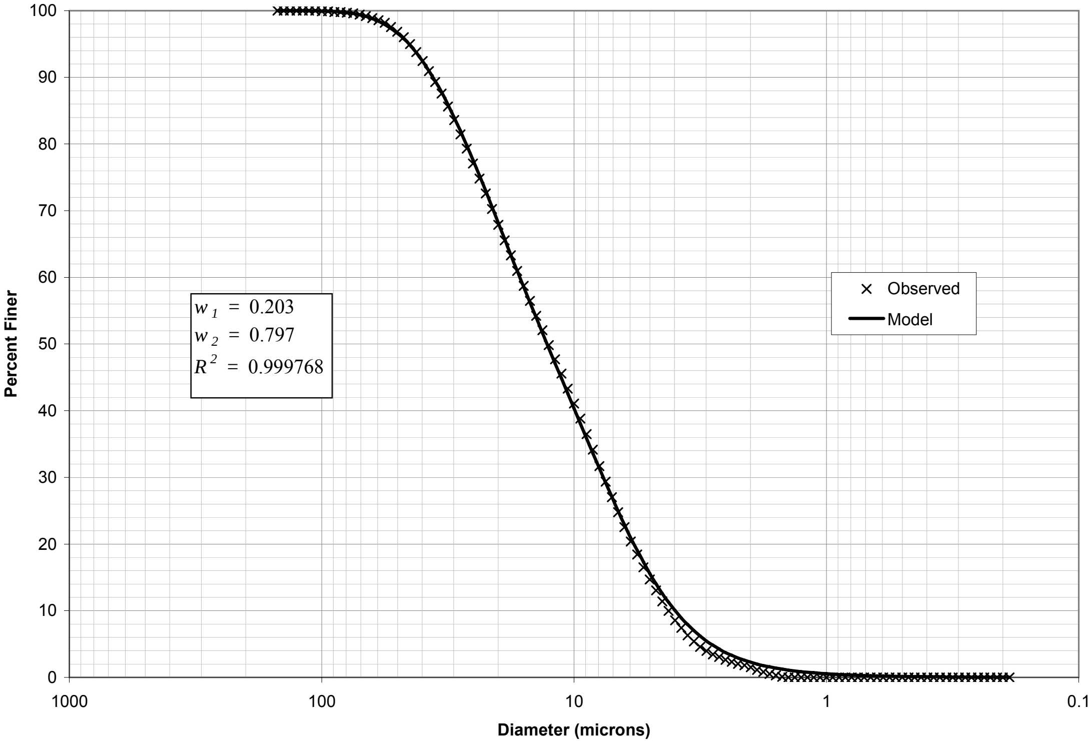

Figure 2 - 22: Bimodal PSD test # 4, replicate # 3.

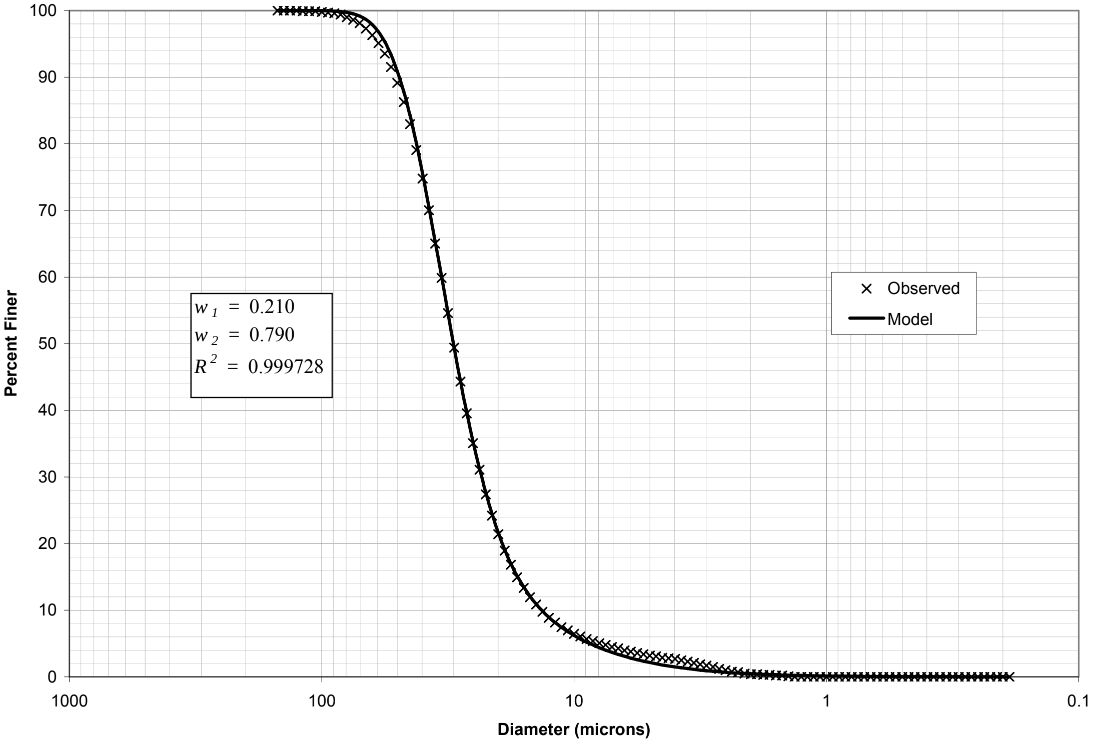

Figure 2 - 23: Bimodal PSD test # 5, replicate # 1.

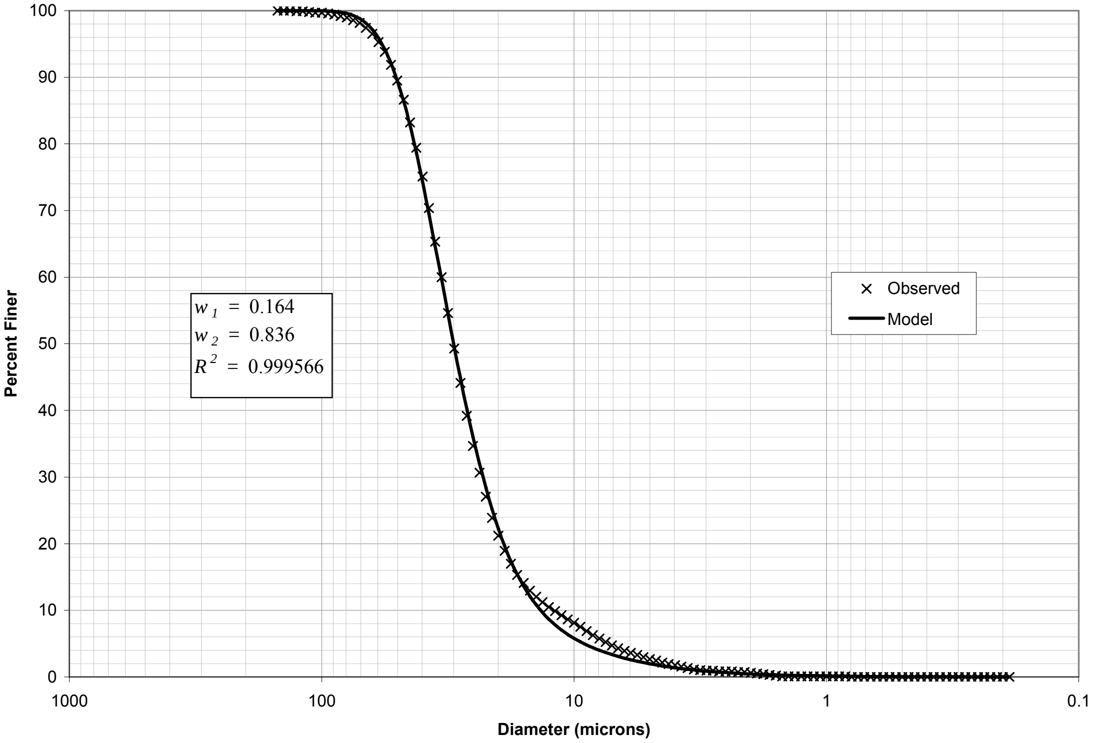

Figure 2 - 24: Bimodal PSD test # 5, replicate # 2.

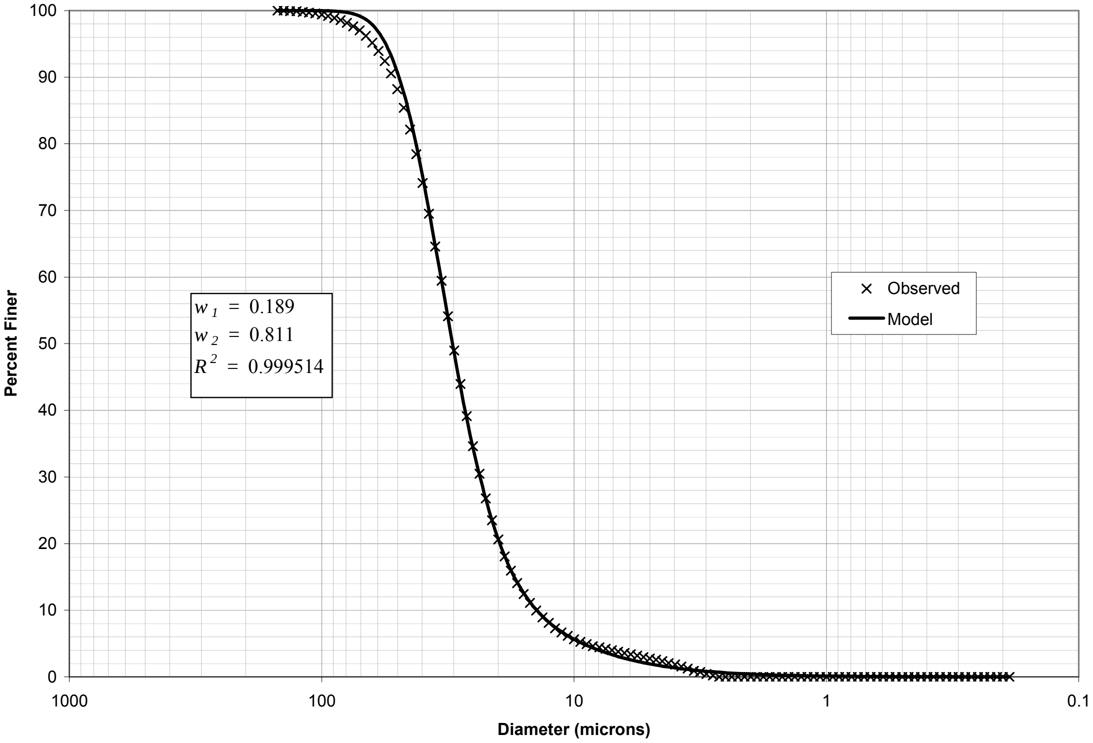

Figure 2 - 25: Bimodal PSD test # 5, replicate # 3.

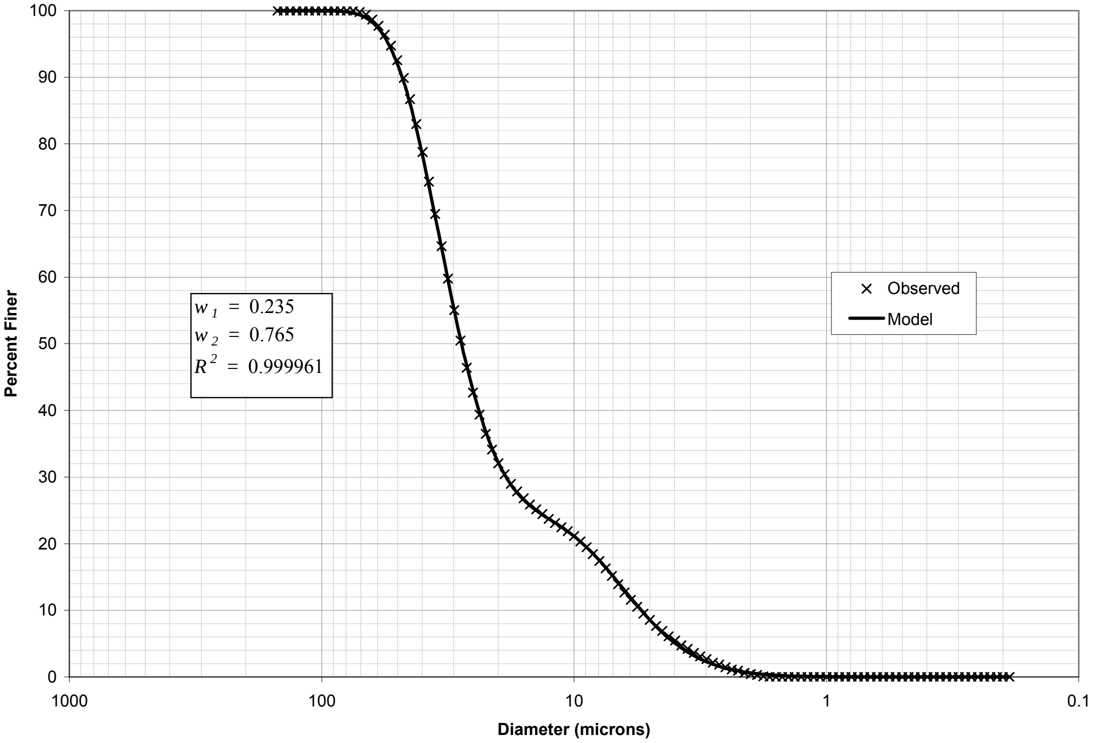

Figure 2 - 26: Bimodal PSD test # 6, replicate # 1.

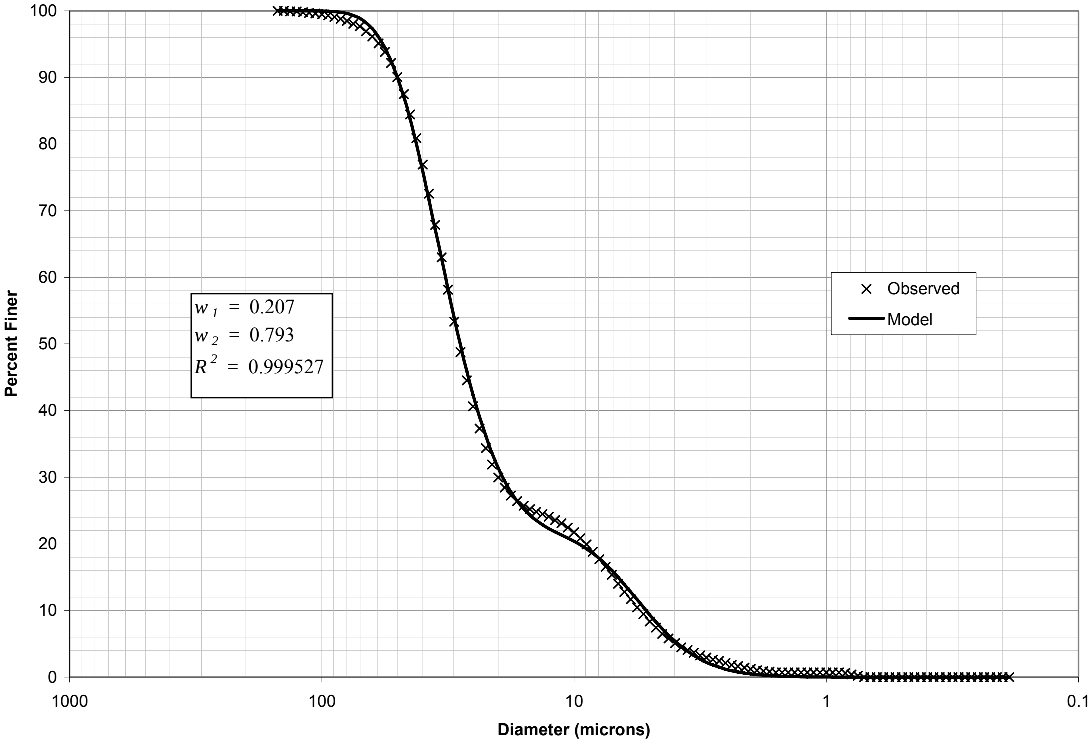

Figure 2 - 27: Bimodal PSD test # 6, replicate # 2.

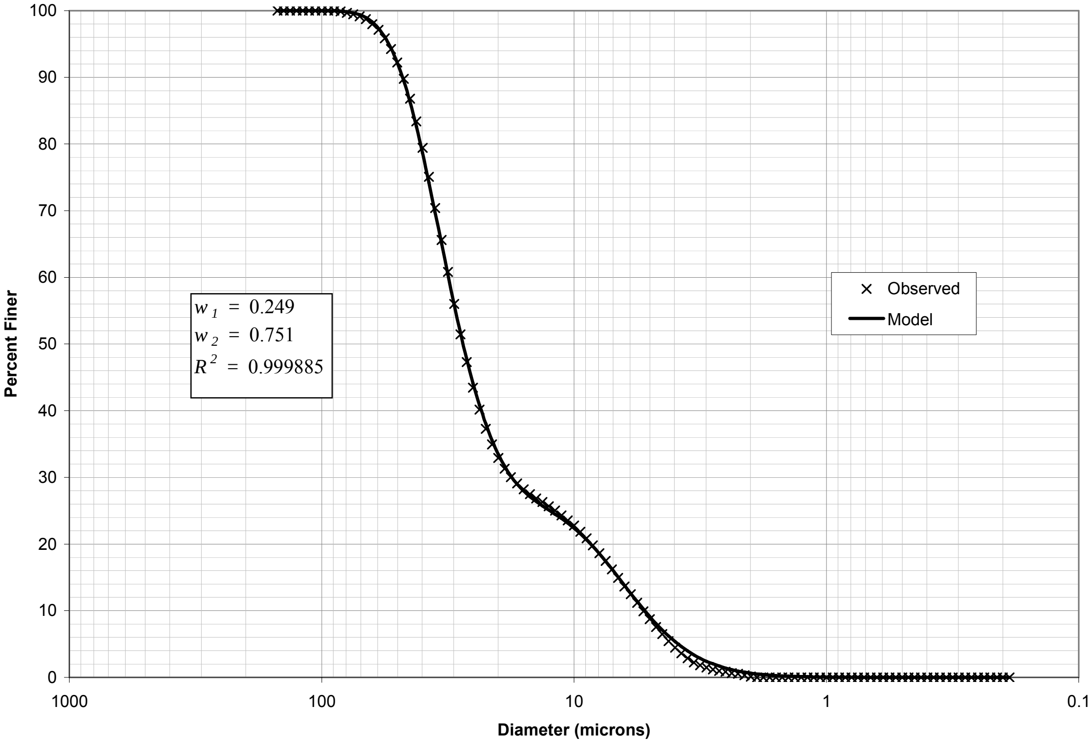

Figure 2 - 28: Bimodal PSD test # 6, replicate # 3.

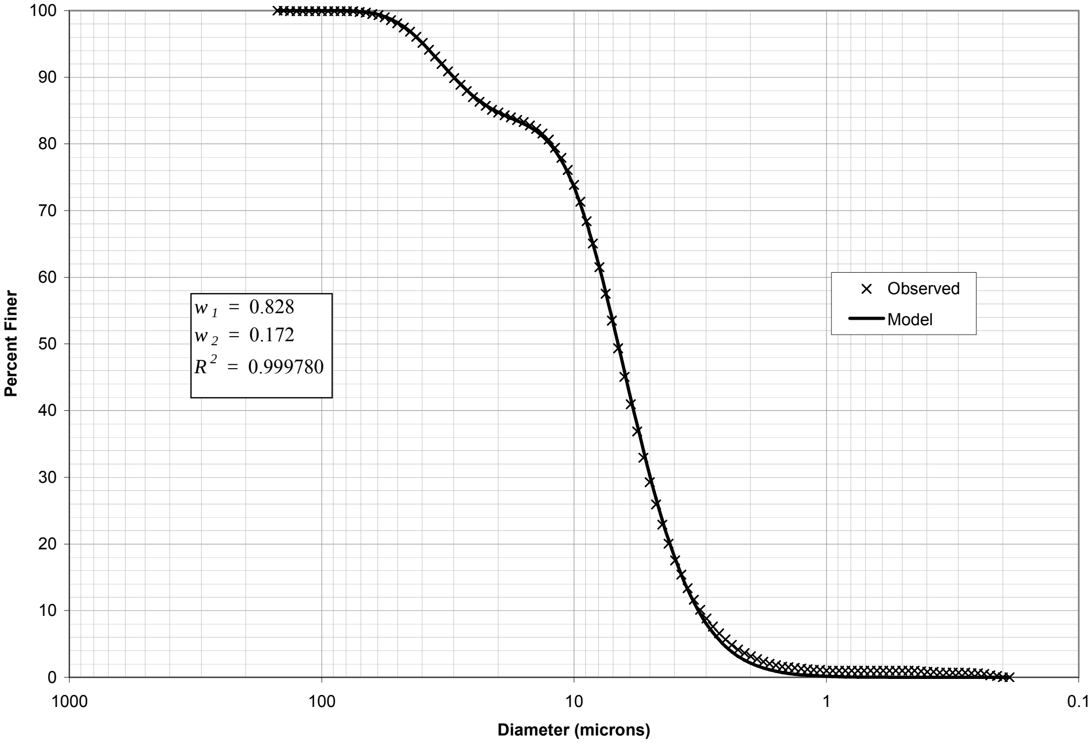

Figure 2 - 29: Bimodal PSD test # 7, replicate # 1.

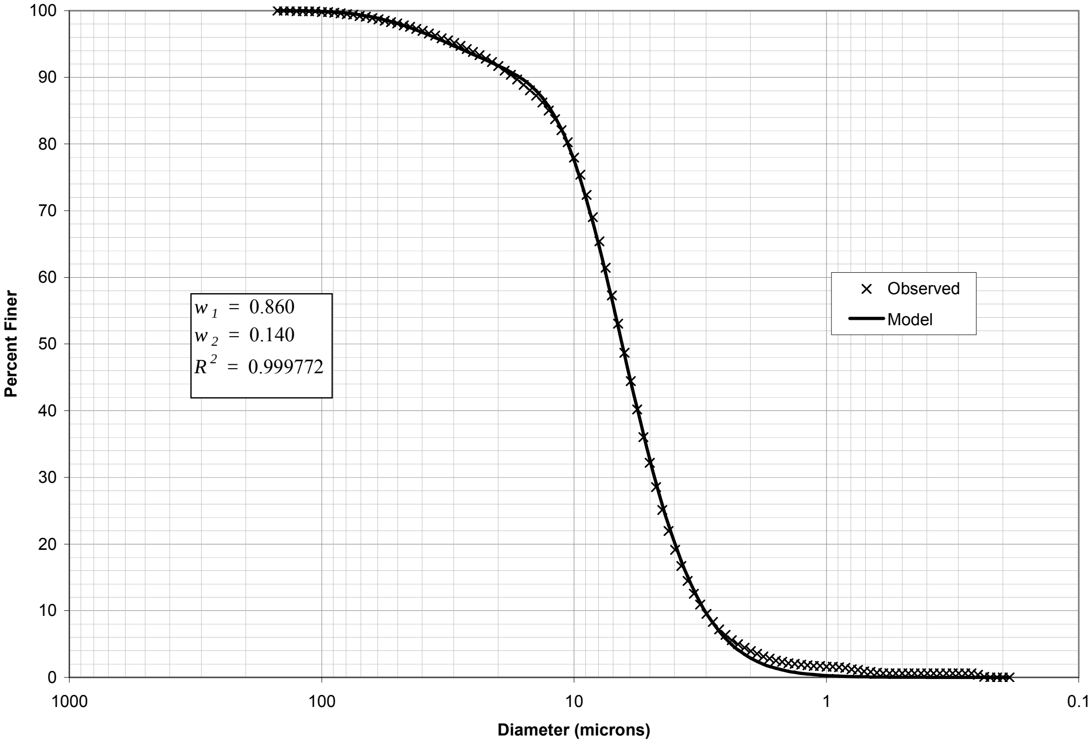

Figure 2 - 30: Bimodal PSD test # 7, replicate # 2.

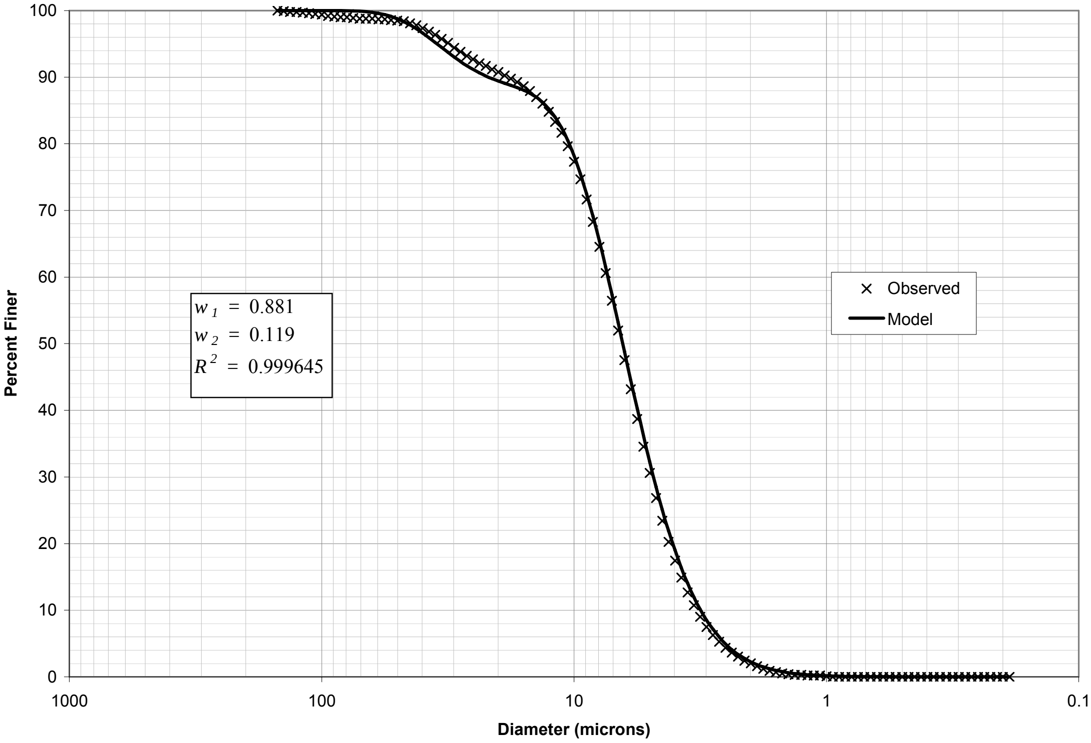

Figure 2 - 31: Bimodal PSD test # 7, replicate # 3.

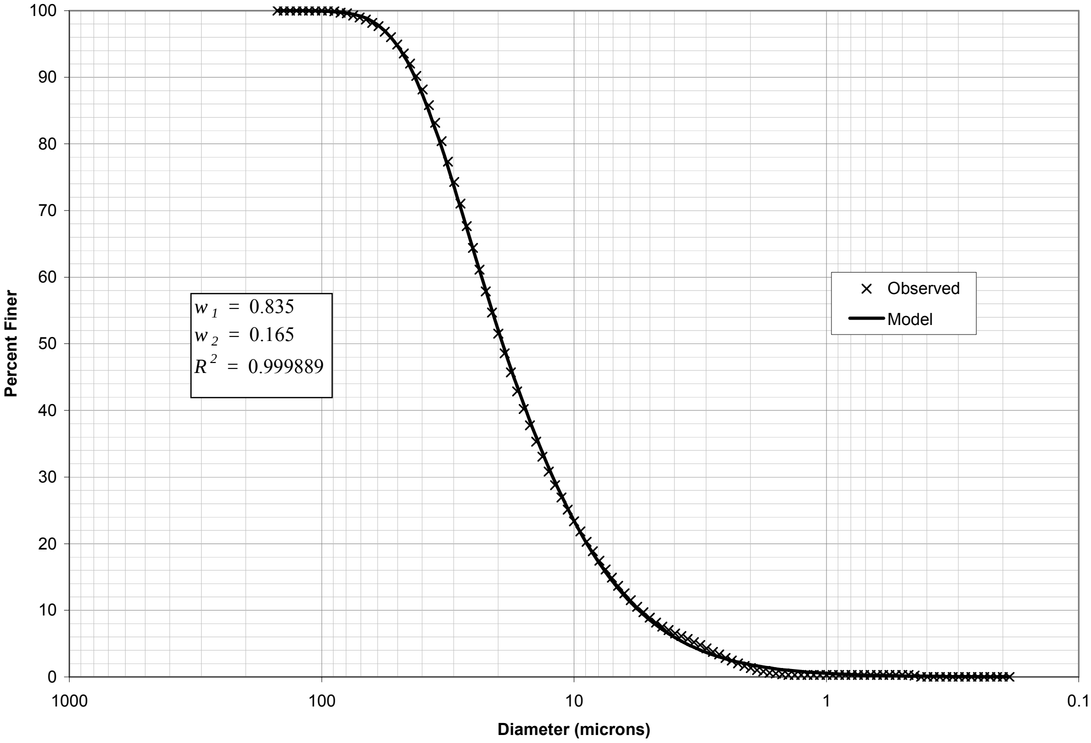

Figure 2 - 32: Bimodal PSD test # 8, replicate # 1.

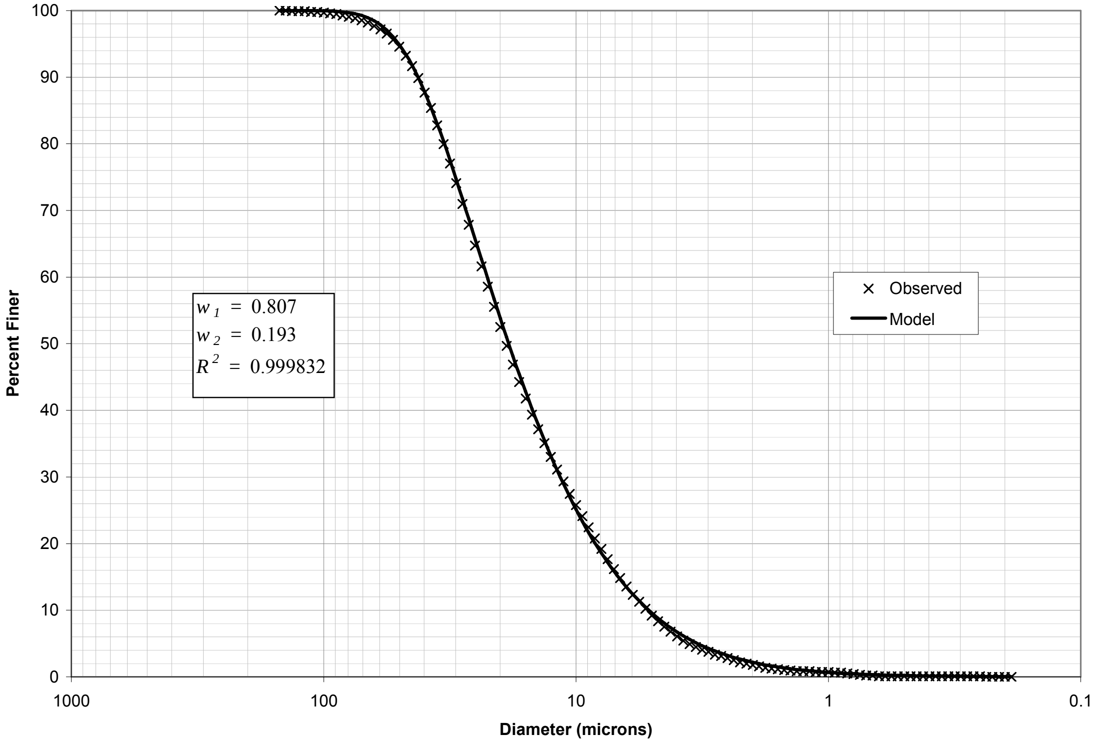

Figure 2 - 33: Bimodal PSD test # 8, replicate # 2.

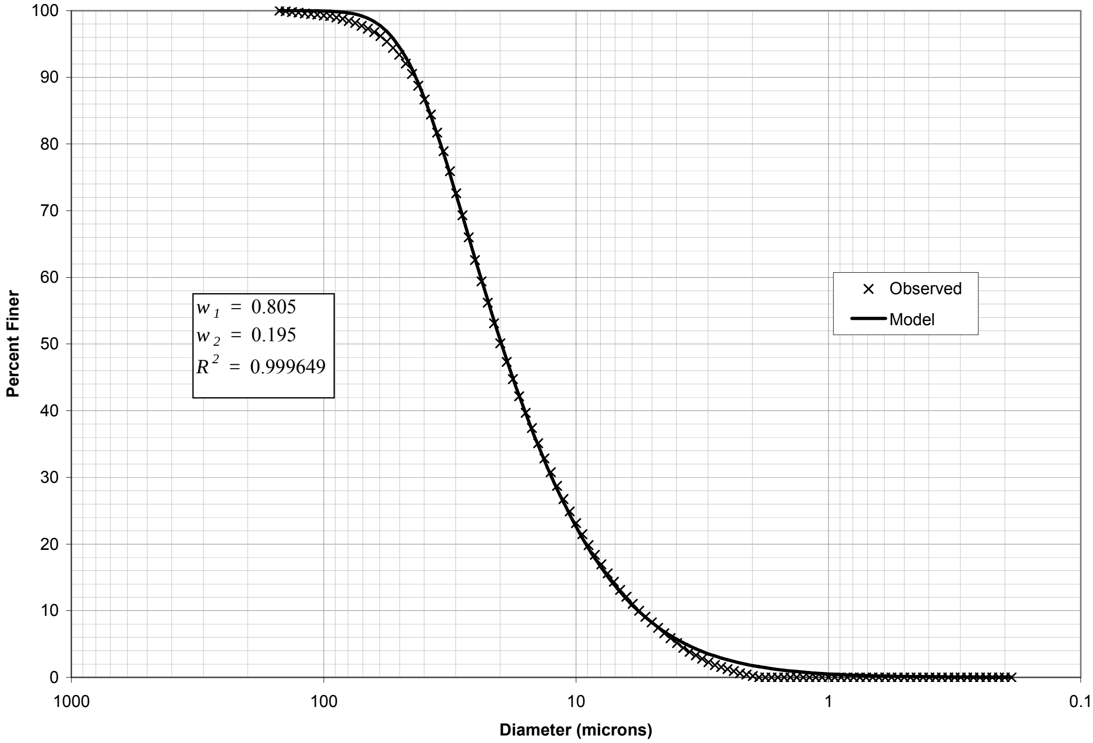

Figure 2 - 34: Bimodal PSD test # 8, replicate # 3.

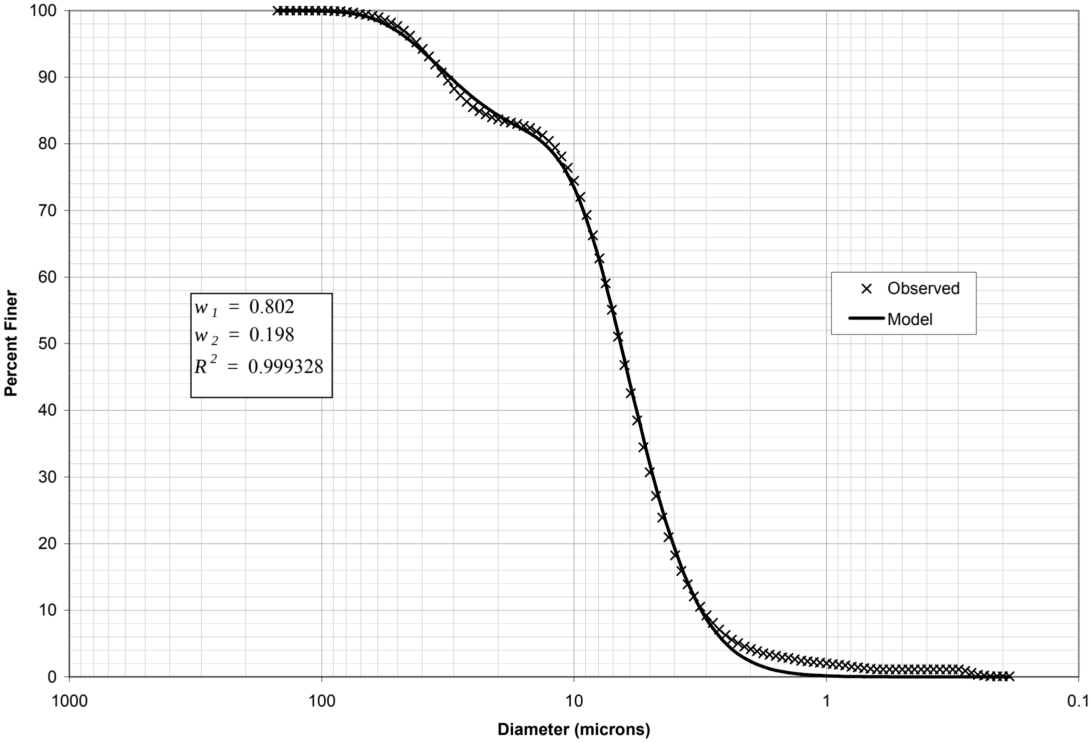

Figure 2 - 35: Bimodal PSD test # 9, replicate # 1.

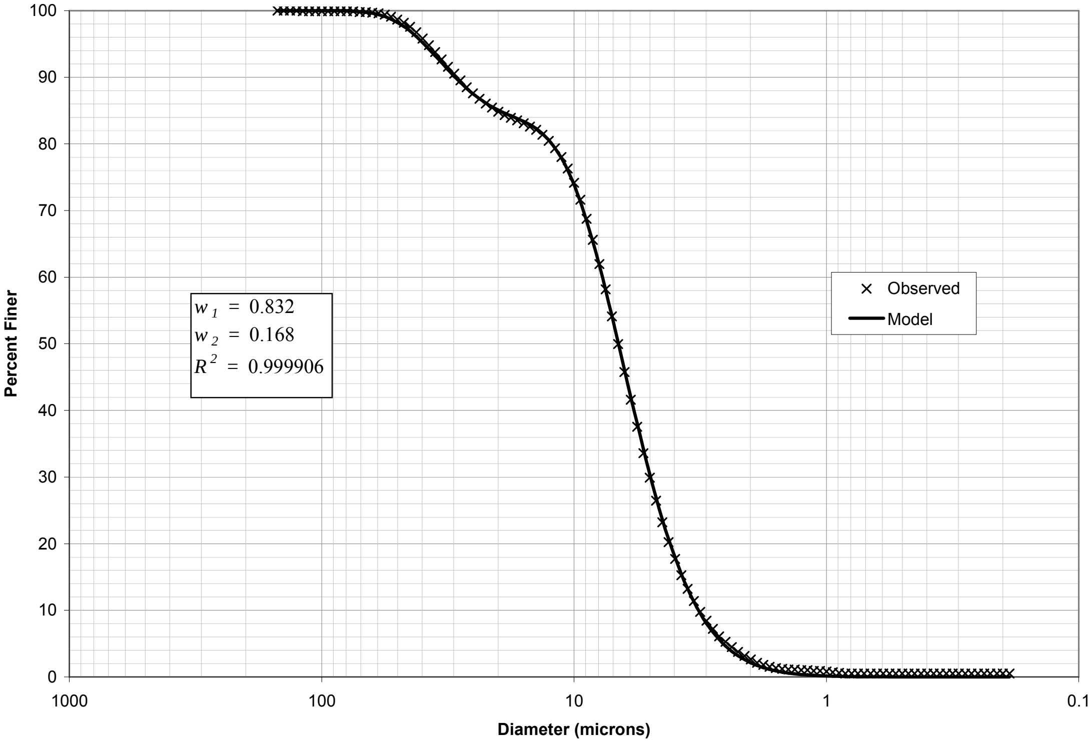

Figure 2 - 36: Bimodal PSD test # 9, replicate # 2.

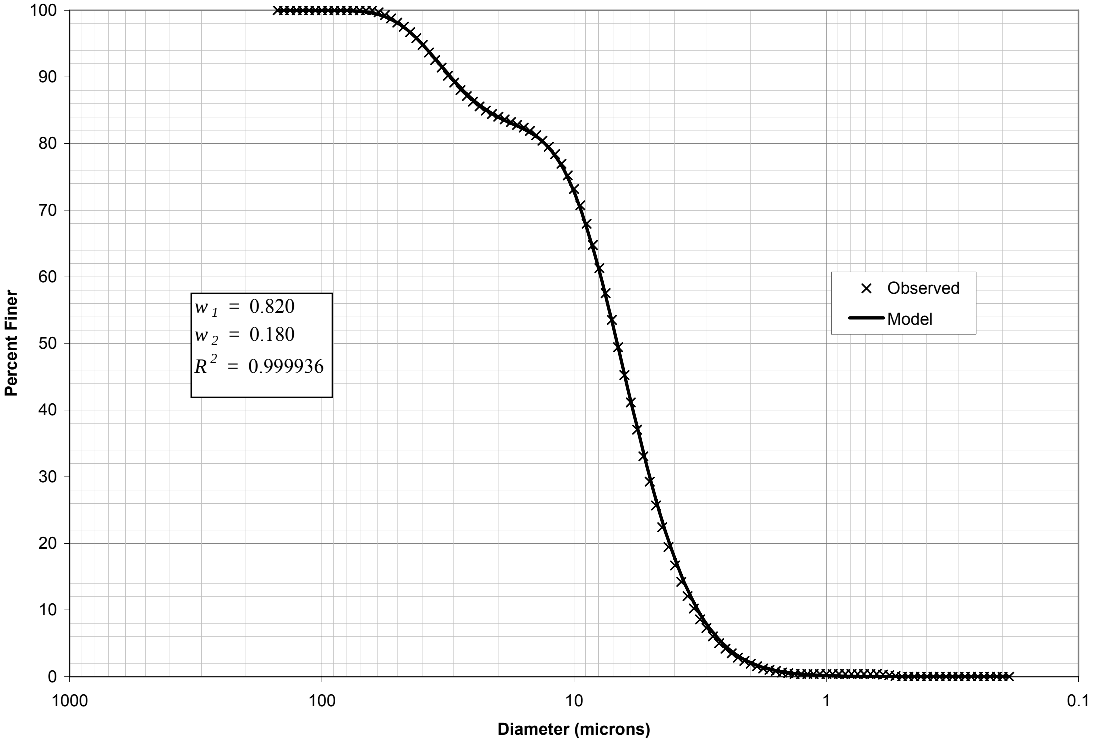

Figure 2 - 37: Bimodal PSD test # 9, replicate # 3.

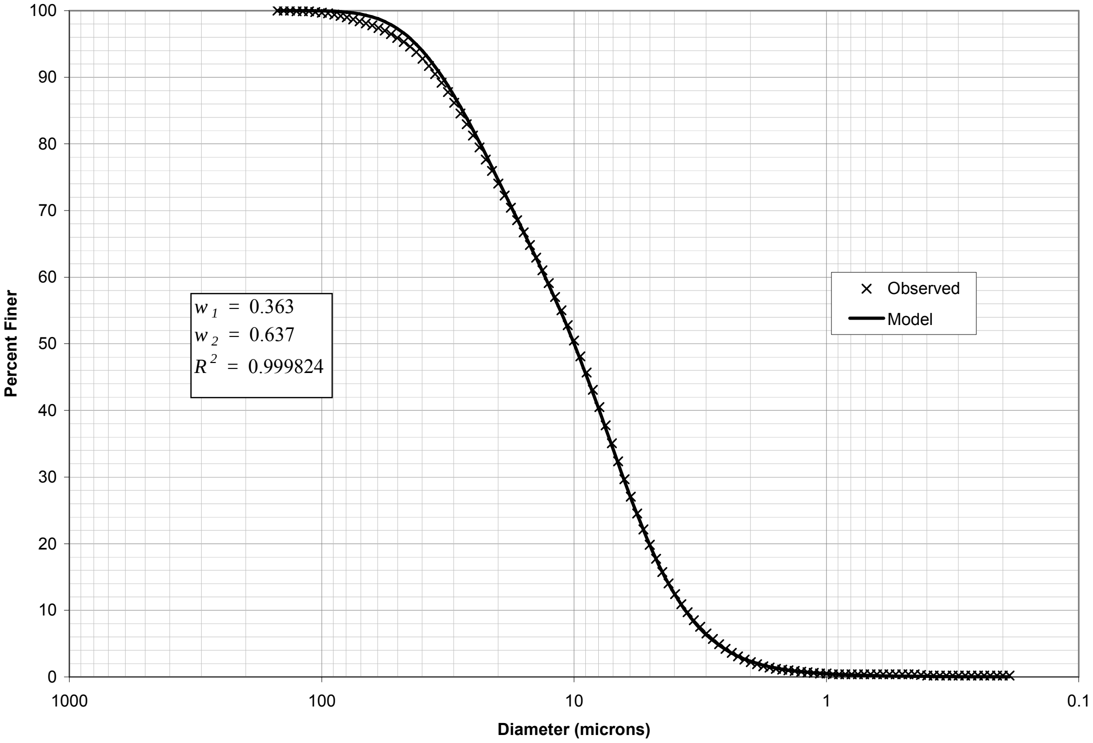

Figure 2 - 38: Bimodal PSD test # 10, replicate # 1.

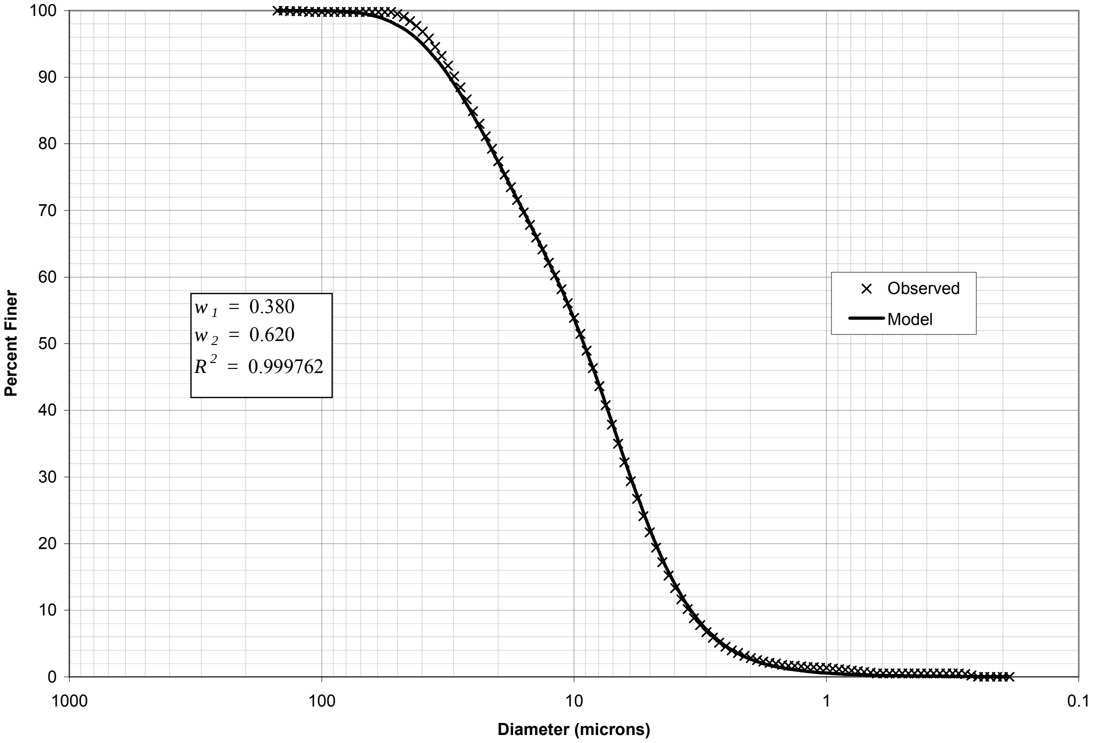

Figure 2 - 39: Bimodal PSD test # 10, replicate # 2.

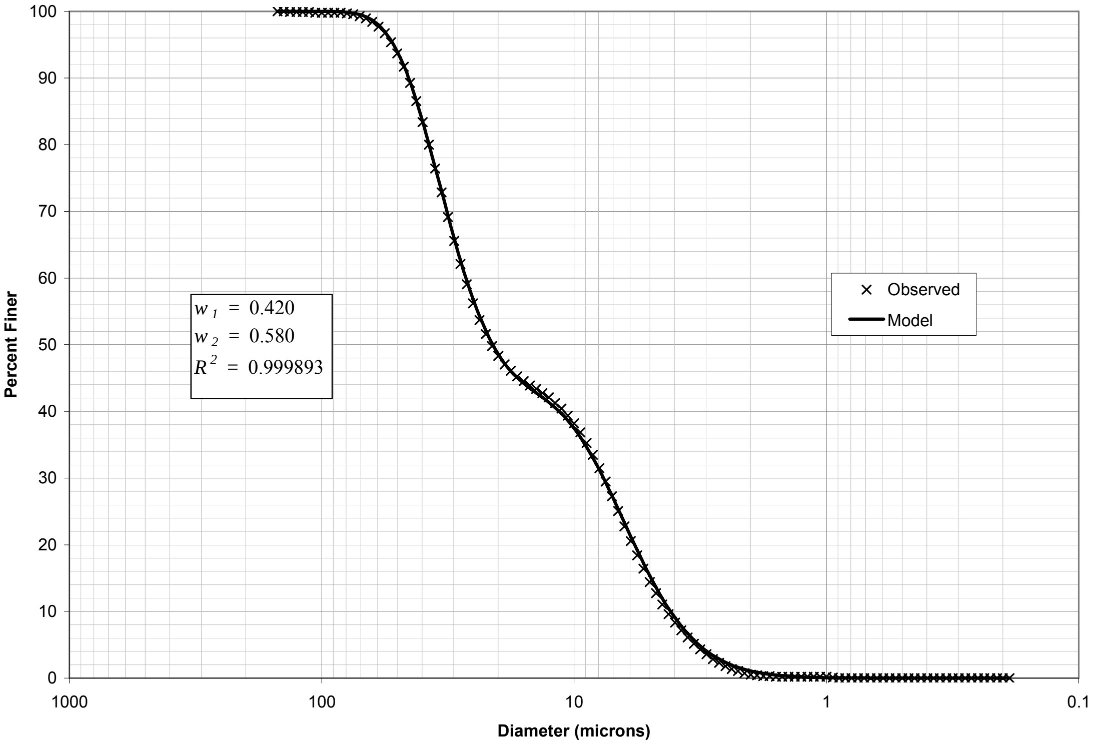

Figure 2 - 40: Bimodal PSD test # 10, replicate # 3.

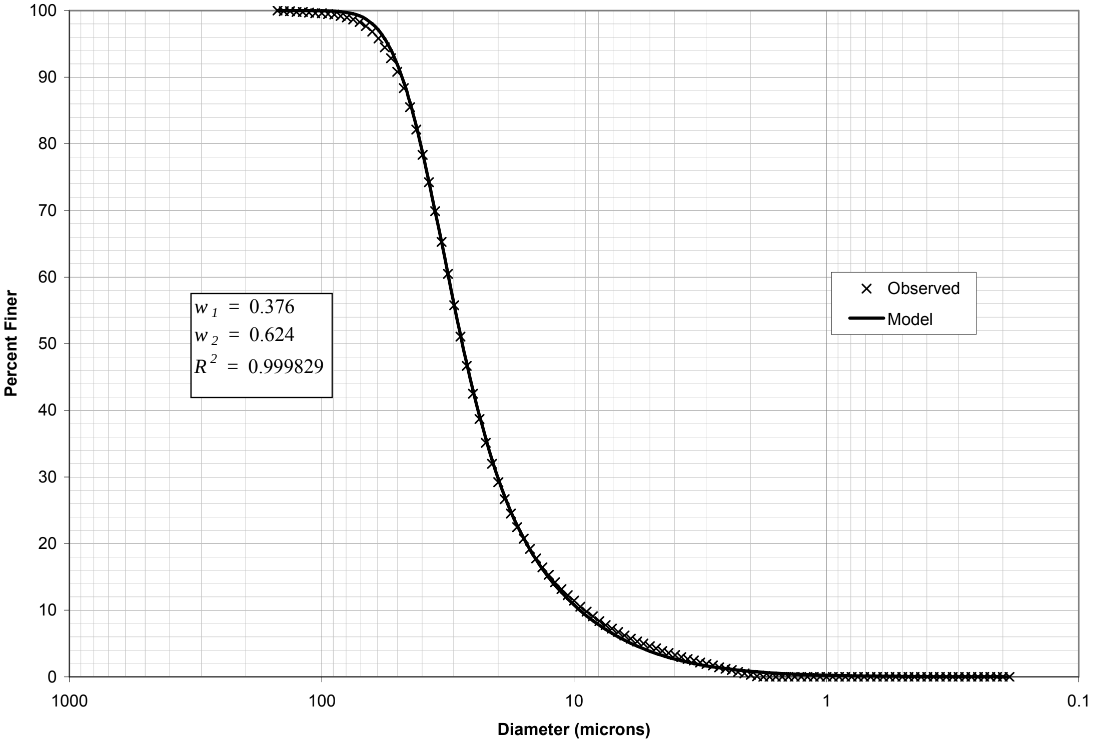

Figure 2 - 41: Bimodal PSD test # 11, replicate # 1.

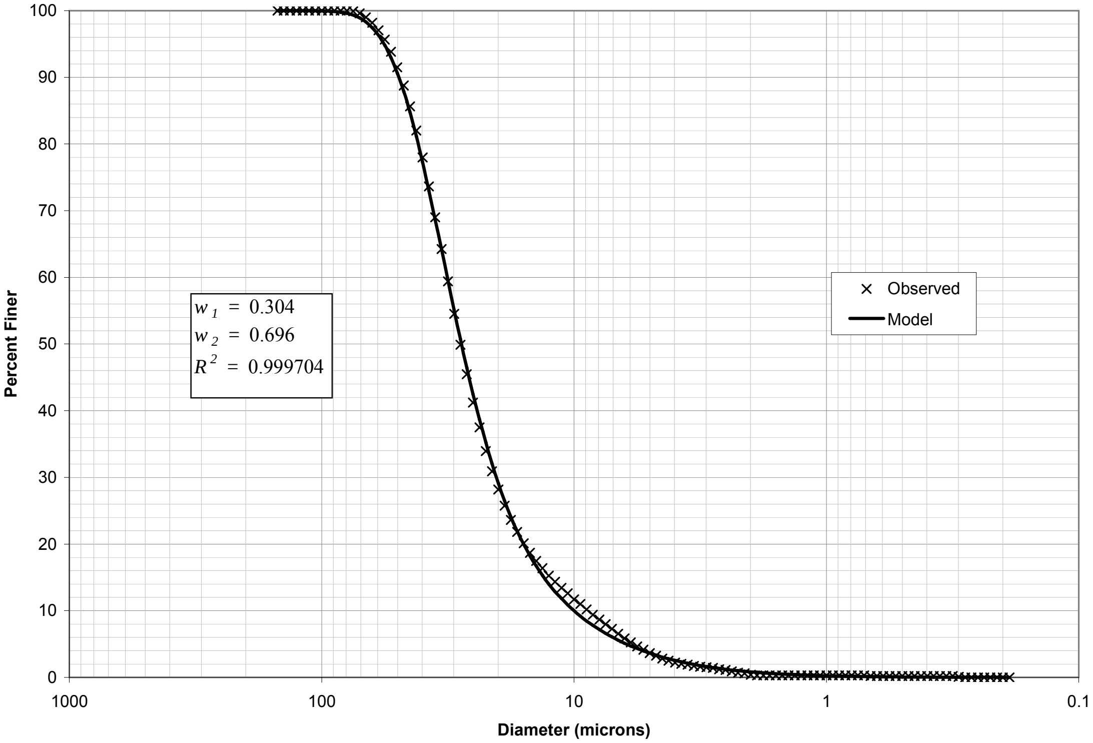

Figure 2 - 42: Bimodal PSD test # 11, replicate # 2.

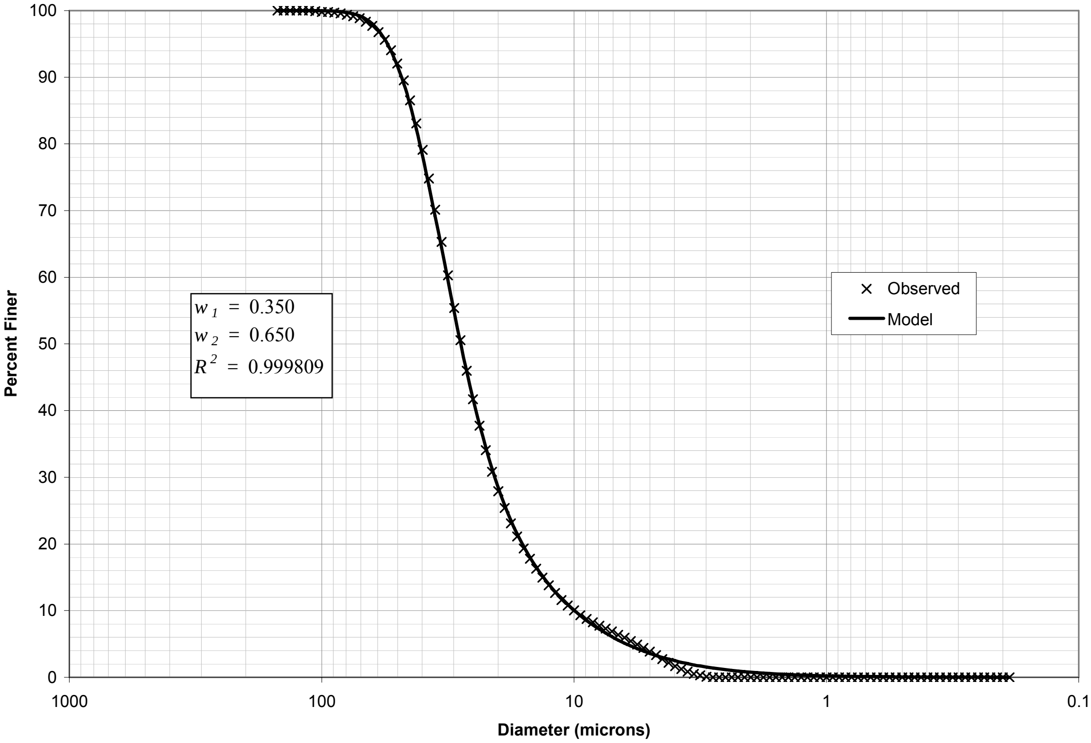

Figure 2 - 43: Bimodal PSD test # 11, replicate # 3.

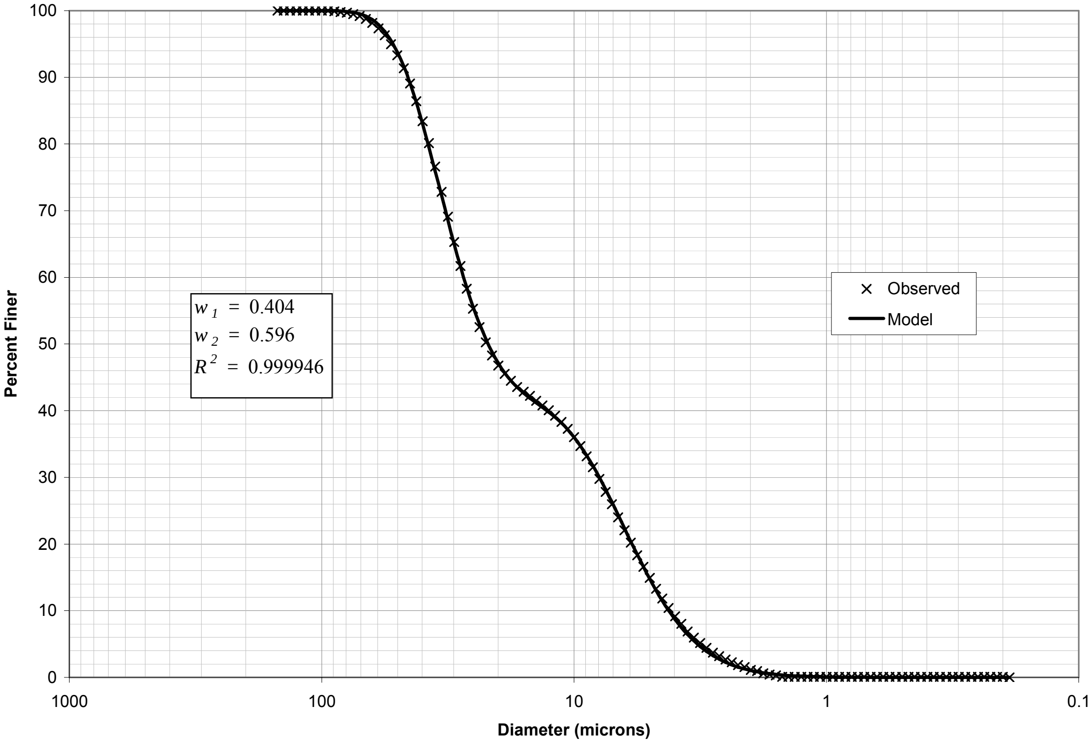

Figure 2 - 44: Bimodal PSD test # 12, replicate # 1.

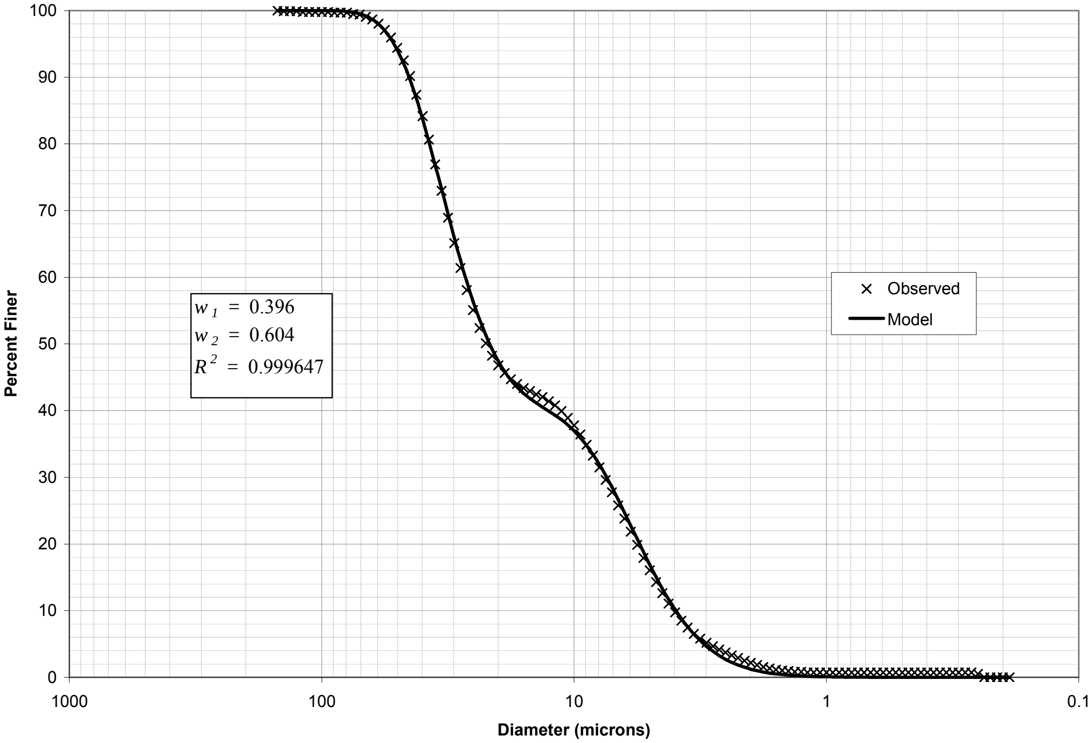

Figure 2 - 45: Bimodal PSD test # 12, replicate # 2.

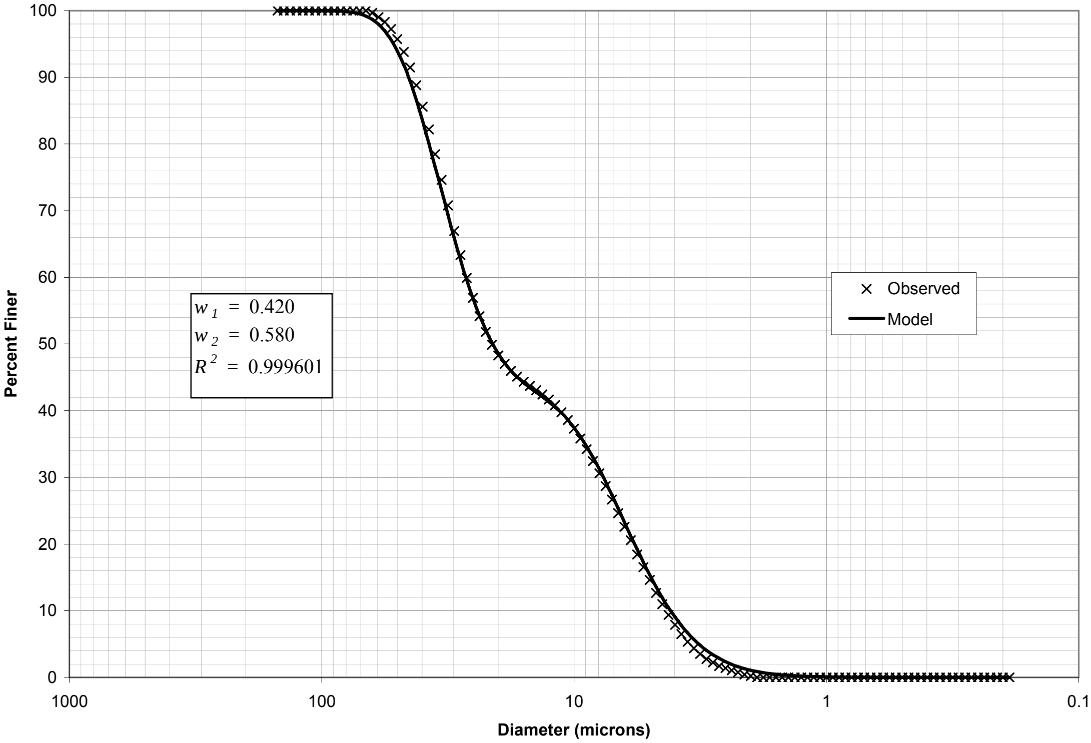

Figure 2 - 46: Bimodal PSD test # 12, replicate # 3.

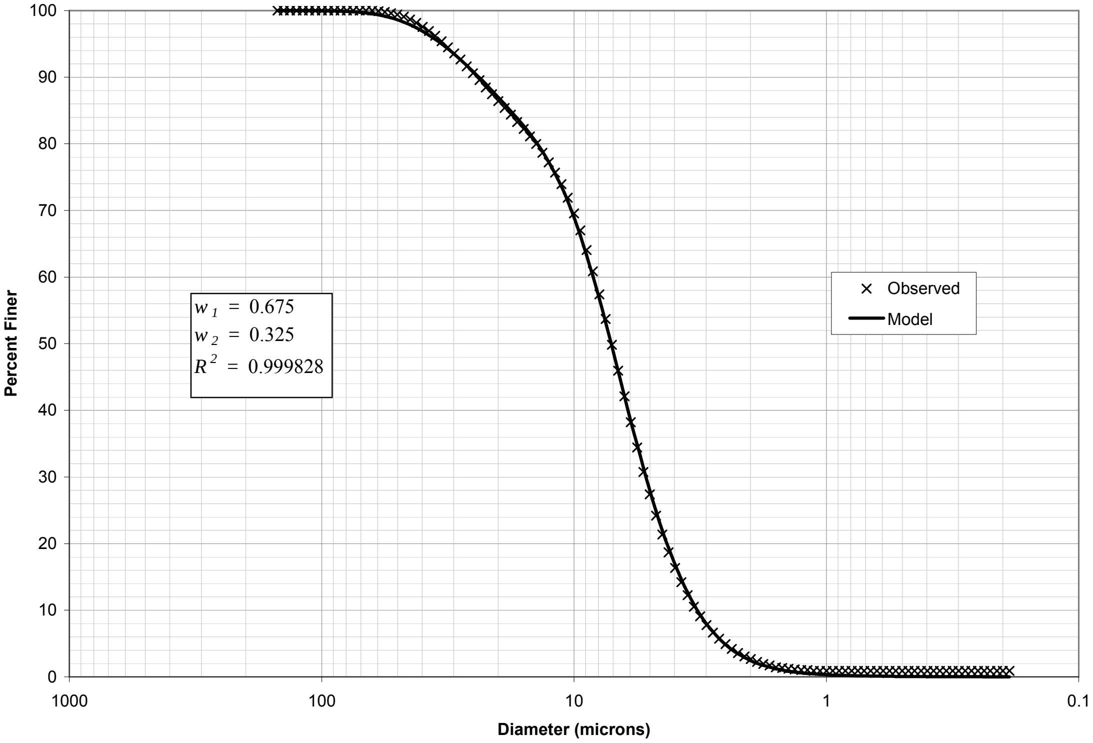

Figure 2 - 47: Bimodal PSD test # 13, replicate # 1.

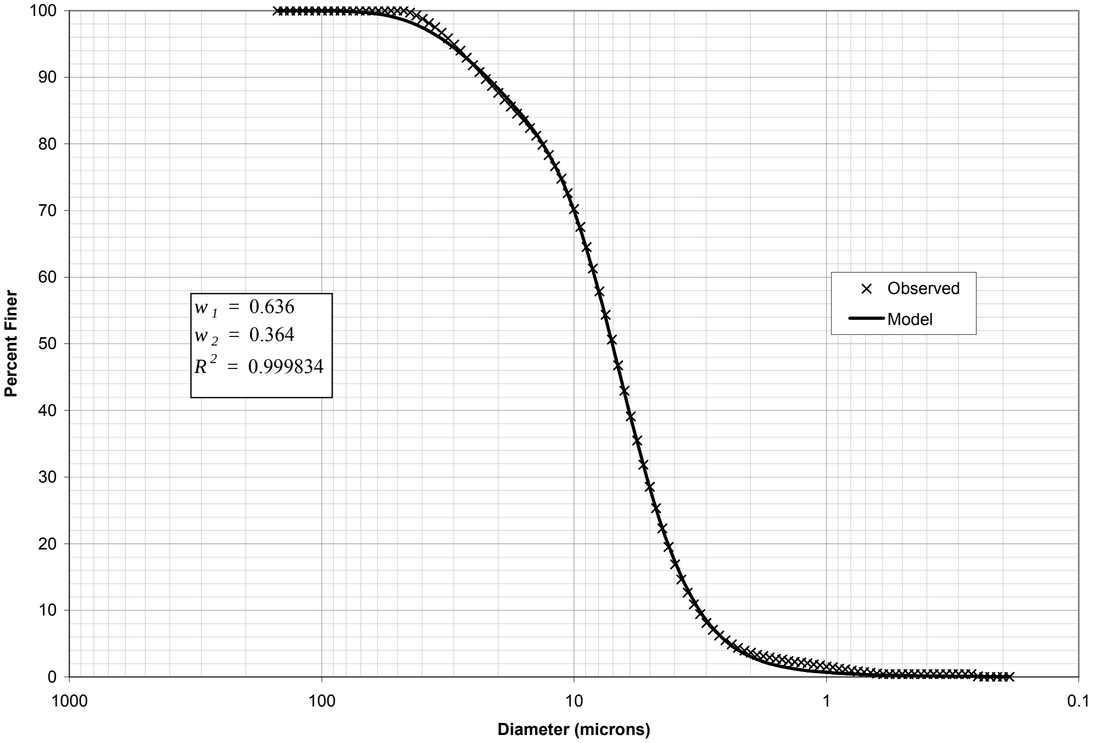

Figure 2 - 48: Bimodal PSD test # 13, replicate # 2.

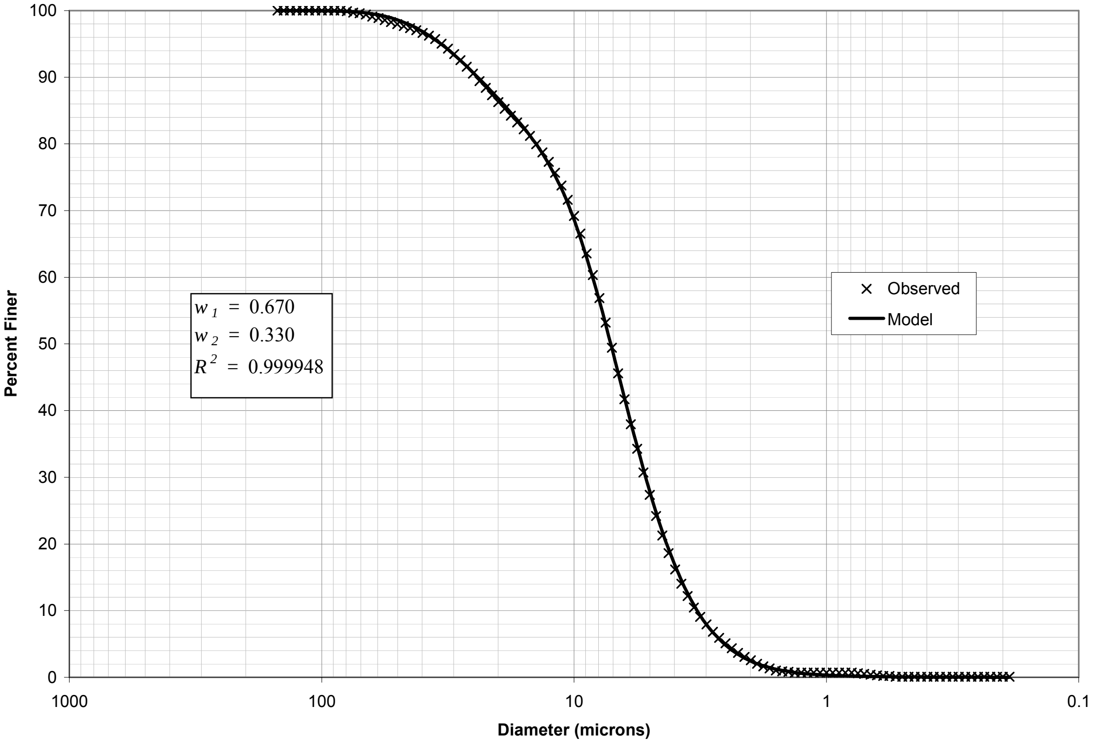

Figure 2 - 49: Bimodal PSD test # 13, replicate # 3.

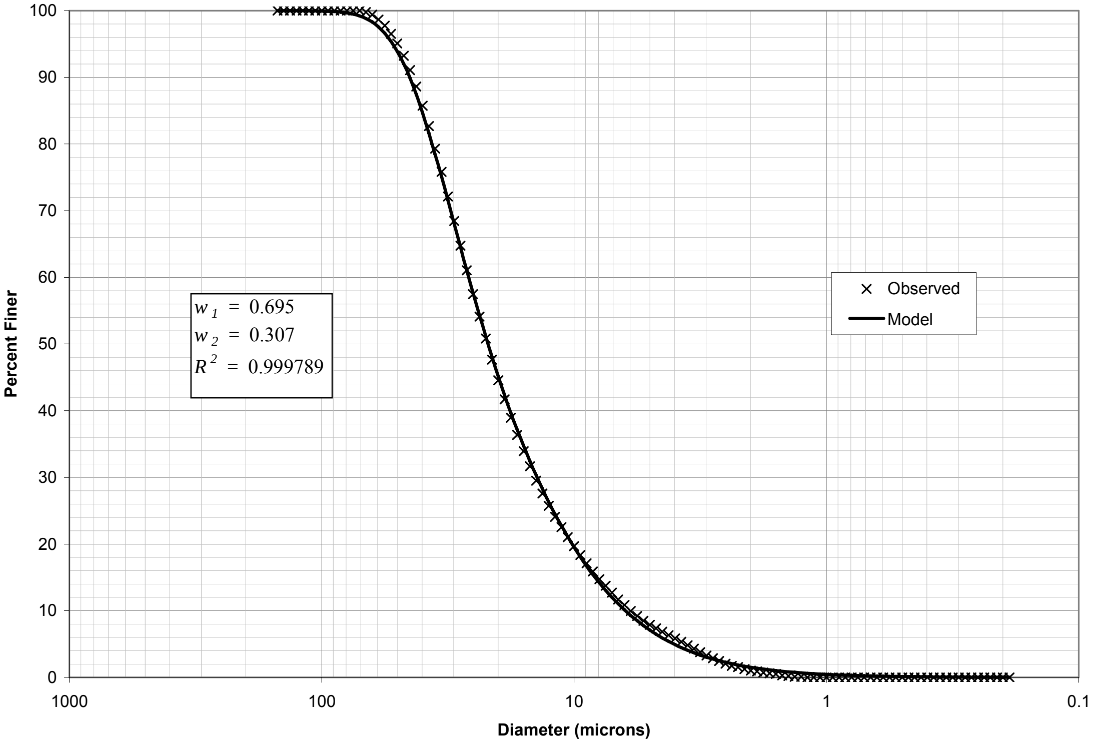

Figure 2 - 50: Bimodal PSD test # 14, replicate # 1.

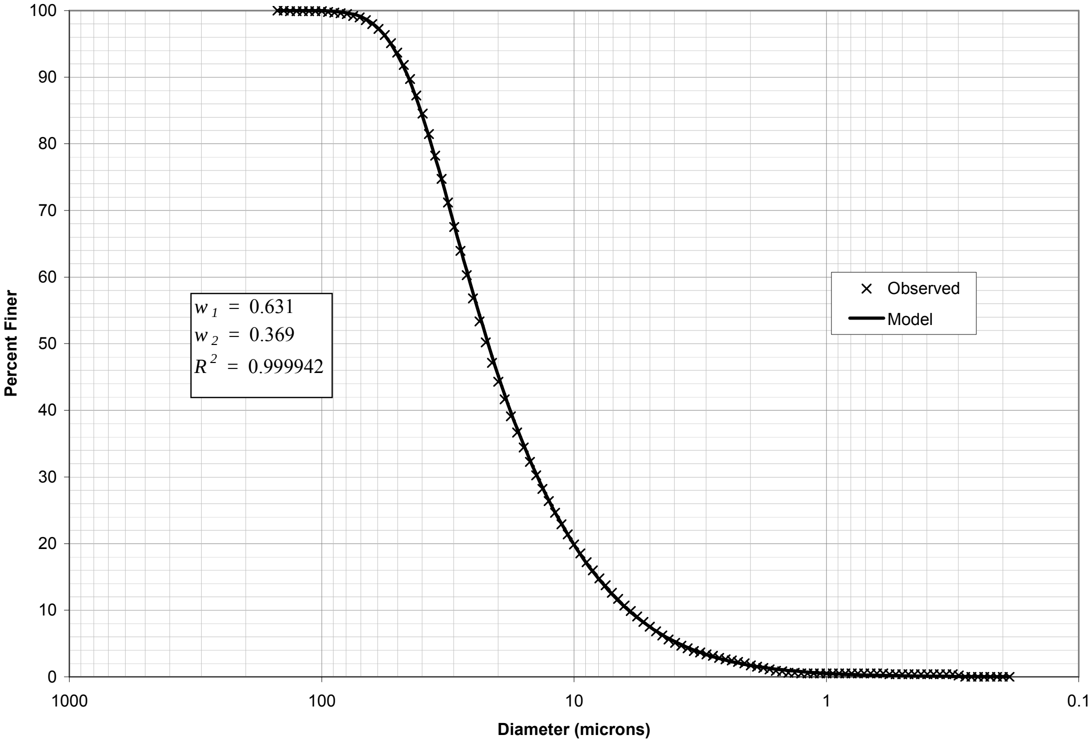

Figure 2 - 51: Bimodal PSD test # 14, replicate # 2.

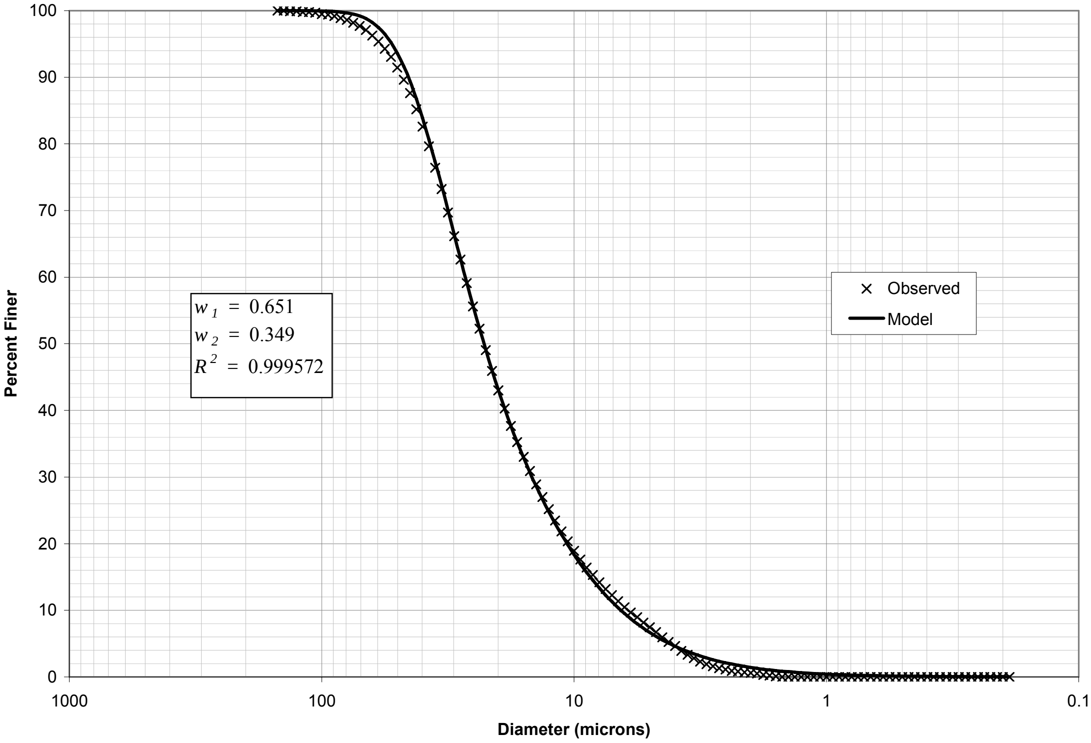

Figure 2 - 52: Bimodal PSD test # 14, replicate # 3.

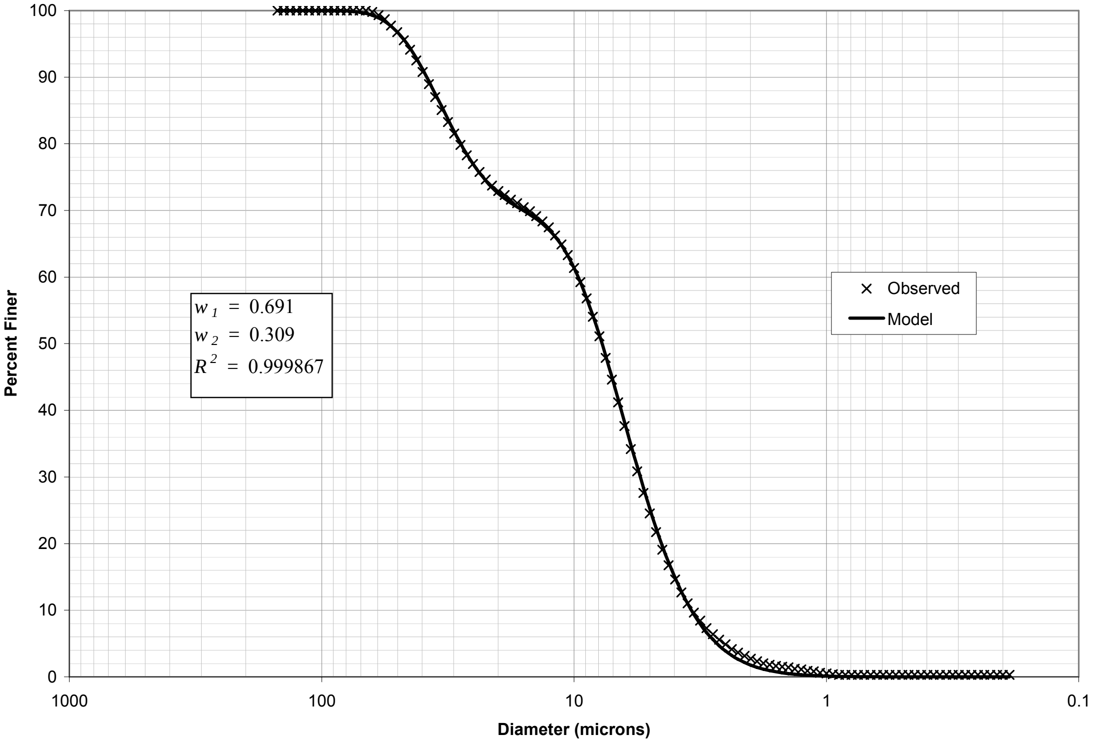

Figure 2 - 53: Bimodal PSD test # 15, replicate # 1.

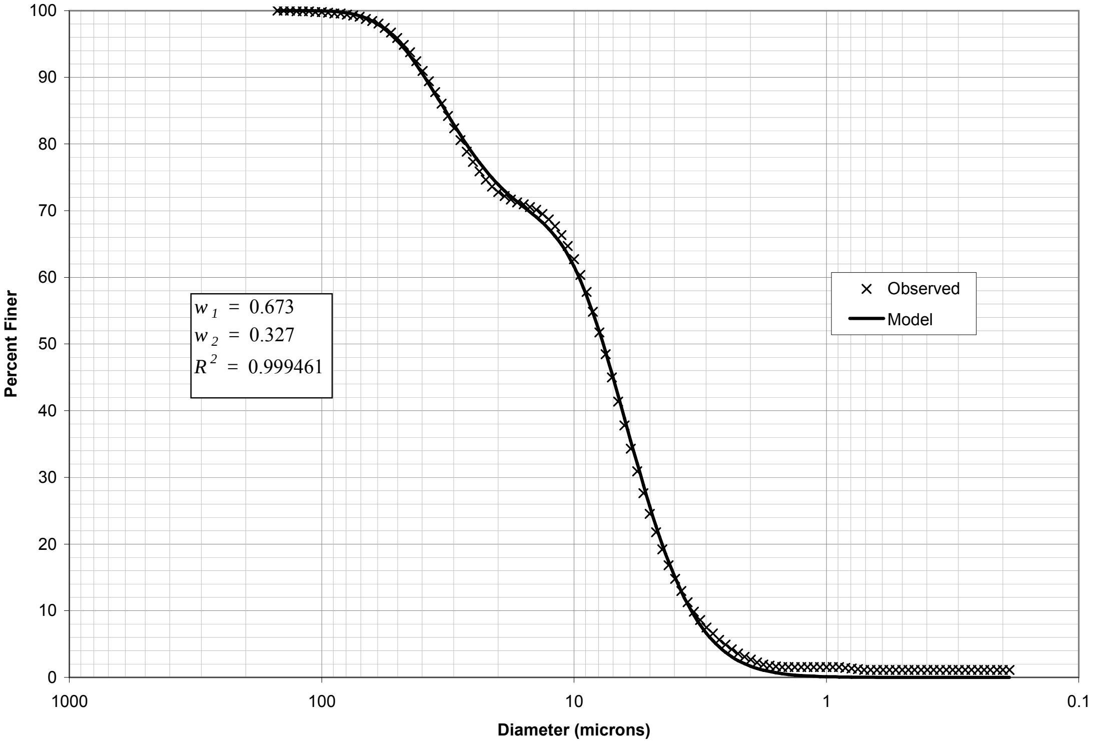

Figure 2 - 54: Bimodal PSD test # 15, replicate # 2.

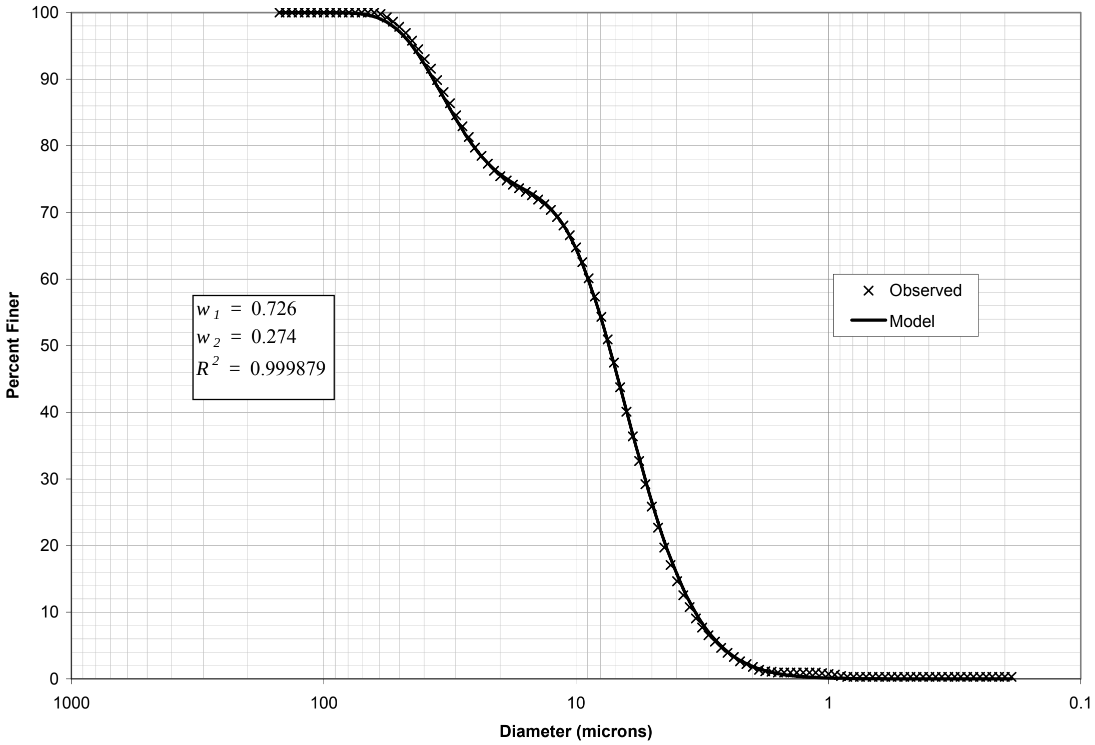

Figure 2 - 55: Bimodal PSD test # 15, replicate # 3.

APPENDIX D: Trimodal PSD Test Results

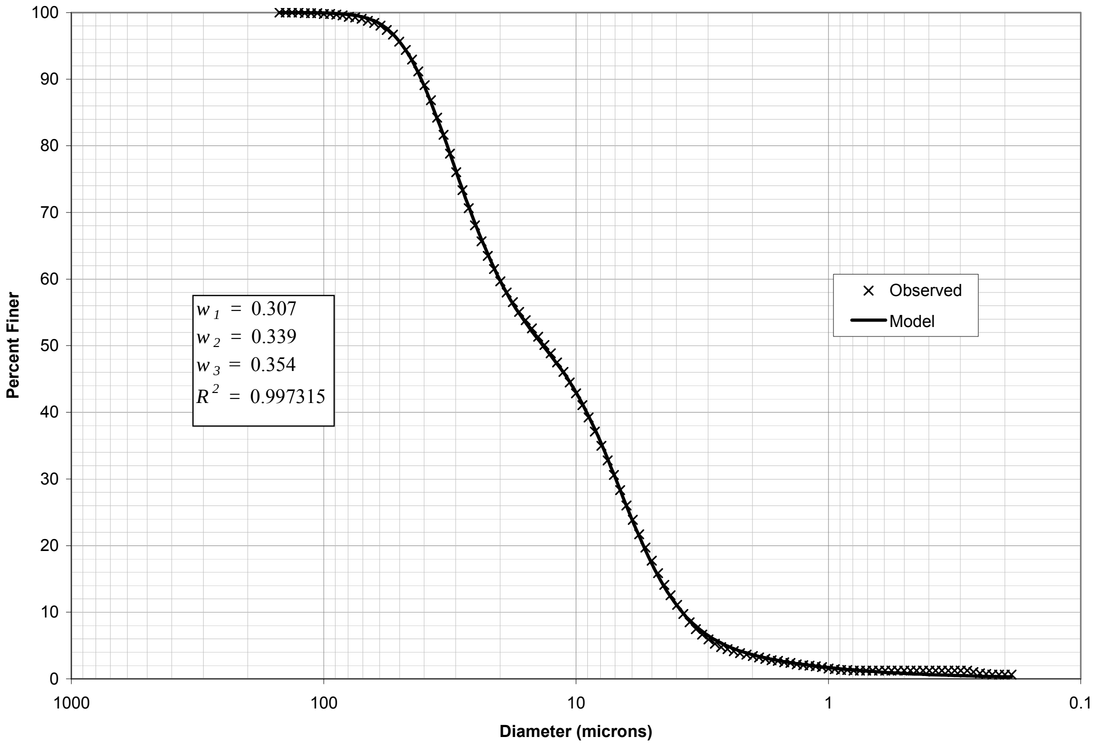

Figure 2 - 56: Trimodal PSD test # 1, replicate # 1.

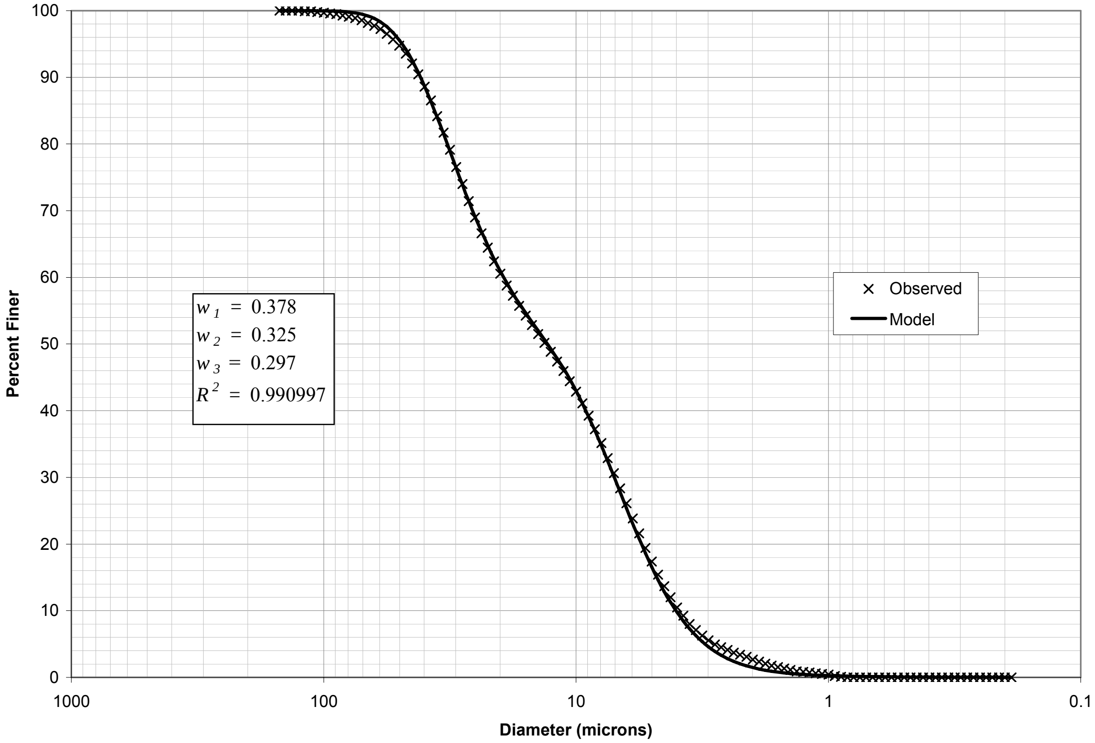

Figure 2 - 57: Trimodal PSD test # 1, replicate # 2.

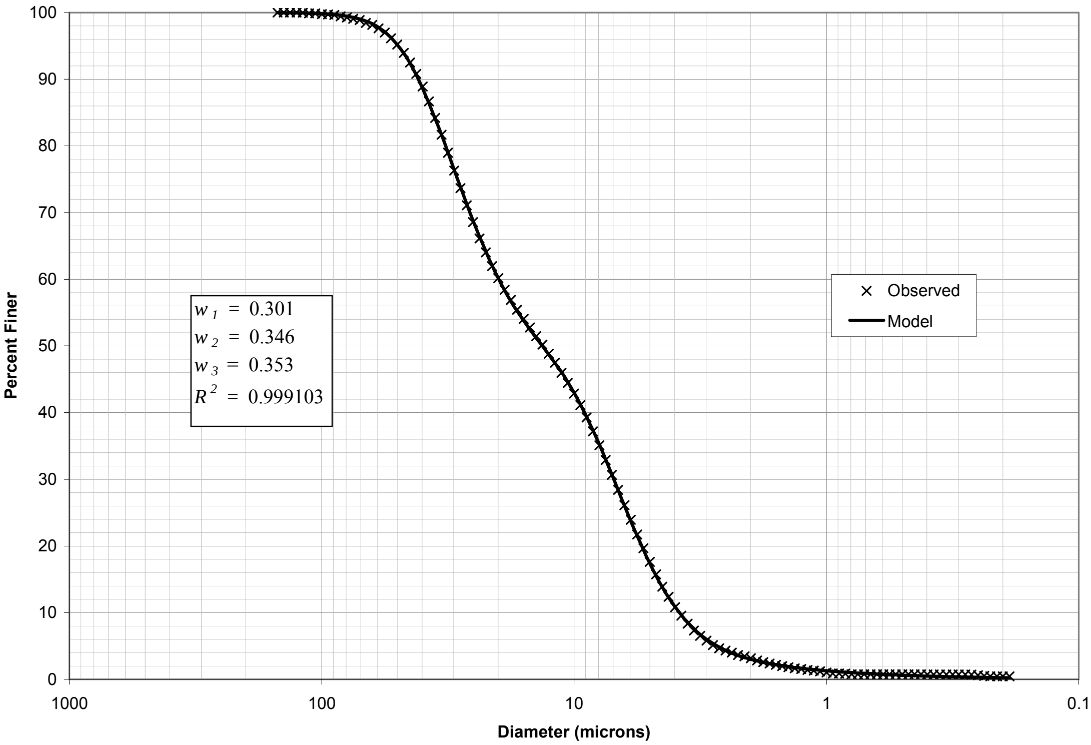

Figure 2 - 58: Trimodal PSD test # 1, replicate # 3.

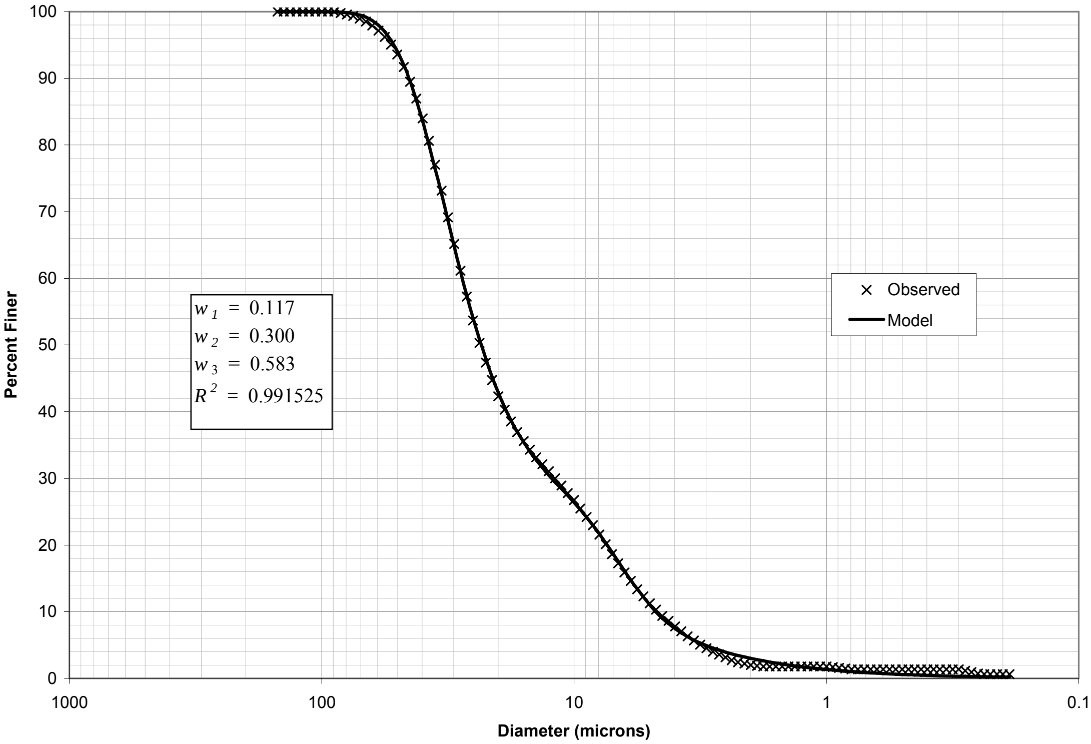

Figure 2 - 59: Trimodal PSD test # 2, replicate # 1.

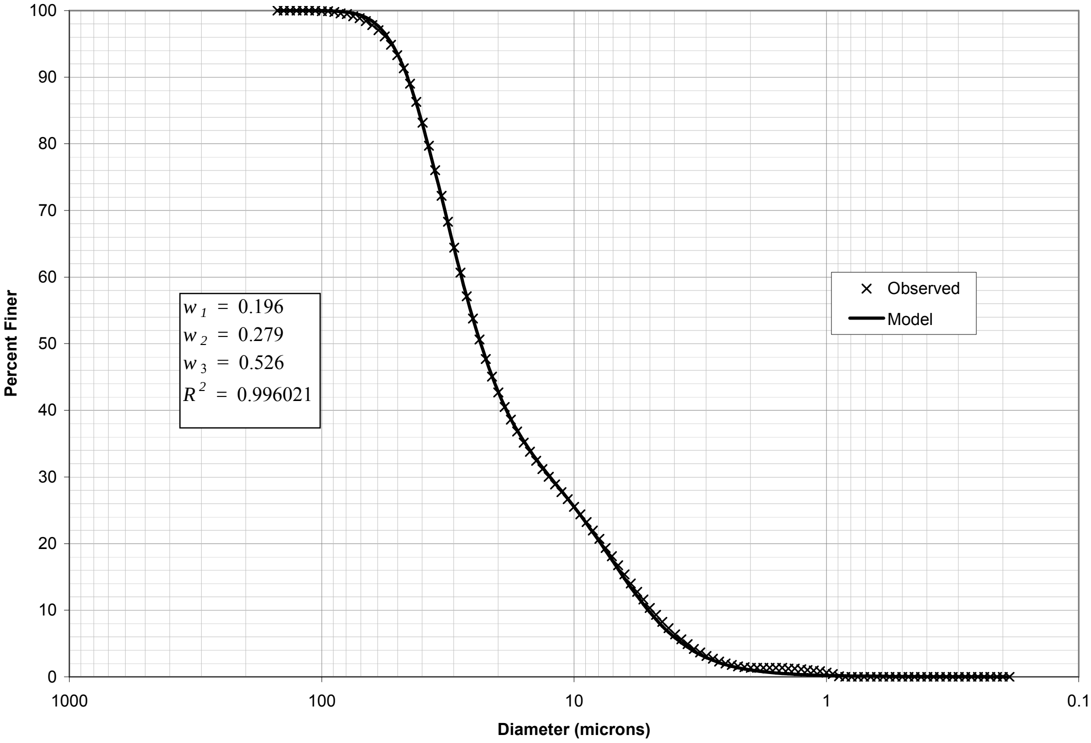

Figure 2 - 60: Trimodal PSD test # 2, replicate # 2.

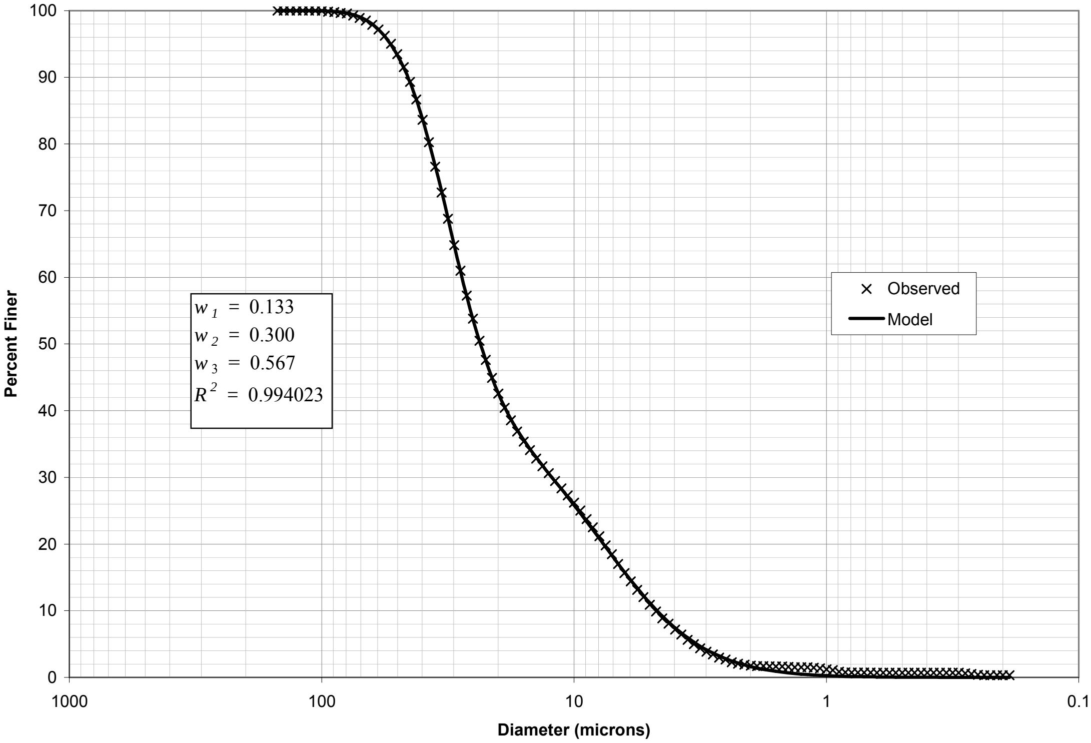

Figure 2 - 61: Trimodal PSD test # 2, replicate # 3.

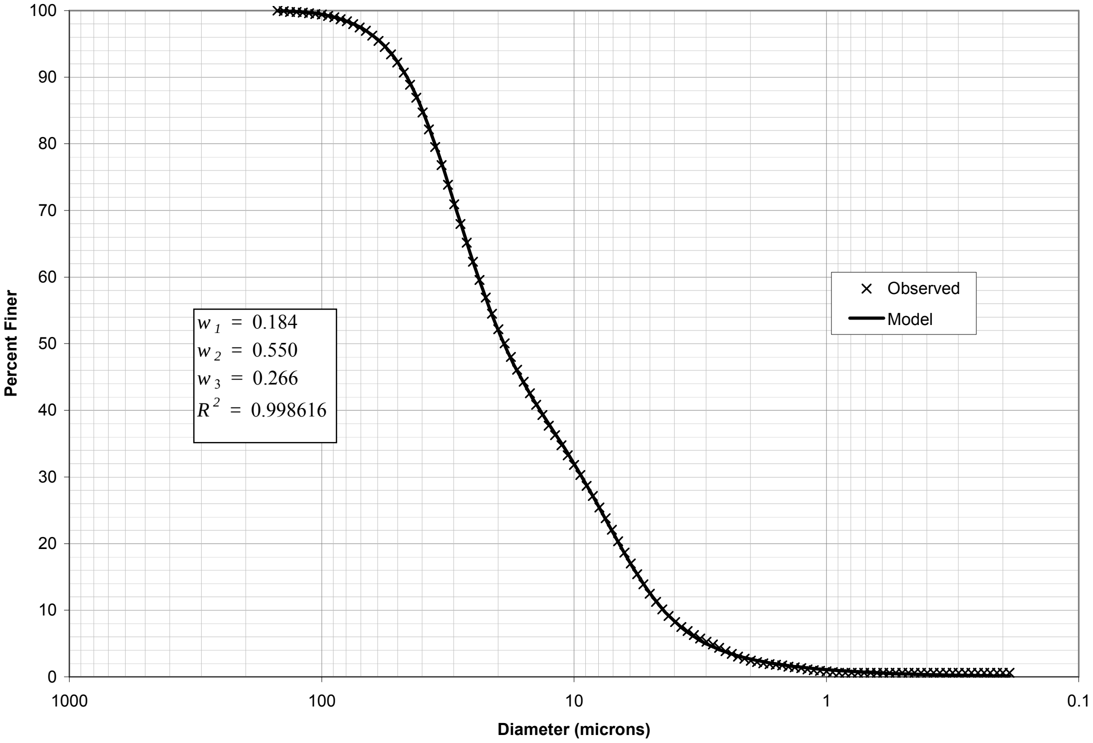

Figure 2 - 62: Trimodal PSD test # 3, replicate # 1.

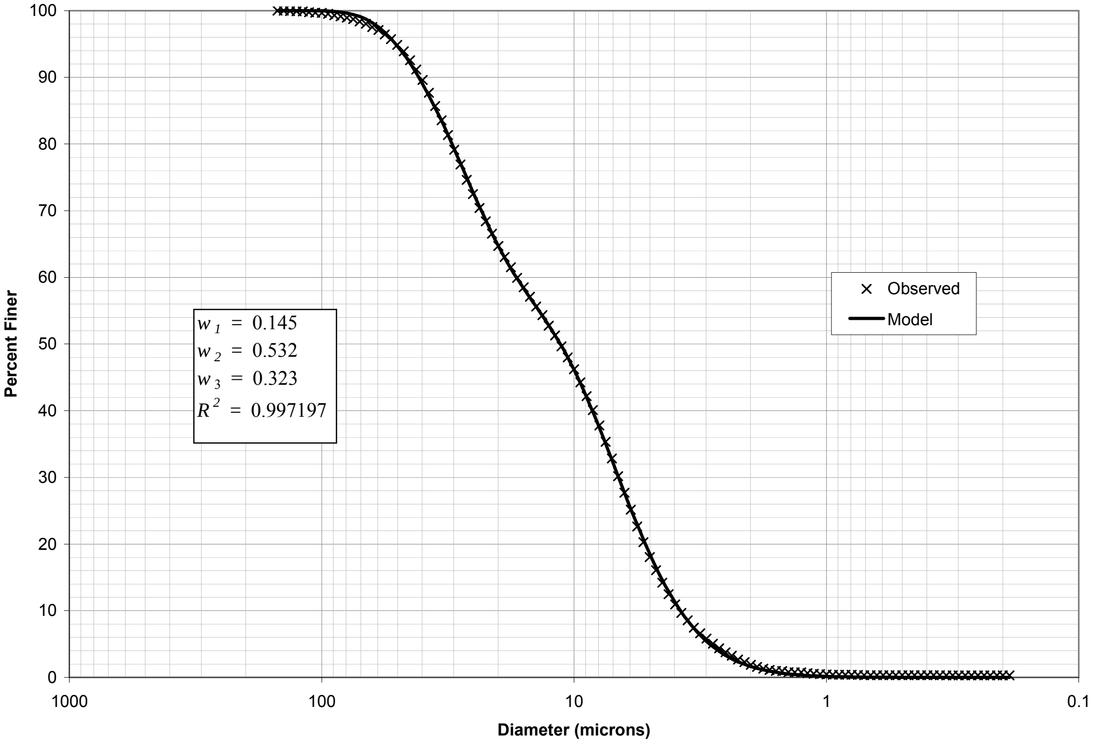

Figure 2 - 63: Trimodal PSD test # 3, replicate # 2.

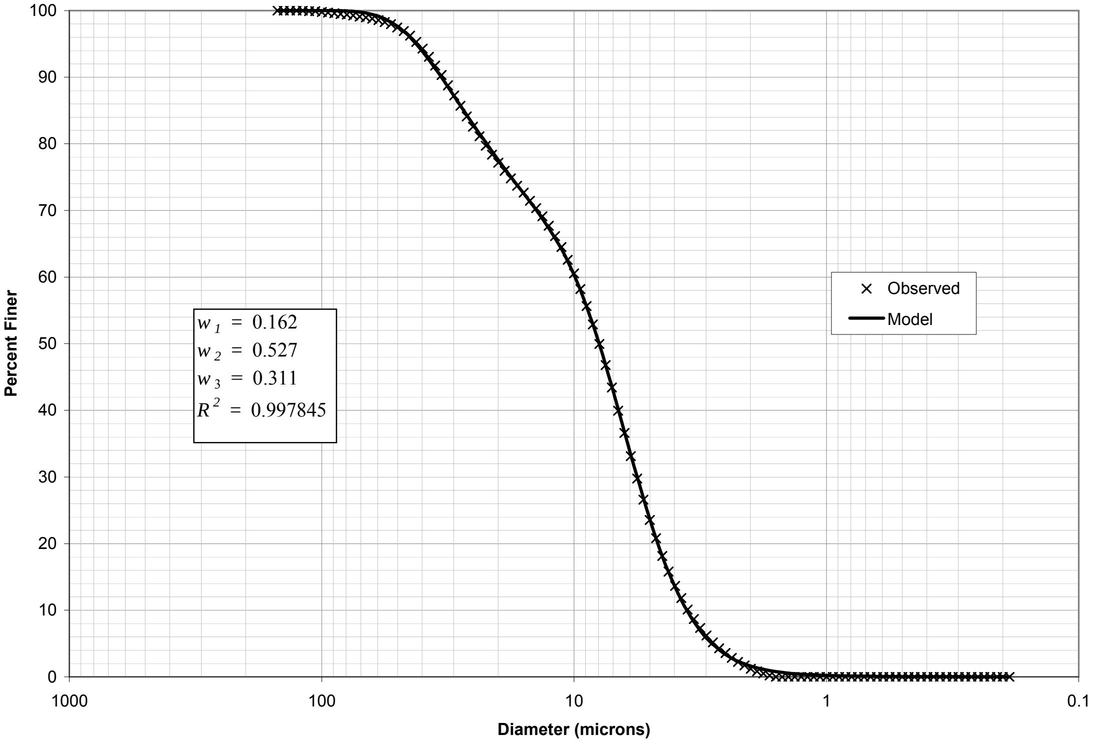

Figure 2 - 64: Trimodal PSD test # 3, replicate # 3.

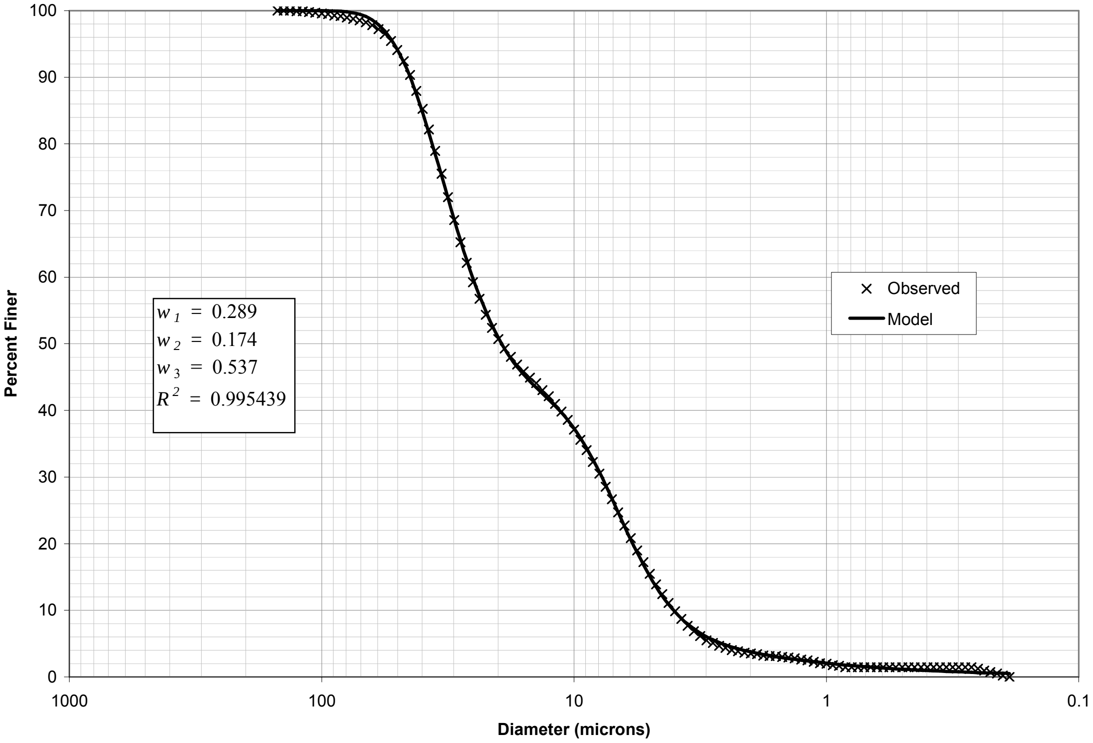

Figure 2 - 65: Trimodal PSD test # 4, replicate # 1.

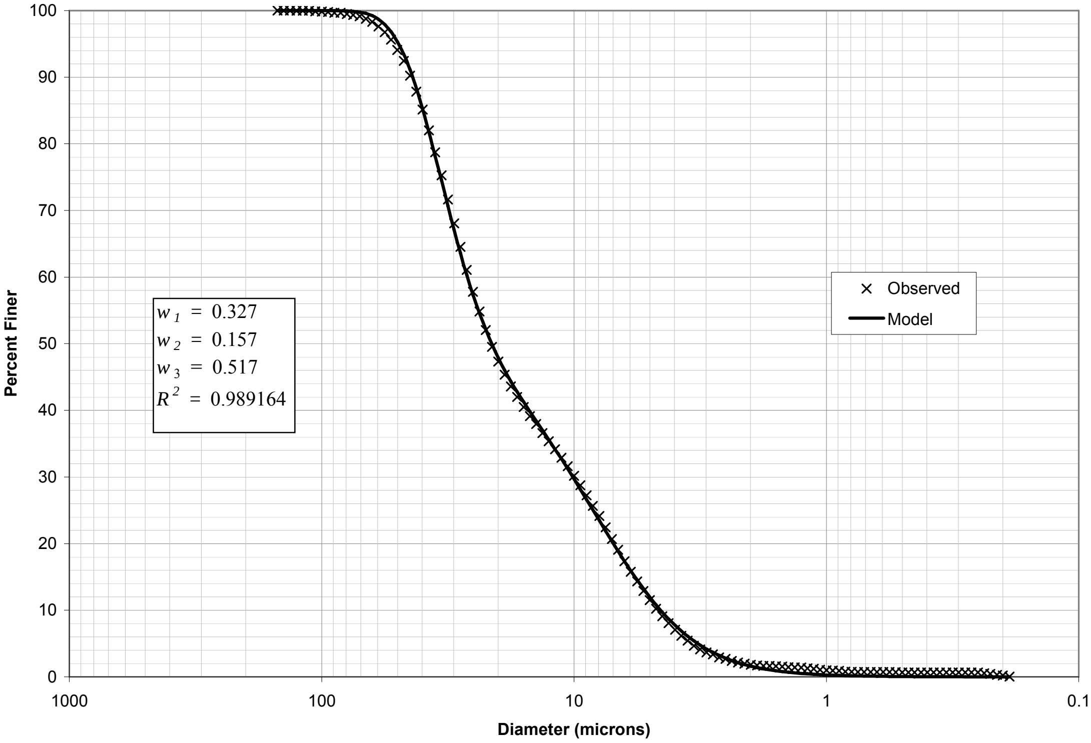

Figure 2 - 66: Trimodal PSD test # 4, replicate # 2.

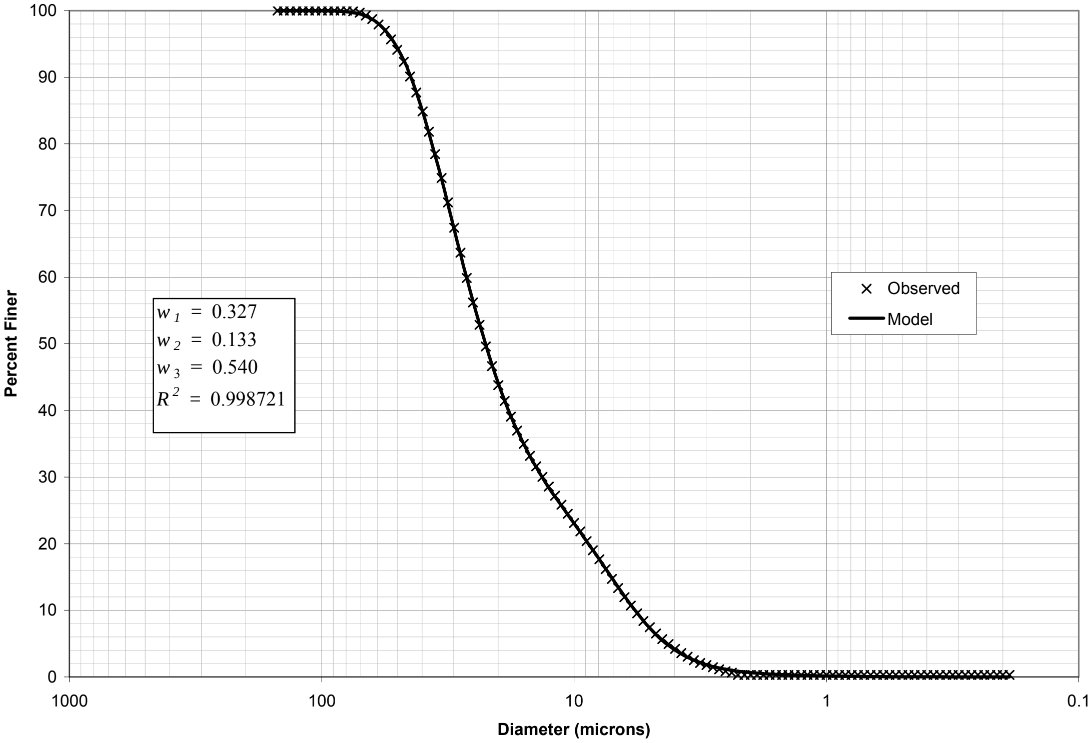

Figure 2 - 67: Trimodal PSD test # 4, replicate # 3.

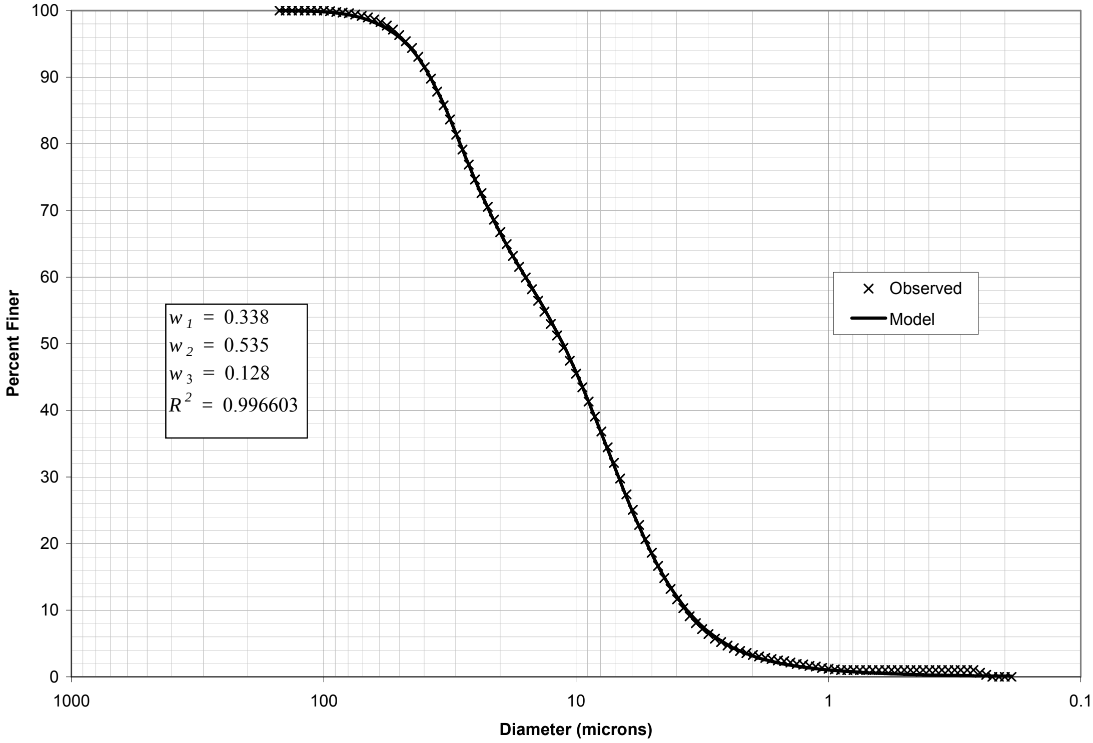

Figure 2 - 68: Trimodal PSD test # 5, replicate # 1.

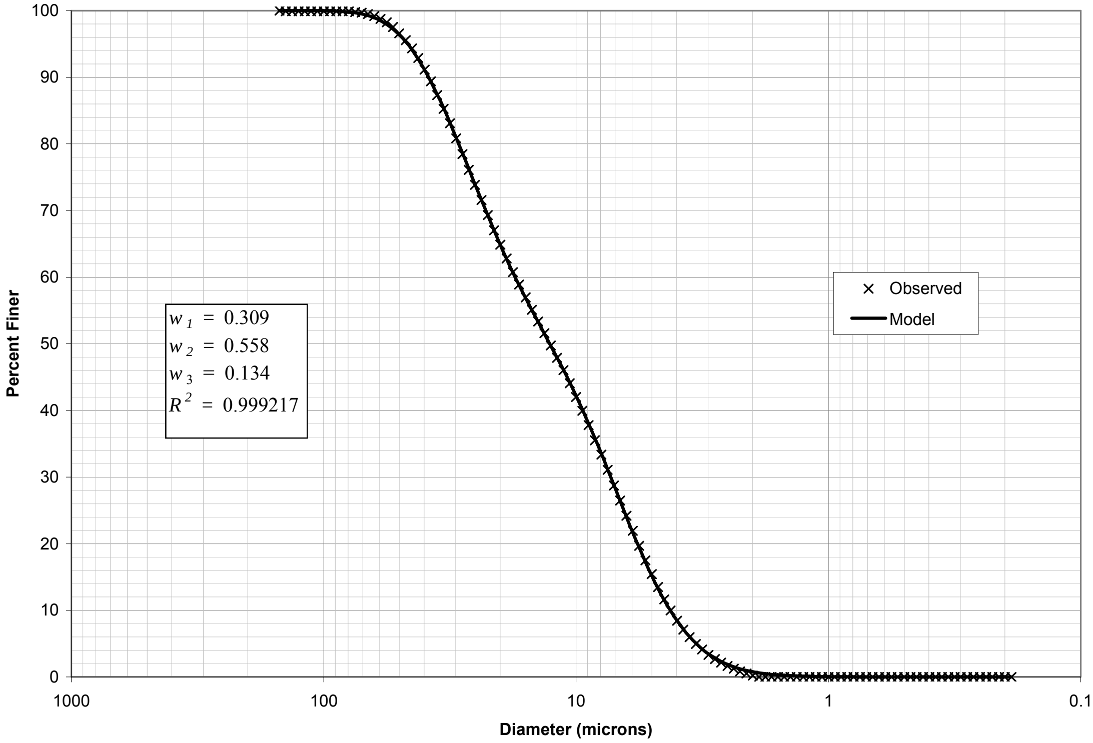

Figure 2 - 69: Trimodal PSD test # 5, replicate # 2.

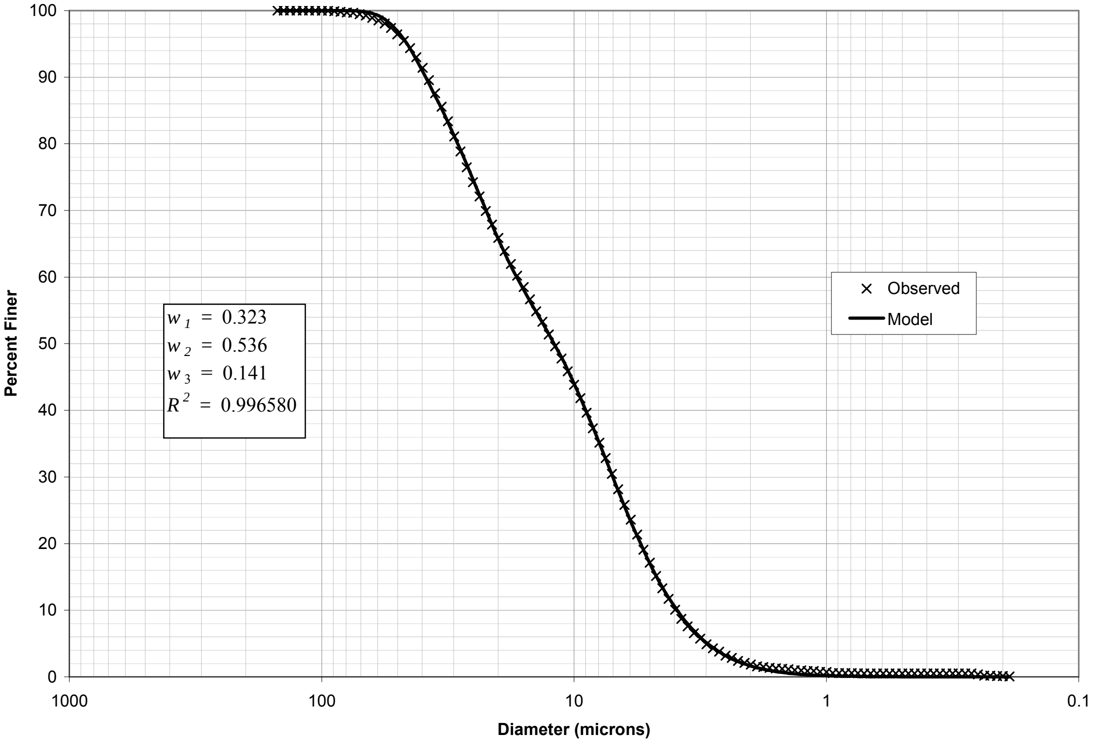

Figure 2 - 70: Trimodal PSD test # 5, replicate # 3.

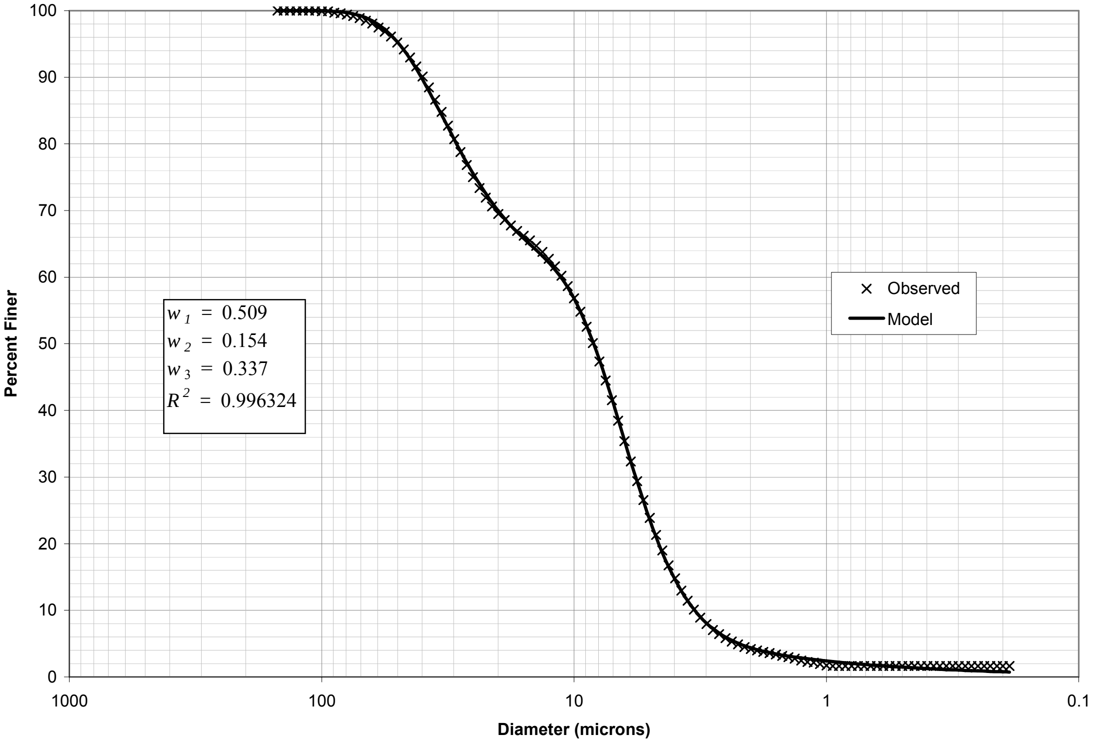

Figure 2 - 71: Trimodal PSD test # 6, replicate # 1.

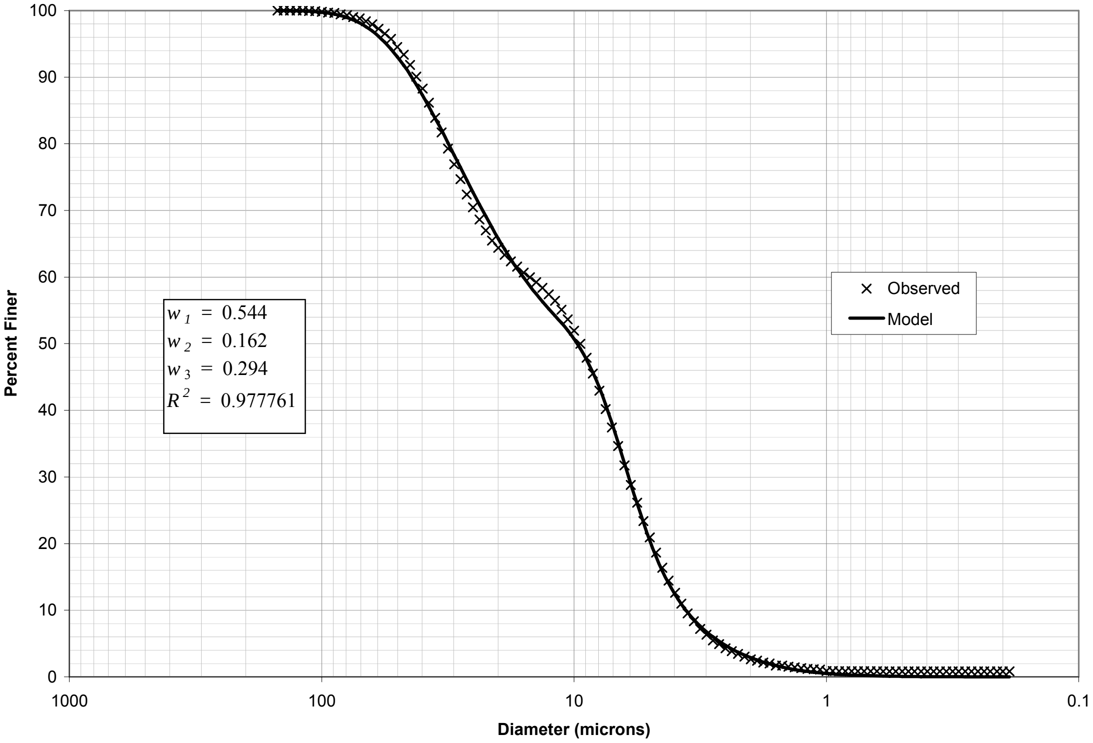

Figure 2 - 72: Trimodal PSD test # 6, replicate # 2.

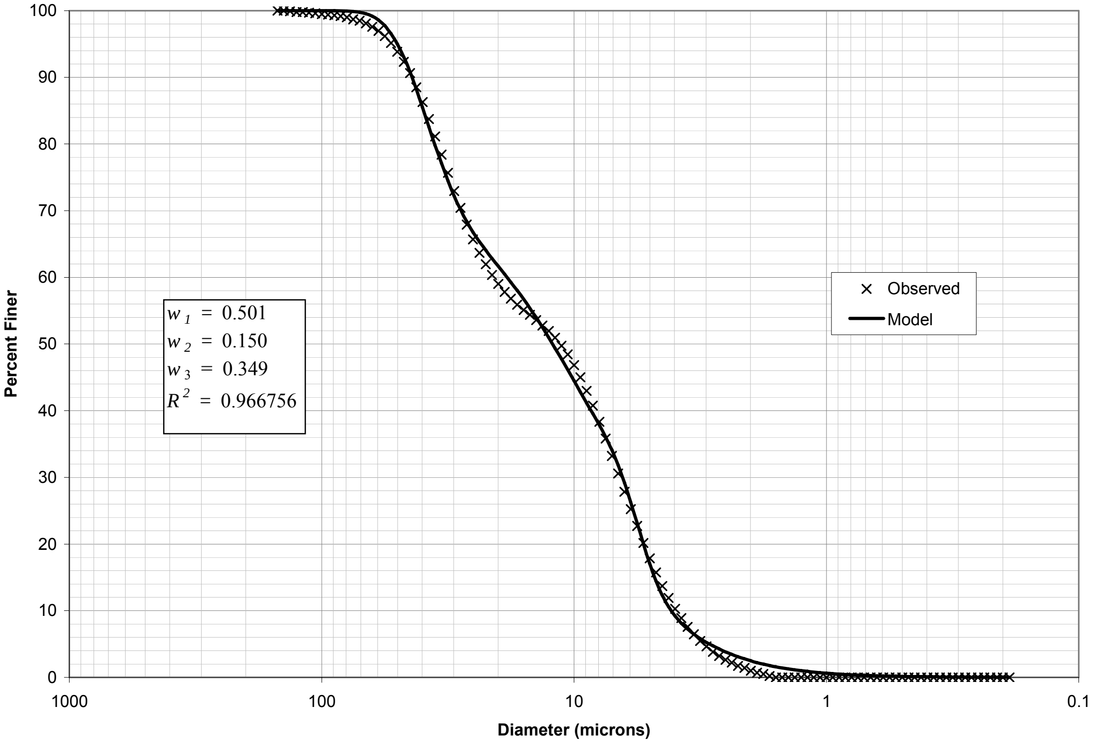

Figure 2 - 73: Trimodal PSD test # 7, replicate # 3.

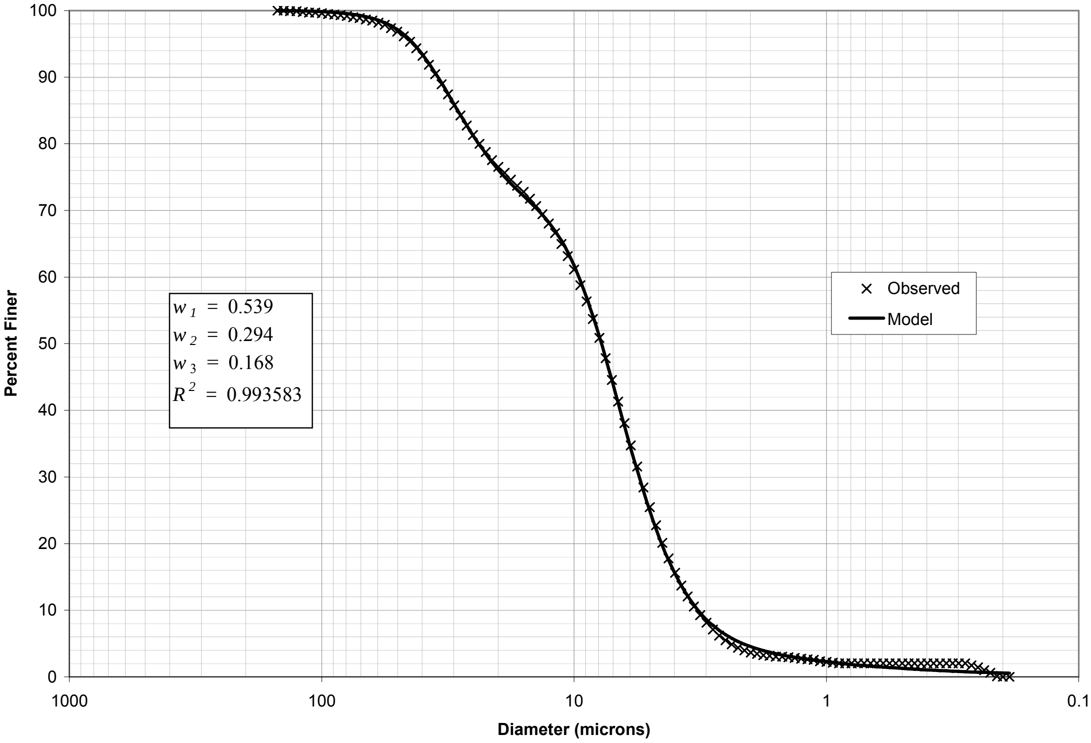

Figure 2 - 74: Trimodal PSD test # 7, replicate # 1.

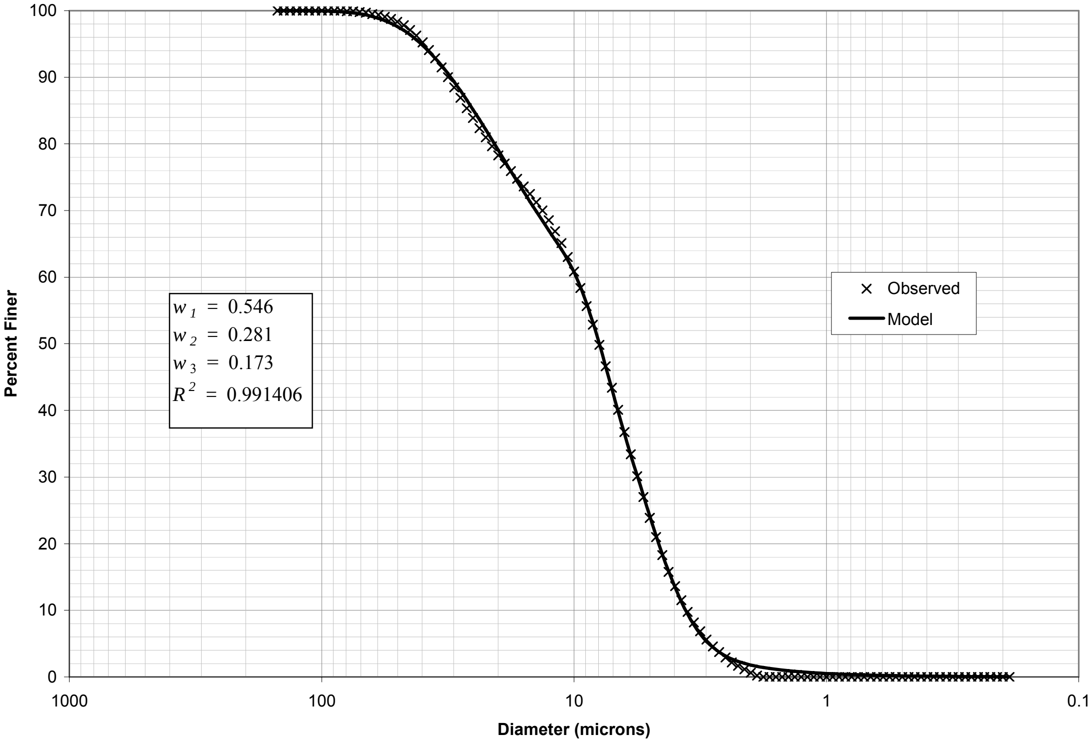

Figure 2 - 75: Trimodal PSD test # 7, replicate # 2.

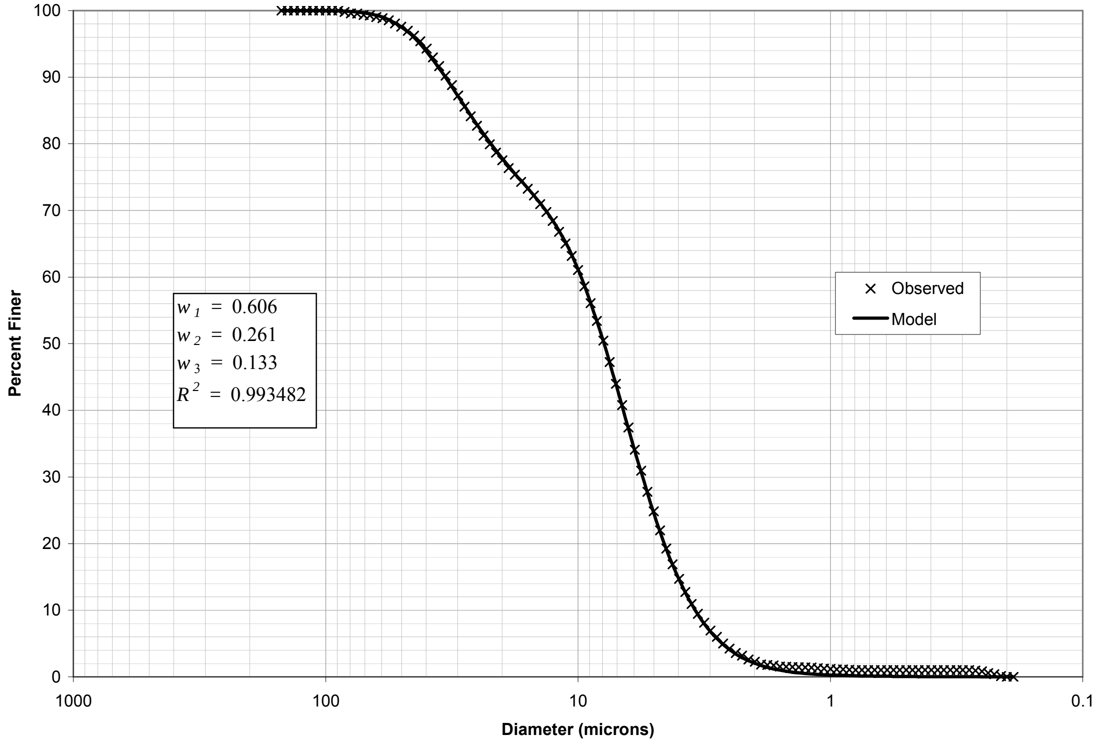

Figure 2 - 76: Trimodal PSD test # 7, replicate # 3.

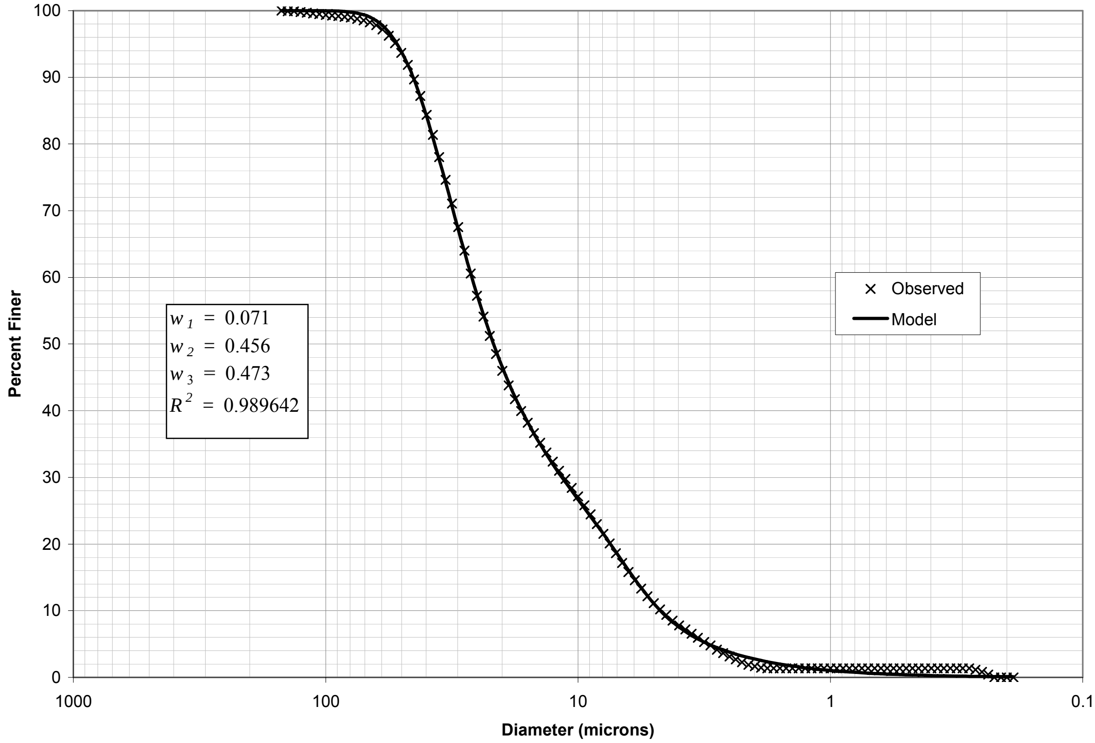

Figure 2 - 77: Trimodal PSD test # 8, replicate # 1.

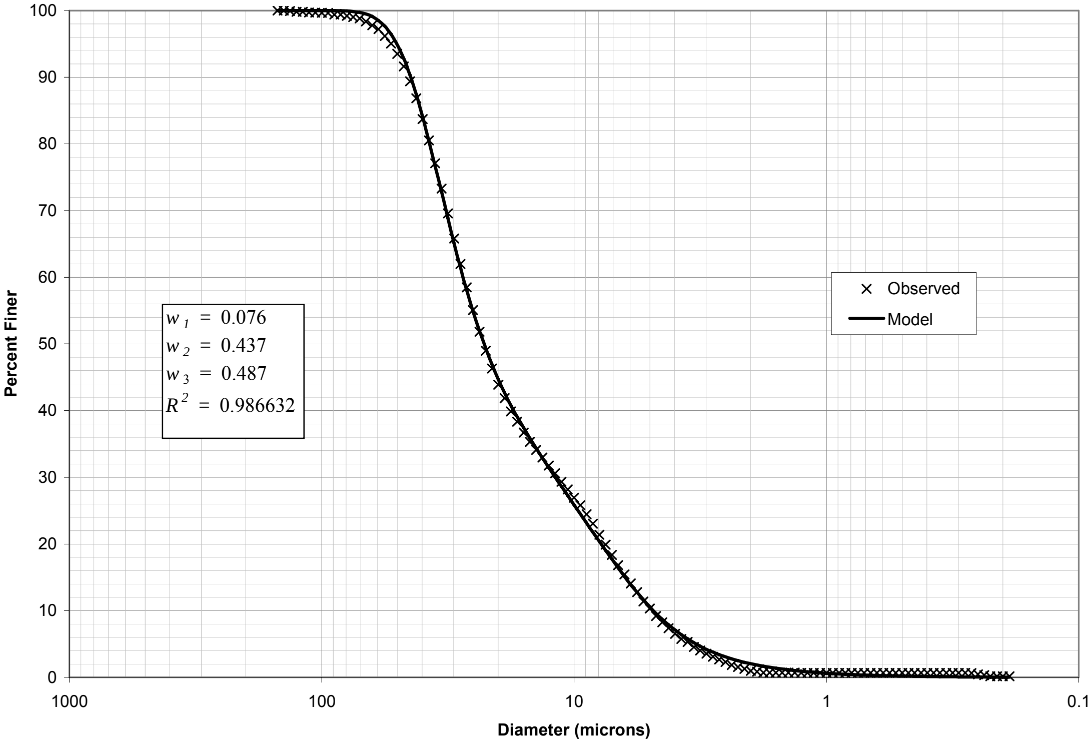

Figure 2 - 78: Trimodal PSD test # 8, replicate # 2.

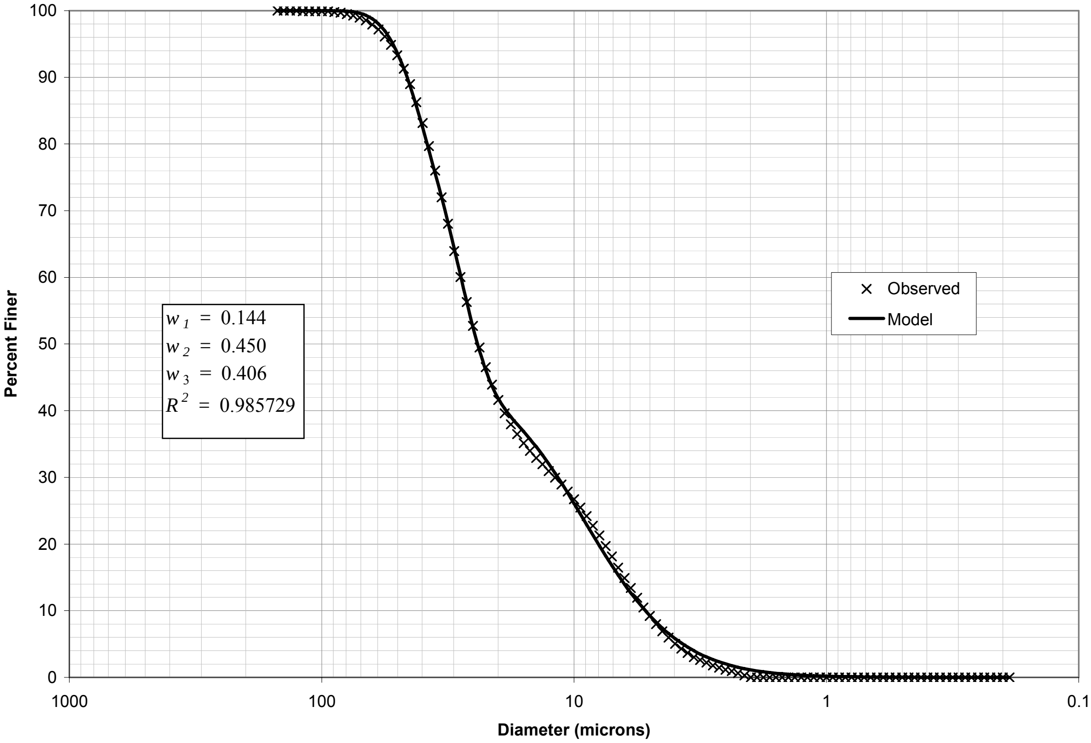

Figure 2 - 79: Trimodal PSD test # 8, replicate # 3.

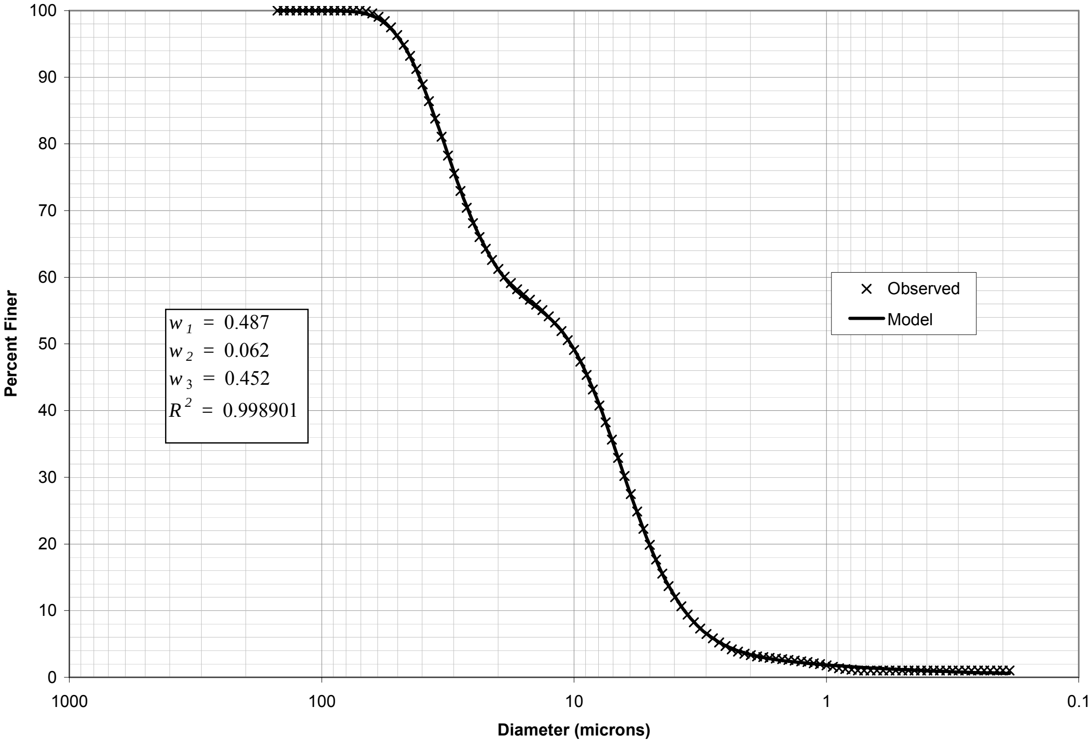

Figure 2 - 80: Trimodal PSD test # 9, replicate # 1.

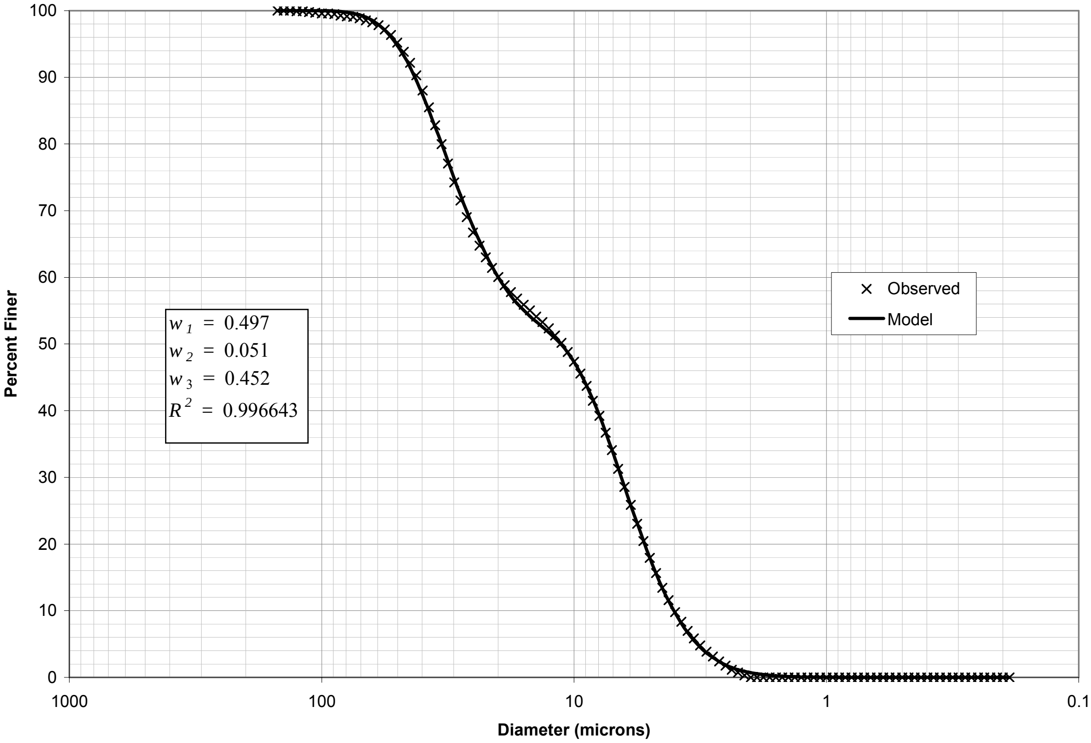

Figure 2 - 81: Trimodal PSD test # 9, replicate # 2.

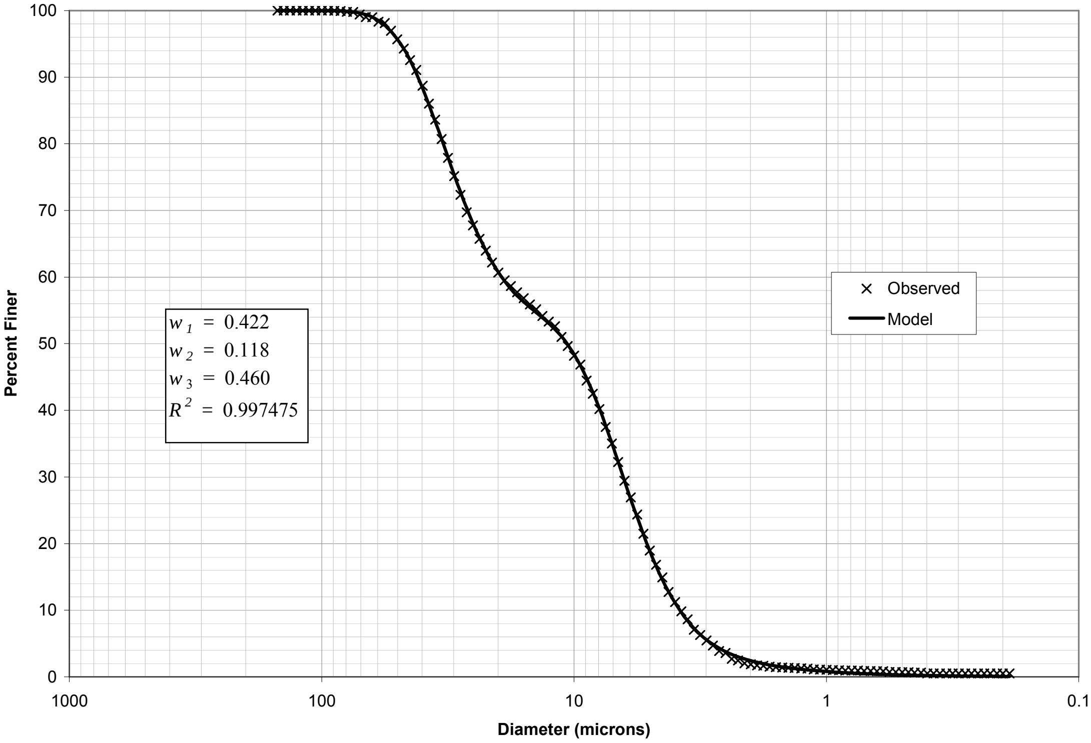

Figure 2 - 82: Trimodal PSD test # 9, replicate # 3.

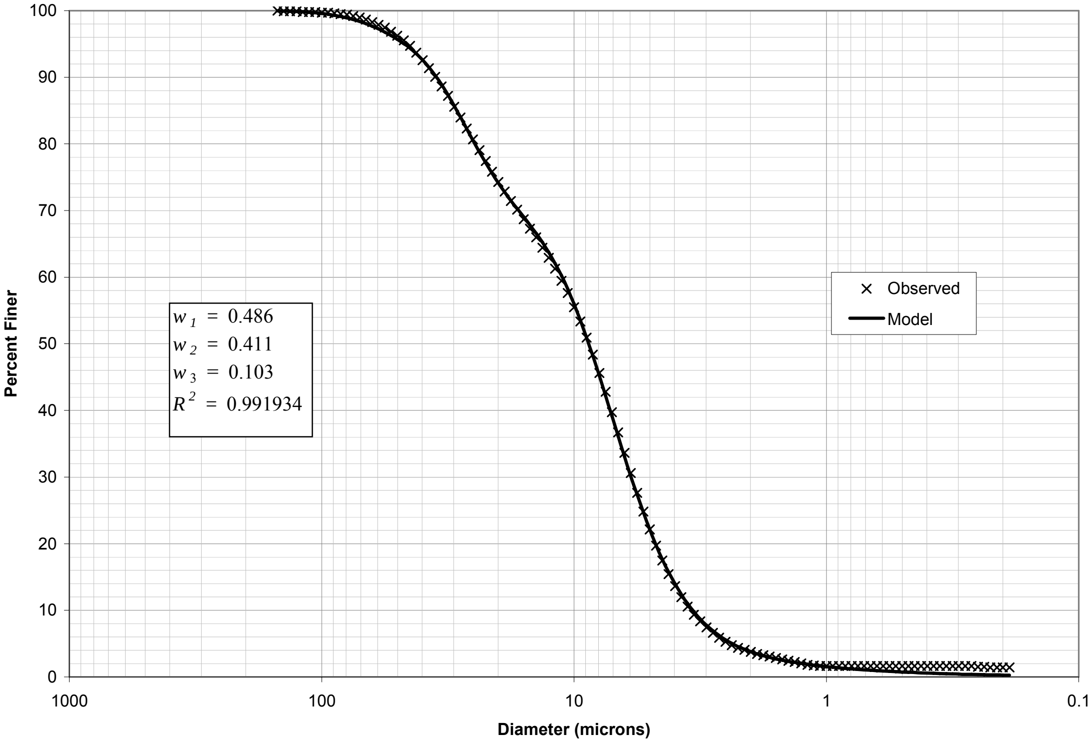

Figure 2 - 83: Trimodal PSD test # 10, replicate # 1.

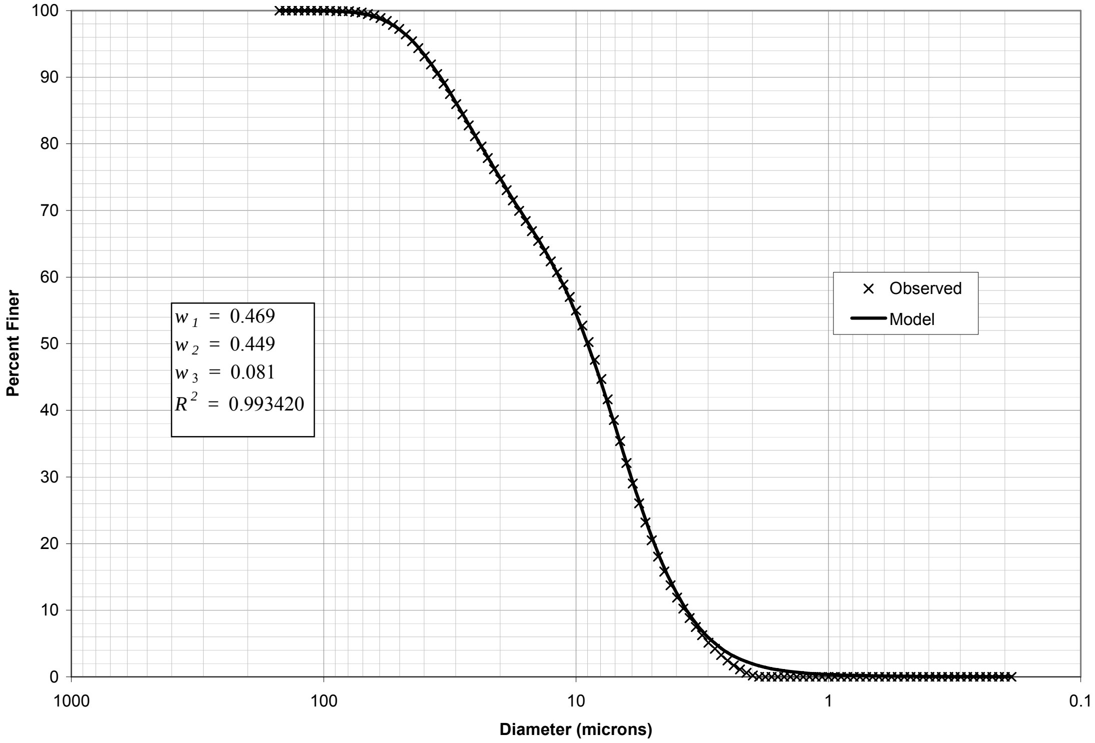

Figure 2 - 84: Trimodal PSD test # 10, replicate # 2.

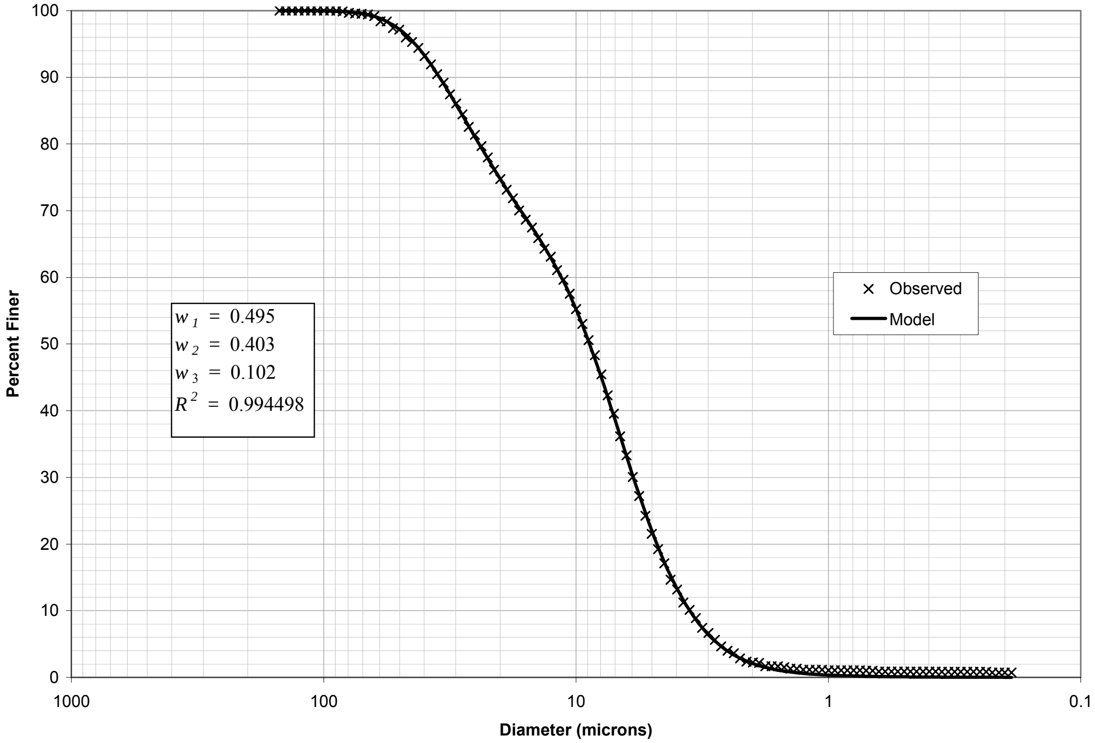

Figure 2 - 85: Trimodal PSD test # 10, replicate # 3.

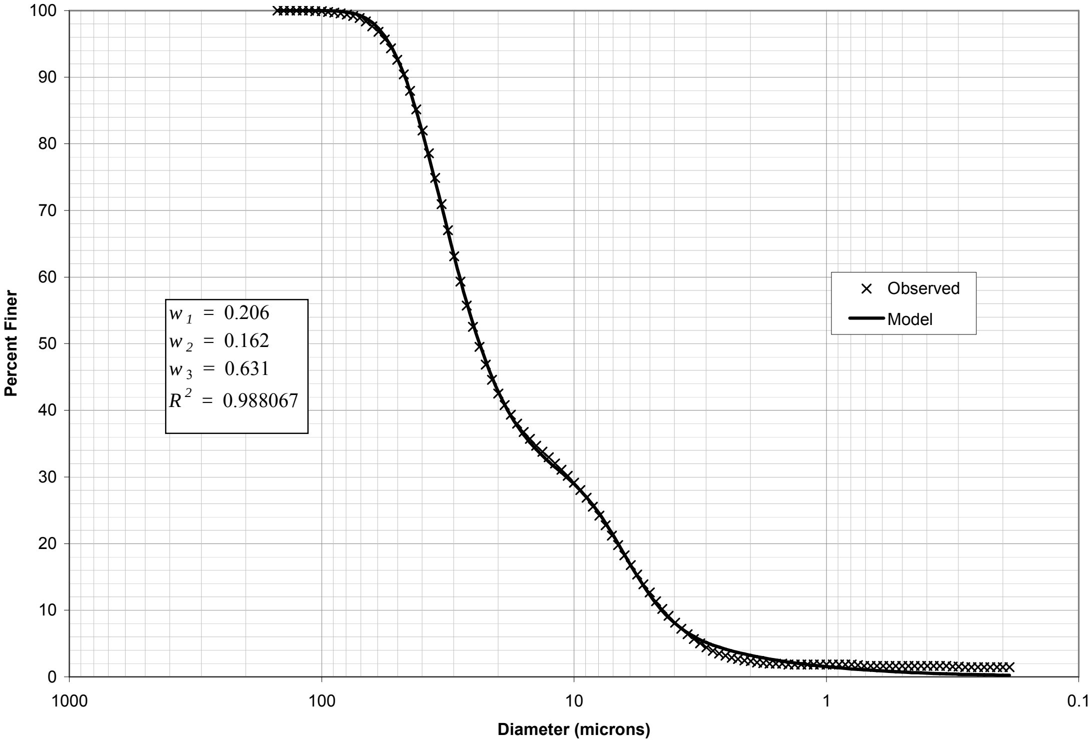

Figure 2 - 86: Trimodal PSD test # 11, replicate # 1.

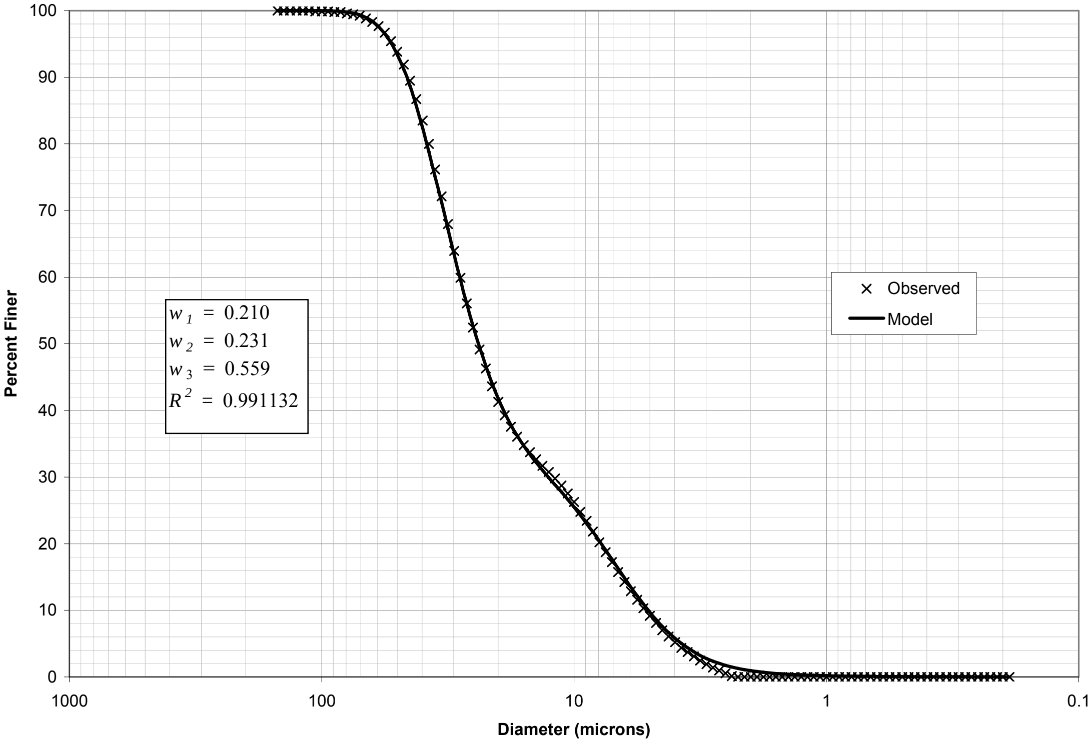

Figure 2 - 87: Trimodal PSD test # 11, replicate # 2.

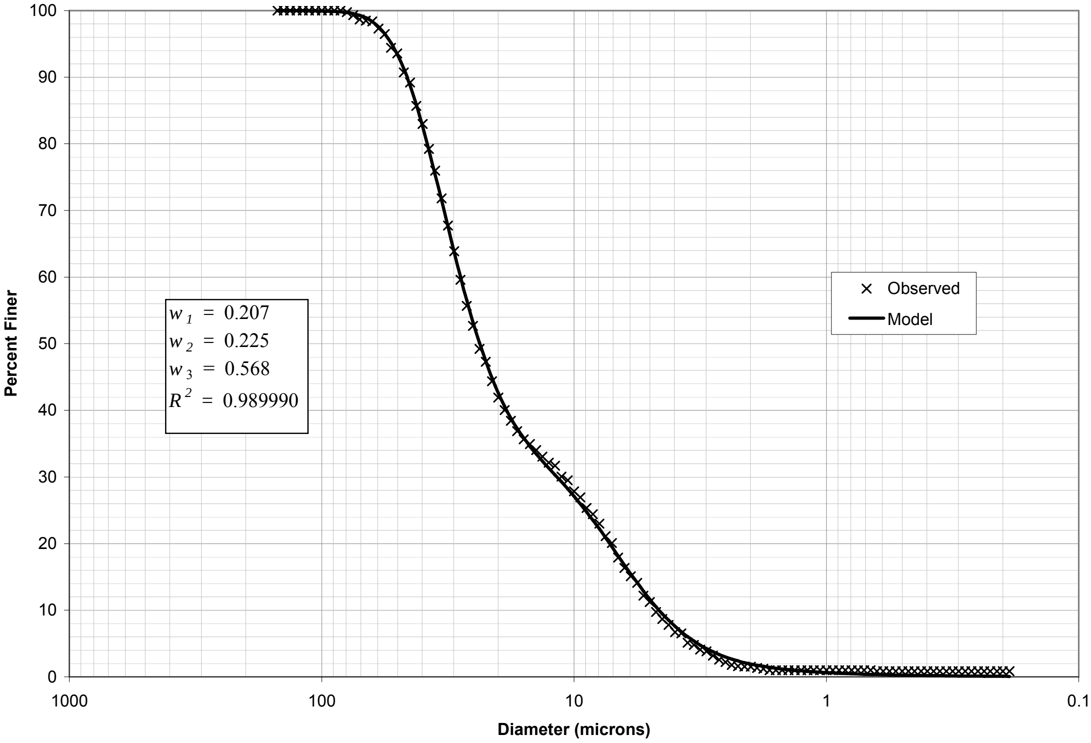

Figure 2 - 88: Trimodal PSD test # 11, replicate # 3.

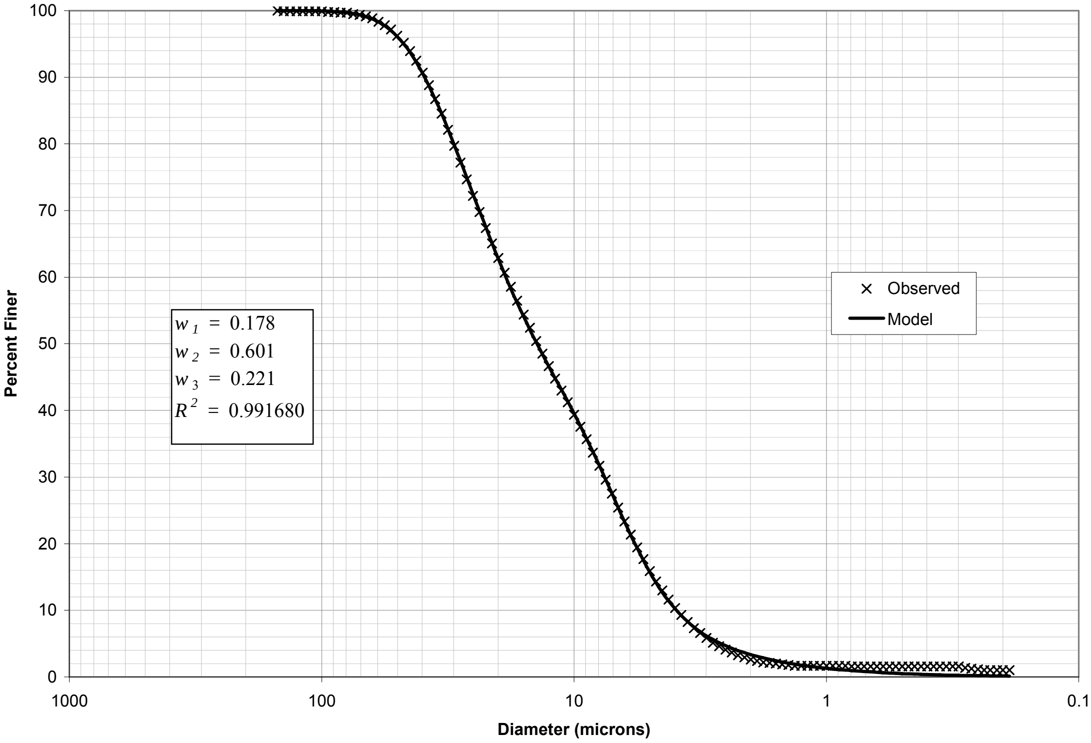

Figure 2 - 89: Trimodal PSD test # 12, replicate # 1.

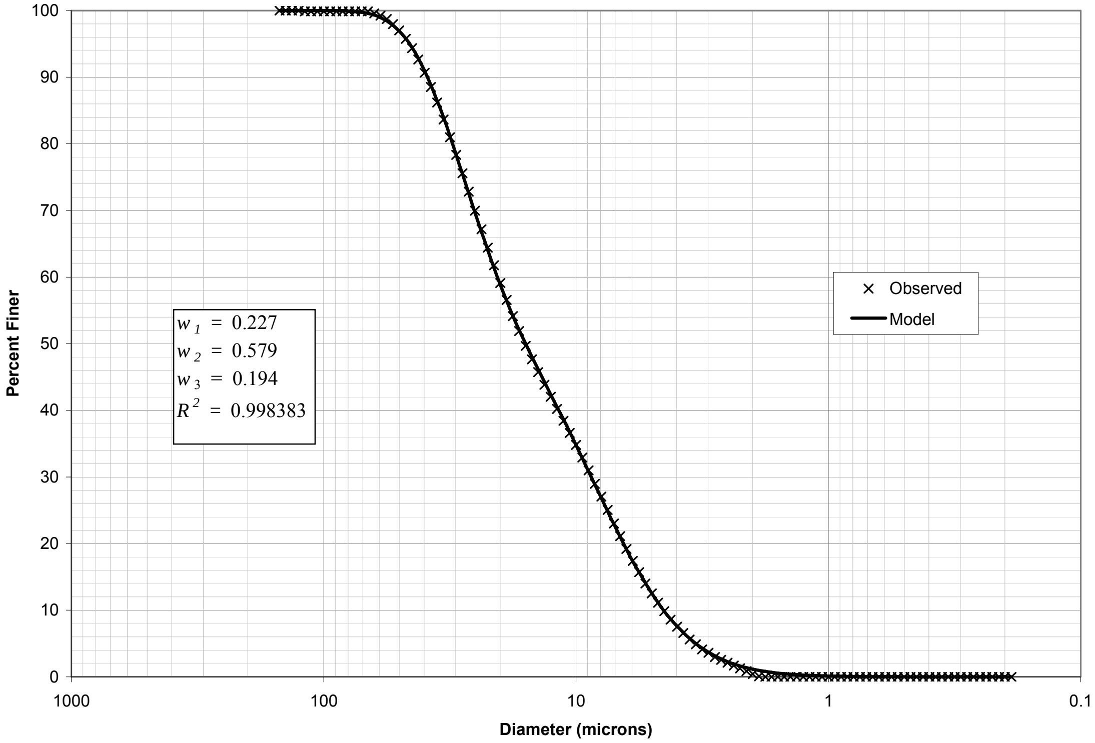

Figure 2 - 90: Trimodal PSD test # 12, replicate # 2.

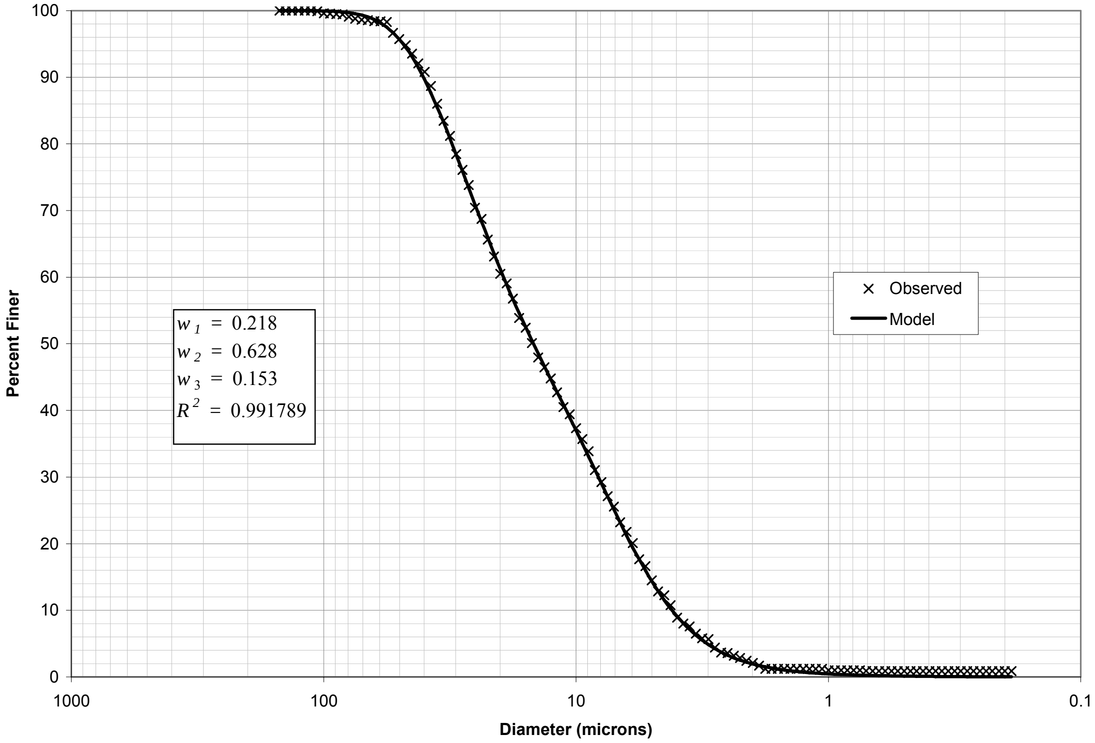

Figure 2 - 91: Trimodal PSD test # 12, replicate # 3.

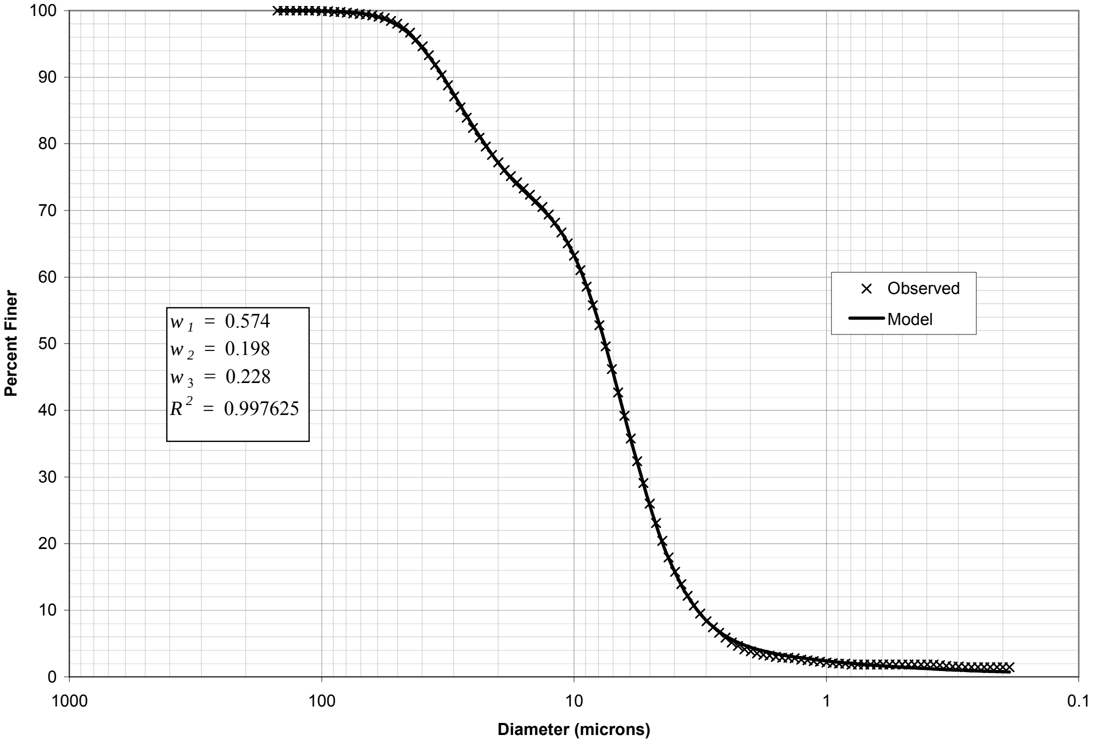

Figure 2 - 92: Trimodal PSD test # 13, replicate # 1.

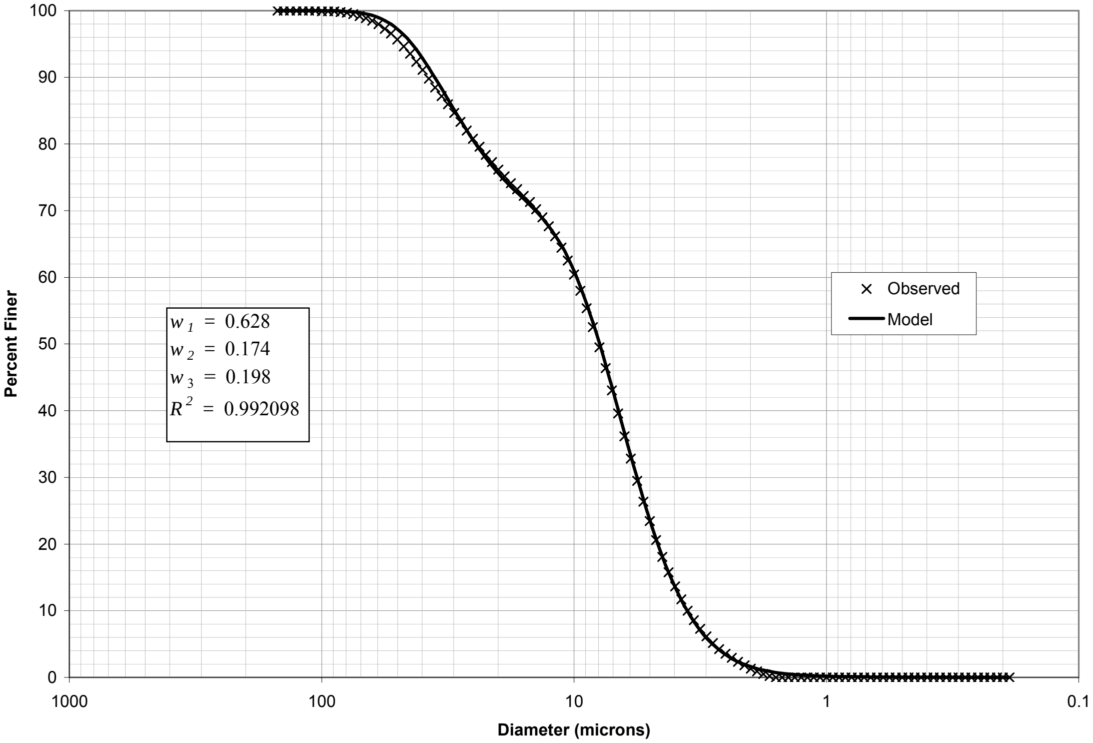

Figure 2 - 93: Trimodal PSD test # 13, replicate # 2.

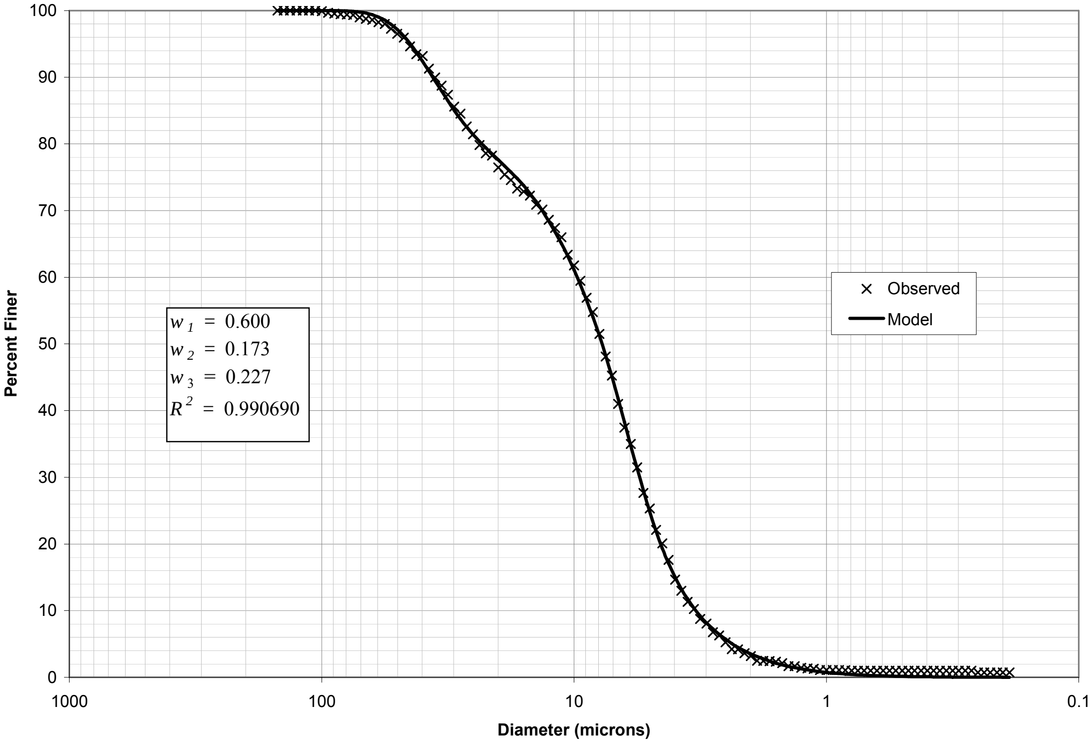

Figure 2 - 94: Trimodal PSD test # 13, replicate # 3.

APPENDIX E: Observed and Predicted PSD Curves Using Sediment Data

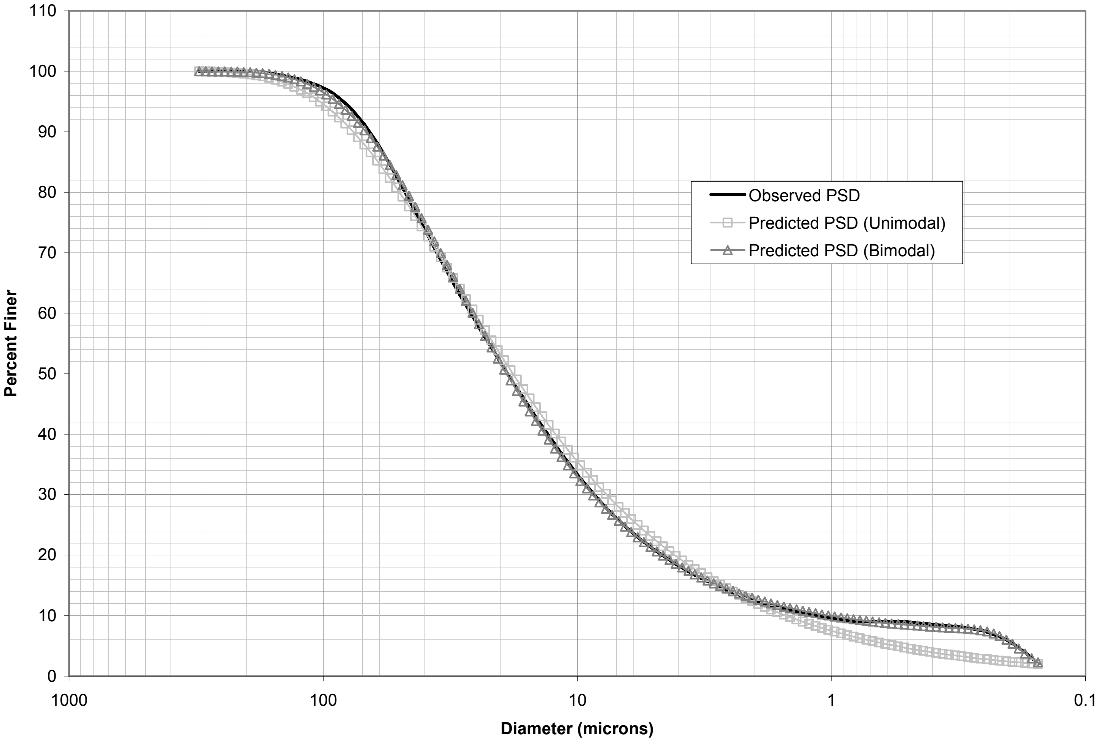

Figure 2 - 95: Observed vs. Predicted PSD data for Sediment Sample 1.

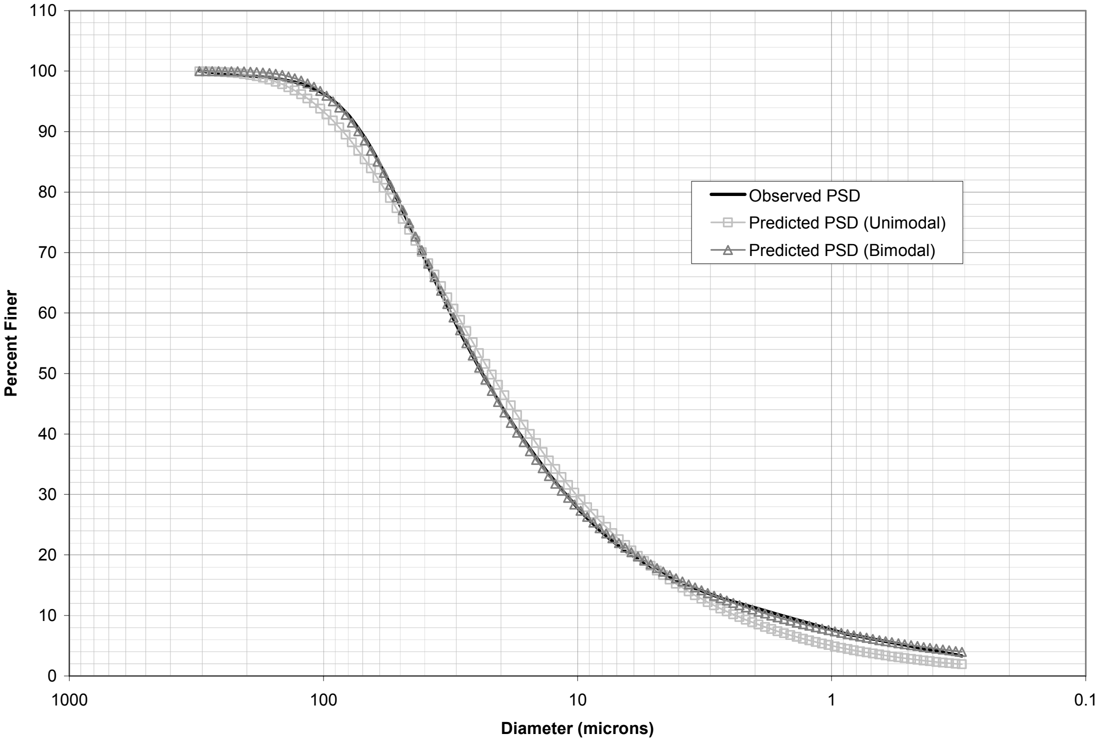

Figure 2 - 96: Observed vs. Predicted PSD data for Sediment Sample 2.

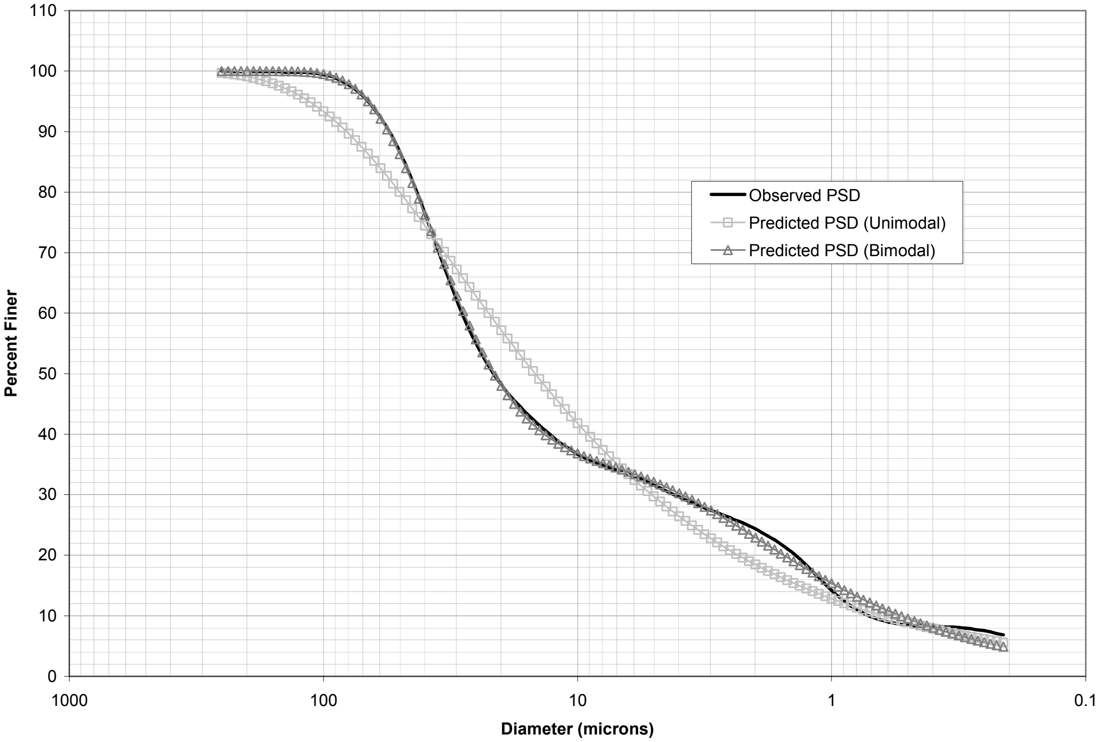

Figure 2 - 97: Observed vs. Predicted PSD data for Sediment Sample 3.

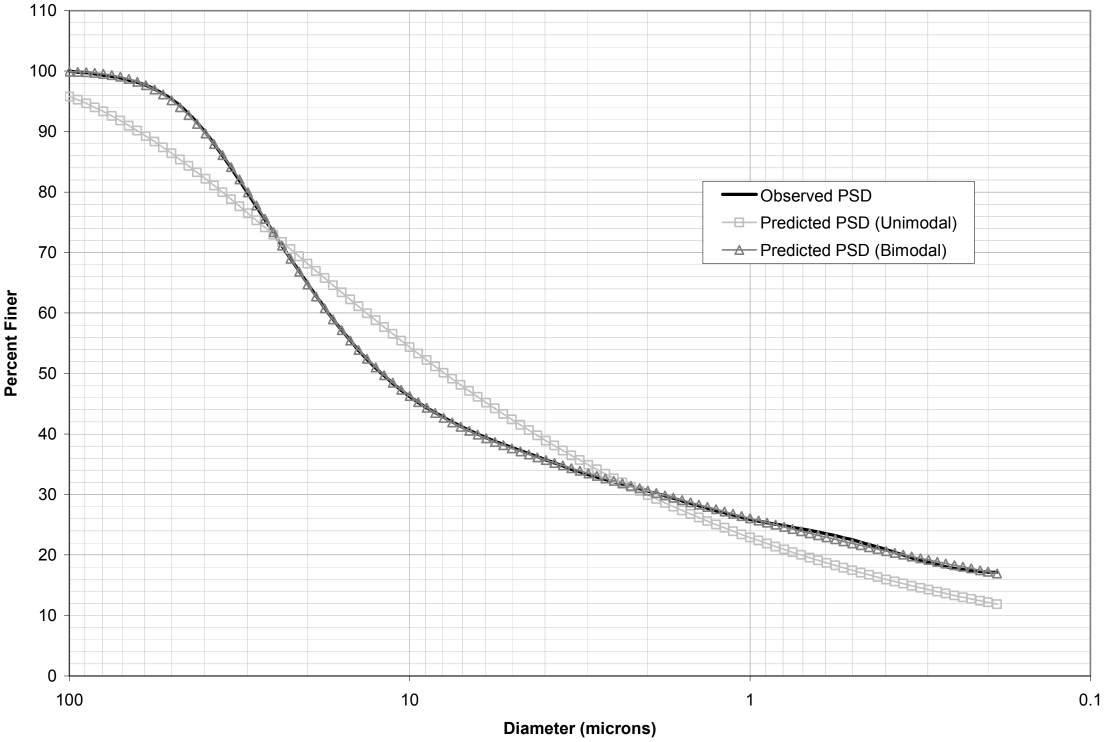

Figure 2 - 98: Observed vs. Predicted PSD data for Sediment Sample 4.

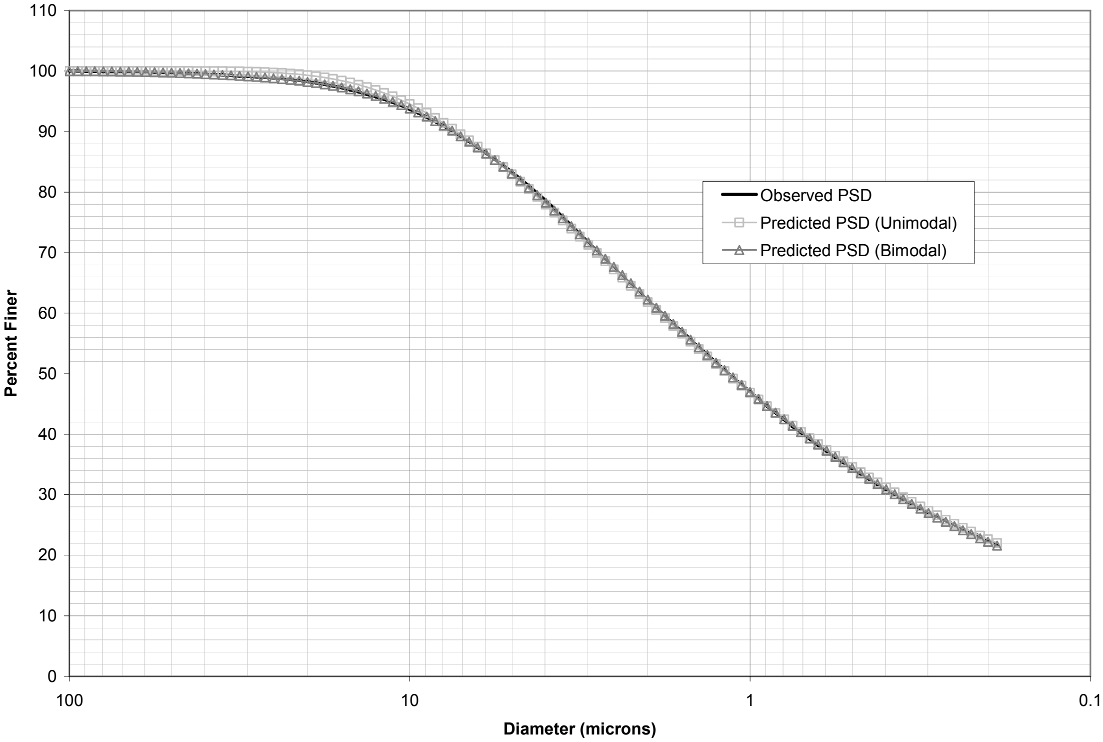

Figure 2 - 99: Observed vs. Predicted PSD data for Sediment Sample 5.

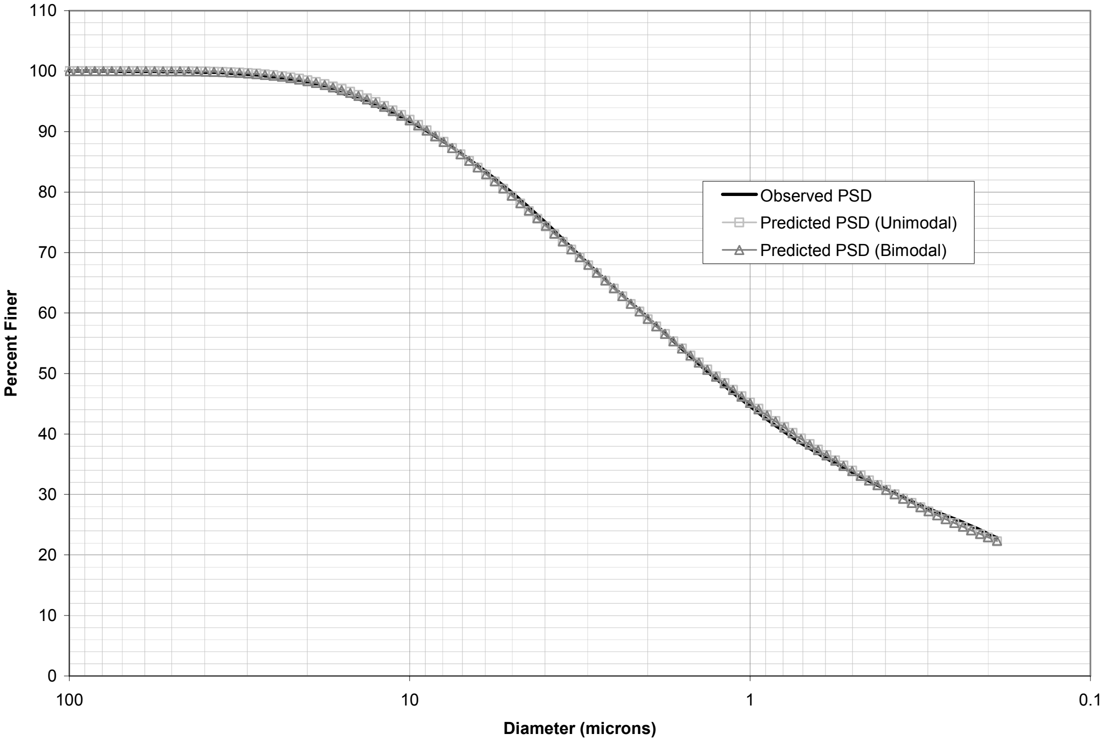

Figure 2 - 100: Observed vs. Predicted PSD data for Sediment Sample 6.

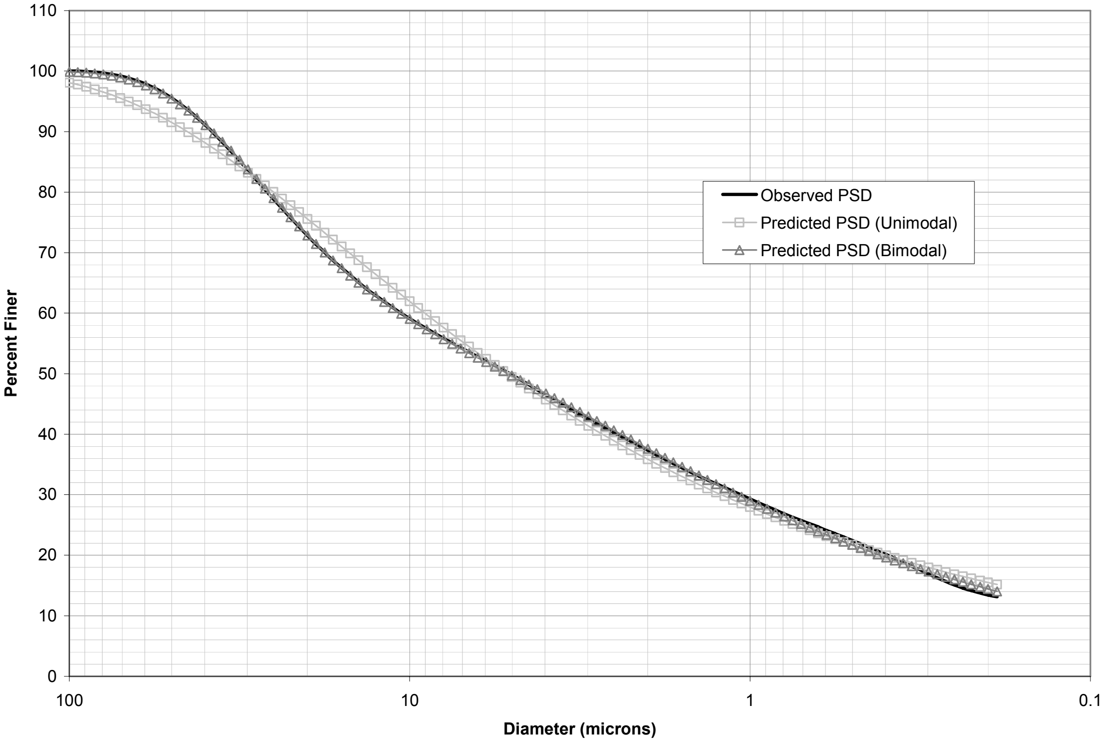

Figure 2 - 101: Observed vs. Predicted PSD data for Sediment Sample 7.

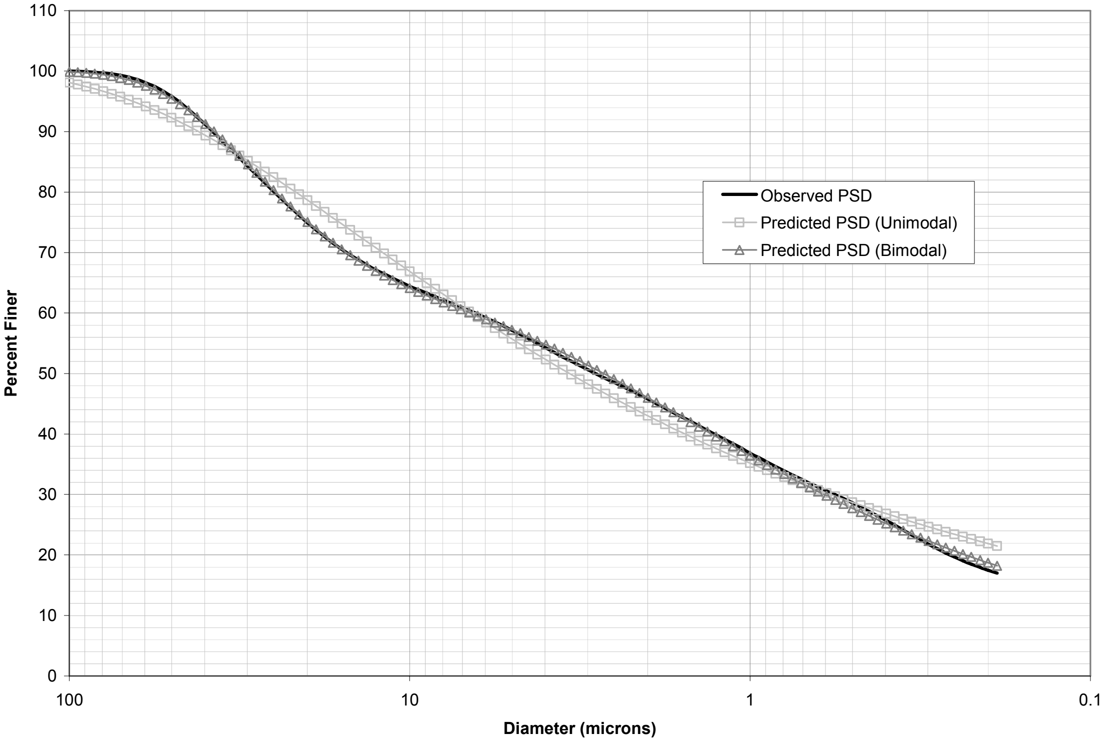

Figure 2 - 102: Observed vs. Predicted PSD data for Sediment Sample 8.

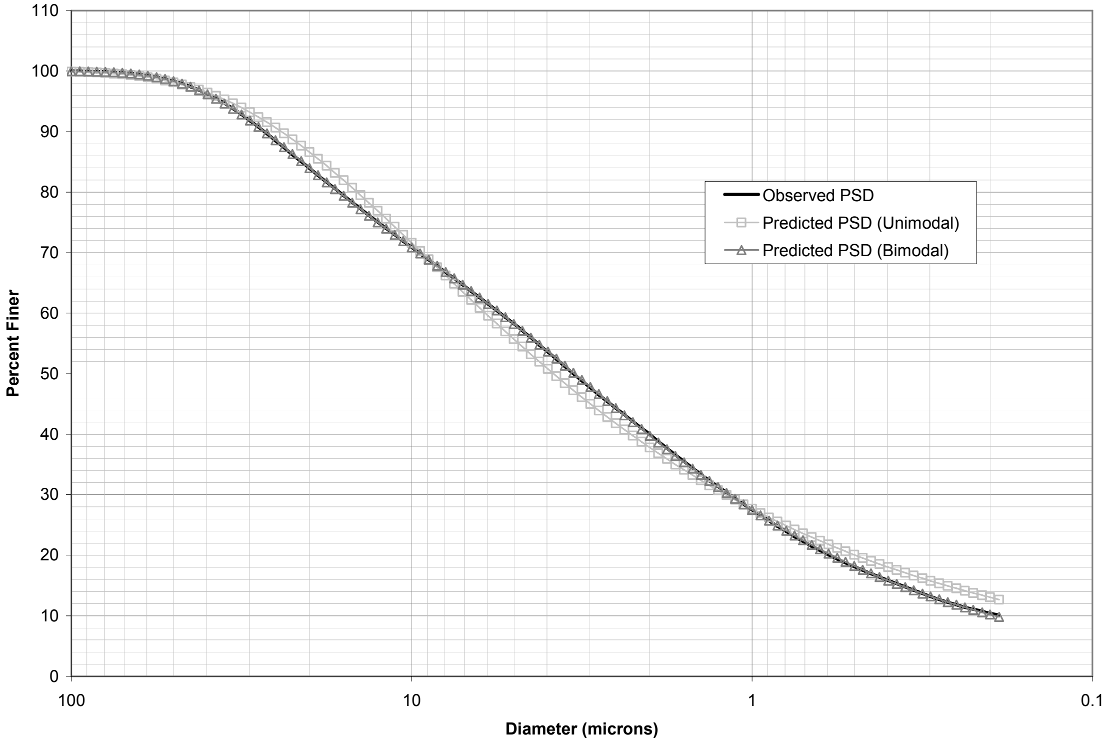

Figure 2 - 103: Observed vs. Predicted PSD data for Sediment Sample 9.

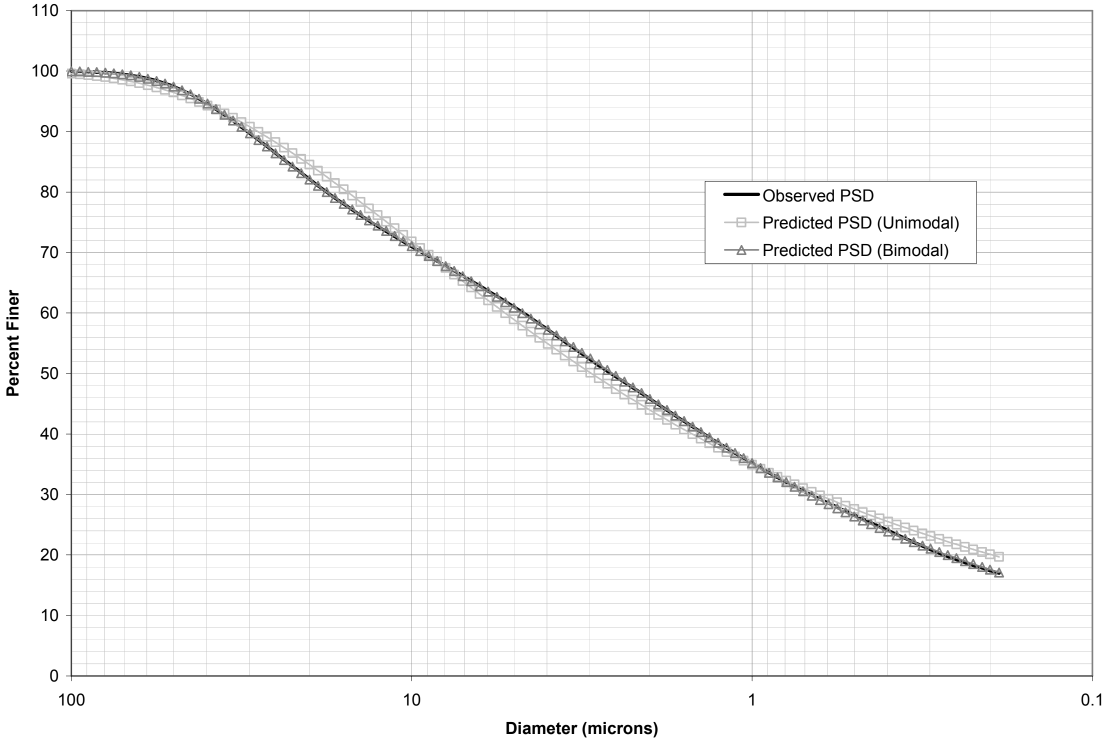

Figure 2 - 104: Observed vs. Predicted PSD data for Sediment Sample 10.

APPENDIX F: Derivation of the Aggregation Terms of the General Moments' Equation

1) Unimodal Case:

The first aggregation term of the general moments' equation is given by the following double integral:

$$I_{1,k} = \frac{1}{2}\int_0^\infty\int_0^\infty (x+y)^k K(x,y) f(x,t) f(y,t) dxc \qquad \text{(F.1)}$$

Inserting the aggregation kernel given by (3.8) and the unimodal Gamma PSD, the above equation results in:

$$\begin{aligned} I_{1,k} &= \frac{A}{2\beta^{2\alpha}\Gamma^2(\alpha)}\int_0^\infty\int_0^\infty (x+y)^k \left(x^r + y^r\right)^p x^{\alpha-1}e^{-x/\beta}y^{\alpha-1}e^{-y/\beta}dxdy \\ &= \frac{A}{2\beta^{2\alpha}\Gamma^2(\alpha)}\int_0^\infty\int_0^\infty \sum_{l=0}^{k}\binom{k}{l}x^{k-l}y^l \sum_{m=0}^{p}\binom{p}{m}x^{r(p-m)}y^{rm}x^{\alpha-1}e^{-x/\beta}y^{\alpha-1}e^{-y/\beta}dx \\ &= \frac{A}{2\beta^{2\alpha}\Gamma^2(\alpha)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\int_0^\infty x^{k-l+r(p-m)+\alpha-1}e^{-x/\beta}dx\int_0^\infty y^{l+rm+\alpha-1}e^{-y/\beta}dy. \end{aligned} \qquad \text{(F.2)}$$

Applying the variable substitutions $u = x/\beta$ and $v = y/\beta$ yields:

$$\begin{aligned} I_{1,k} &= \frac{A\beta^{k+rp}}{2\Gamma^2(\alpha)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\int_0^\infty u^{k-l+r(p-m)+\alpha-1}e^{-u}du\int_0^\infty v^{l+rm+\alpha-1}e^{-v}d \\ &= \frac{A\beta^{k+rp}}{2\Gamma^2(\alpha)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\Gamma(k-l+r(p-m)+\alpha)\Gamma(l+rm+ \end{aligned} \qquad \text{(F.3)}$$

after using the definition of the Gamma function.

Solving for the second term of (3.18):

$$\begin{aligned} I_{2,k} &= \frac{A}{\beta^{2\alpha}\Gamma^2(\alpha)}\int_0^\infty\int_0^\infty x^k\left(x^r + y^r\right)^p x^{\alpha-1}e^{-x/\beta}y^{\alpha-1}e^{-y/\beta}dxdy \\ &= \frac{A}{\beta^{2\alpha}\Gamma^2(\alpha)}\sum_{m=0}^{p}\binom{p}{m}\int_0^\infty x^{k+r(p-m)+\alpha-1}e^{-x/\beta}dx\int_0^\infty y^{rm+\alpha-1}e^{-x/\beta} \end{aligned} \qquad \text{(F.4)}$$

Similarly, applying the substitutions $u = -x/\beta$ and $v = -y/\beta$ results in:

$$\begin{aligned} I_{2,k} &= \frac{A\beta^{k+rp}}{\Gamma^2(\alpha)}\sum_{m=0}^{p}\binom{p}{m}\int_0^\infty u^{k+r(p-m)+\alpha-1}e^{-u}du\int_0^\infty v^{rm+\alpha-1}e^{-v} \\ &= \frac{A\beta^{k+rp}}{\Gamma^2(\alpha)}\sum_{m=0}^{p}\binom{p}{m}\Gamma(k+r(p-m)+\alpha)\Gamma(rm+\alpha) \end{aligned} \qquad \text{(F.5)}$$

after using the definition of the Gamma function.

2) Multimodal Case:

For the multimodal case the PSD $f(x,t)$ in (3.A.1) needs to be replaced by $\sum_{i=1}^{n} \omega_i f_i(x,t)$ and thus:

$$I_{1,k} = \frac{A}{2}\int_0^\infty\int_0^\infty (x+y)^k \left(x^r + y^r\right)^p \sum_{i=1}^{n} \omega_i f_i(x,t) \sum_{j=1}^{n} \omega_j f_j(y,t) dxc \tag{F.6}$$

Following the same steps conducted for unimodal case:

$$\begin{aligned} I_{1,k} &= \frac{A}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\omega_i\omega_j \int_0^\infty\int_0^\infty (x+y)^k \left(x^r + y^r\right)^p \frac{x^{\alpha_i - 1}e^{-x/\beta_i}}{\beta_i^{\alpha_i}\Gamma(\alpha_i)} \frac{y^{\alpha_j - 1}e^{-y/\beta_j}}{\beta_j^{\alpha_j}\Gamma(\alpha_j)} dxdy \\ &= \frac{A}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\omega_i\omega_j}{\beta_i^{\alpha_i}\beta_j^{\alpha_j}\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\int_0^\infty x^{k-l+r(p-m)+\alpha_i-1}e^{-x/\beta_i}dx \cdot \\ &\qquad \cdot \int_0^\infty y^{l+rm+\alpha_j-1}e^{-y/\beta_j}dy. \end{aligned} \tag{F.7}$$

Applying the variable substitutions $u = x/\beta_i$ and $v = y/\beta_j$ yields:

$$
\begin{aligned}
I_{1,k} &= \frac{A}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\omega_i\omega_j}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\beta_i^{k-l+r(p-m)}\beta_j^{l+rm}\int_0^\infty u^{k-l+r(p-m)+\alpha_i-1}e^{-u}du\,\cdot \\
&\qquad\qquad \cdot\int_0^\infty v^{l+rm+\alpha_j-1}e^{-v}dv \\
&= \frac{A}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\omega_i\omega_j}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{l=0}^{k}\sum_{m=0}^{p}\binom{k}{l}\binom{p}{m}\beta_i^{k-l+r(p-m)}\beta_j^{l+rm}\Gamma(k-l+r(p-m)+\alpha_i)\cdot. \\
&\qquad\qquad \cdot\Gamma(l+rm+\alpha_j),
\end{aligned}
\tag{F.8}
$$

The second aggregation term using a multimodal Gamma distribution as the initial PSD is given by:

$$
\begin{aligned}
I_{2,k} &= A\int_0^\infty\int_0^\infty x^k\left(x^r+y^r\right)^p\sum_{i=1}^{n}\omega_i\frac{x^{\alpha_i-1}e^{-x/\beta_i}}{\beta_i^{\alpha_i}\Gamma(\alpha_i)}\sum_{j=1}^{n}\omega_j\frac{y^{\alpha_j-1}e^{-y/\beta_j}}{\beta_j^{\alpha_j}\Gamma(\alpha_j)}dxdy \\
&= A\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\omega_i\omega_j}{\beta_i^{\alpha_i}\beta_j^{\alpha_j}\Gamma(\alpha_i)\Gamma(\beta_j)}\sum_{m=0}^{p}\binom{p}{m}\int_0^\infty x^{k+r(p-m)+\alpha_i-1}e^{-x/\beta_i}dx\int_0^\infty y^{rm+\alpha_j-1}e^{-x/\beta_j}dy.
\end{aligned}
\tag{F.9}
$$

Once again applying the substitutions $u=-x/\beta_i$ and $v=-y/\beta_i$ results in:

$$
\begin{aligned}
I_{2,k} &= A\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\omega_i\omega_j}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{m=0}^{p}\binom{p}{m}\beta_i^{k+r(p-m)}\beta_j^{rm}\int_0^\infty u^{k+r(p-m)+\alpha-1}e^{-u}du\int_0^\infty v^{rm+\alpha-1}e^{-v}dv \\
&= A\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\omega_i\omega_j}{\Gamma(\alpha_i)\Gamma(\alpha_j)}\sum_{m=0}^{p}\binom{p}{m}\beta_i^{k+r(p-m)}\beta_j^{rm}\Gamma(k+r(p-m)+\alpha)\Gamma(rm+\alpha),
\end{aligned}
\tag{F.10}
$$

after using the definition of the Gamma function.

APPENDIX G: Derivation of the Fragmentation Terms of the General Moments' Equation

1) Unimodal Case:

The two terms of the moments' equation (3.24) related to fragmentation only are given by the following expressions:

$$J_{1,k} = \int_x^\infty \int_0^\infty x^k \varphi(y) f(y/x) f(y,t) dxdy, \tag{G.1}$$

and

$$J_{2,k} = \int_0^\infty x^k \varphi(x) f(x) dx. \tag{G.2}$$

where $\varphi(x) = Bx^\lambda$. The fragmentation PSD is assumed to also follow a Gamma distribution; however the parameters are different. Expanding (3.B.1) results in:

$$\begin{aligned} J_{1,k} &= \frac{B}{b^a \beta^\alpha \Gamma(a)\Gamma(\alpha)} \int_x^\infty \int_0^\infty x^k y^\lambda \left(y/x\right)^{a-1} e^{-y/bx} y^{\alpha-1} e^{-y/\beta} dxdy \\ &= \frac{B}{b^a \beta^\alpha \Gamma(a)\Gamma(\alpha)} \int_0^\infty \int_x^\infty x^{k-a+1} y^{\lambda+a+\alpha-2} \exp\left(-y\left(\frac{bx+\beta}{b\beta x}\right)\right) dyc \end{aligned} \tag{G.3}$$

Let $v = y\left(\frac{bx + \beta}{b\beta x}\right)$, substituting in (3.B.3) results in:

$$\begin{aligned} J_{1,k} &= \frac{Bb^{\lambda+\alpha-1}\beta^{\lambda+a-1}}{\Gamma(a)\Gamma(\alpha)} \int_0^\infty \int_x^\infty \frac{x^{k+\lambda+\alpha}}{(bx+\beta)^{\lambda+a+\alpha-1}} v^{\lambda+a+\alpha-2} e^{-v} dv dx \\ &= \frac{Bb^{\lambda+\alpha-1}\beta^{\lambda+a-1}}{\Gamma(a)\Gamma(\alpha)} \int_0^\infty \frac{x^{k+\lambda+\alpha}}{(bx+\beta)^{\lambda+a+\alpha-1}} \Gamma(\lambda + a + \alpha - 1, x)c \end{aligned} \tag{G.4}$$

For the second term $J_{2,k}$:

$$\begin{aligned} J_{2,k} &= \frac{B}{\beta^\alpha \Gamma(\alpha)} \int_0^\infty x^{k+\lambda+\alpha-1} \exp(-x/\beta) dx \\ &= \frac{B\beta^{k+\lambda+\alpha}}{\beta^\alpha \Gamma(\alpha)} \int_0^\infty u^{k+\lambda+\alpha-1} e^{-u} du, \end{aligned} \tag{G.5}$$

after substitution of $u = x/\beta$ was made. Using the definition of the Gamma function, one finally obtains:

$$J_{2.k} = \frac{B\beta^{k+\lambda}}{\Gamma(\alpha)} \Gamma(k + \lambda + \alpha). \tag{G.6}$$

2) Multimodal Case:

It was assumed for the multimodal case that the PSD of the fragments follows a unimodal Gamma distribution since little knowledge exist on the shape of this distribution and also to simplify the equations. The first term of the fragmentation is given by:

$$J_{1,k} = \int_x^\infty \int_0^\infty x^k \varphi(y) f(y/x) f(y,t) dxdy \tag{G.7}$$

Using the multimodal Gamma distribution for $f(y,t)$ yields:

$$\begin{aligned} J_{1,k} &= \frac{B}{b^a \Gamma(a)} \int_x^\infty \int_0^\infty x^k y^\lambda (y/x)^{a-1} e^{-y/bx} \sum_{i=1}^n \omega_i \frac{y^{\alpha_i - 1} e^{-y/\beta_i}}{\beta_i^{\alpha_i} \Gamma(\alpha_i)} dxdy \\ &= \frac{B}{b^a \Gamma(a)} \sum_{i=1}^n \frac{\omega_i}{\beta_i^{\alpha_i} \Gamma(\alpha_i)} \int_0^\infty \int_x^\infty x^{k-a+1} y^{\lambda + a + \alpha_i - 2} \exp\left(-y \left(\frac{bx + \beta_i}{b\beta_i x} \right) \right) dyc \end{aligned} \tag{G.8}$$

Let $v = y\left(\frac{bx + \beta_i}{b\beta_i x} \right)$ and substituting in (3.B.8) results in:

$$\begin{aligned} J_{1,k} &= \frac{B}{\Gamma(a)} \sum_{i=1}^n \omega_i \frac{b^{\lambda + \alpha_i - 1} \beta_i^{\lambda + a - 1}}{\Gamma(\alpha_i)} \int_0^\infty \int_x^\infty \frac{x^{k + \lambda + \alpha_i}}{(bx + \beta_i)^{\lambda + a + \alpha_i - 1}} v^{\lambda + a + \alpha_i - 2} e^{-v} dvdx \\ &= \frac{B}{\Gamma(a)} \sum_{i=1}^n \omega_i \frac{b^{\lambda + \alpha_i - 1} \beta_i^{\lambda + a - 1}}{\Gamma(\alpha_i)} \int_0^\infty \frac{x^{k + \lambda + \alpha_i}}{(bx + \beta_i)^{\lambda + a + \alpha_i - 1}} \Gamma(\lambda + a + \alpha_i - 1, x) dx. \end{aligned} \tag{G.9}$$

For the second term $J_{2,k}$:

$$\begin{aligned} J_{2,k} &= B\sum_{i=1}^{n}\frac{\omega_i}{\beta_i^{\alpha_i}\Gamma(\alpha_i)}\int_0^\infty x^{k+\lambda+\alpha_i-1}\exp\left(-x/\beta_i\right)dx \\ &= B\sum_{i=1}^{n}\omega_i\frac{\beta_i^{k+\lambda+\alpha_i}}{\beta_i^{\alpha_i}\Gamma(\alpha_i)}\int_0^\infty u^{k+\lambda+\alpha_i-1}e^{-u}du, \end{aligned} \tag{G.10}$$

after substitution of $u = x/\beta_i$ was made. Using the definition of the Gamma function, one finally obtains:

$$J_{2.k} = B\beta^{k+\lambda}\sum_{i=1}^{n}\frac{\omega_i}{\Gamma(\alpha_i)}\Gamma(k+\lambda+\alpha_i). \tag{G.11}$$

APPENDIX H: Initial and Flocculated PSD Curves

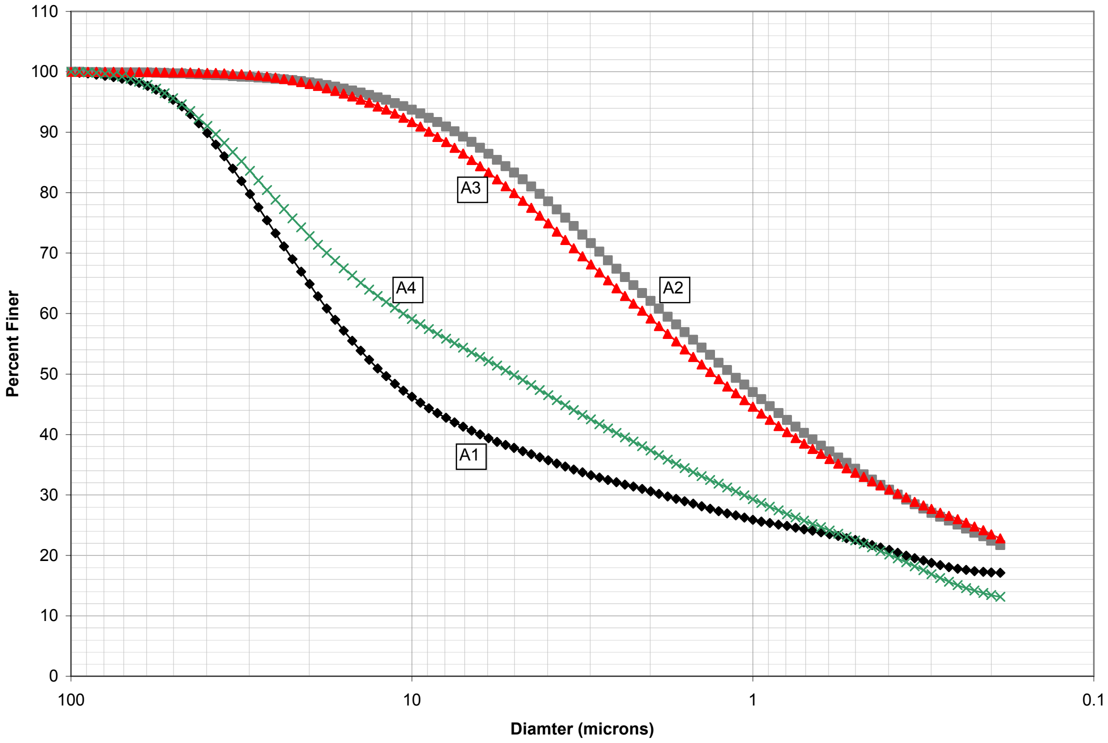

Figure 4 - 4: Initial PSD for Sediment Samples A1 through A6

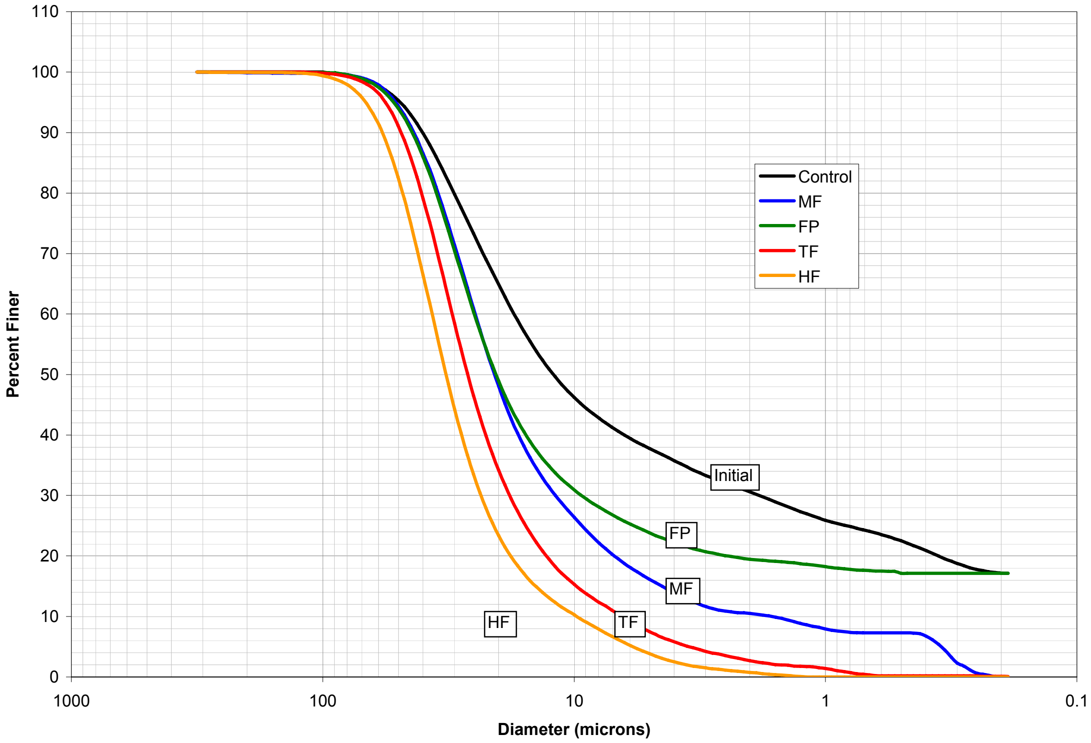

Figure 4 - 5: Flocculated PSD curves of the A1 sediment sample and the four different flocculated cases

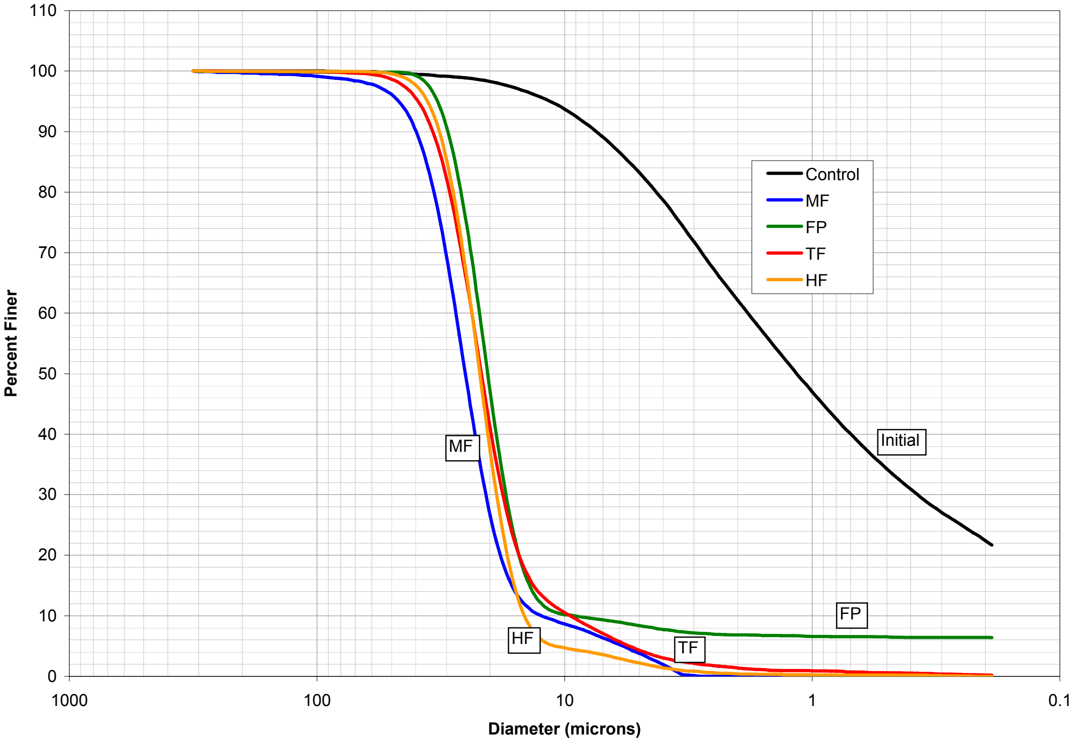

Figure 4 - 6: Flocculated PSD curves of the A2 sediment sample and the four different flocculated cases

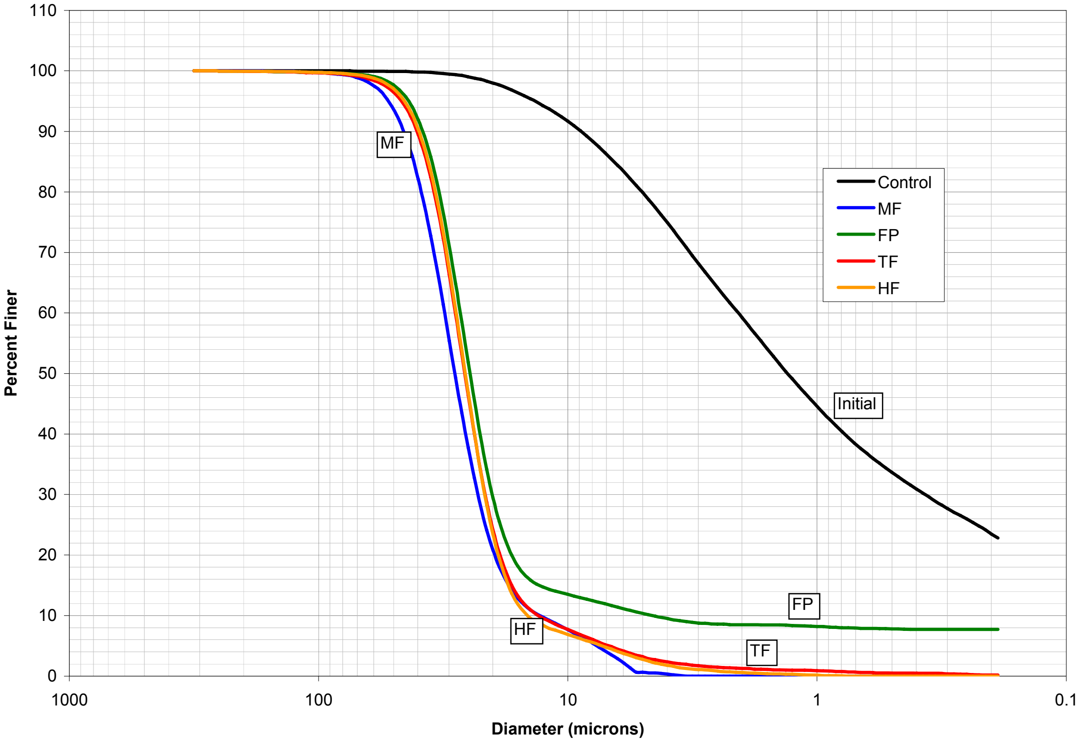

Figure 4 - 7: Flocculated PSD curves of the A3 sediment sample and the four different flocculated cases

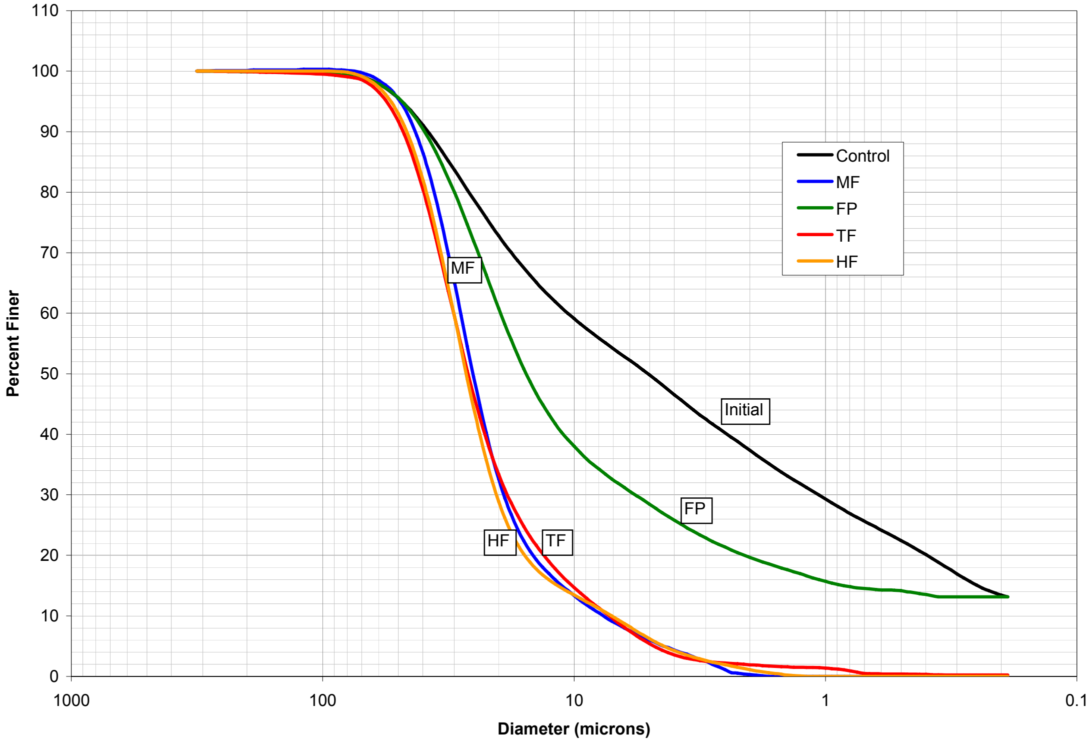

Figure 4 - 8: Flocculated PSD curves of the A4 sediment sample and the four different flocculated cases

APPENDIX I: Computer Code of the Genetic Algorithm

Note: The Meta Numeric library was used in this program (www.meta-numerics.net).

1) The Multimodal Aggregation and Fragmentation Model

```
namespace Chapter4_Code_Multimodal
{
    class Multimodal
    {
        /// <summary>
        /// Class attributes/members
        /// <summary>
        int nmodes;
        double A;
        double r;
        int p;
        double B;
        double lambda;
        double a;
        double b;
        ColumnVector IniVec;

        /// <summary>
        /// Constructor
        /// <summary>
        public Multimodal(int num_modes, double A_coeff, double
r_coeff, int p_coeff, double B_coeff, double lambda_coeff, double
a_coeff, double b_coeff, ColumnVector Initial_Vector)
        {
            nmodes = num_modes;
            A = A_coeff;
            r = r_coeff;
            p = p_coeff;
            B = B_coeff;
            lambda = lambda_coeff;
            a = a_coeff;
            b = b_coeff;
            IniVec = Initial_Vector.Clone();
        }

        /// <summary>
        /// Properties
        /// <summary>
        public double A_coefficient
        {
            get { return A; }
        }
        public double r_coefficient
        {
            get { return r; }
        }
        public int p_coefficient
        {
            get { return p; }
        }
        public double B_coefficient
        {
            get { return B; }
        }
        public double lambda_coefficient
        {
            get { return lambda; }
        }
        public double a_coefficient
        {
            get { return a; }
        }
        public double b_coefficient
        {
            get { return b; }
        }

        /// <summary>
        /// Static Method that is equivalent to the product of a
diagonal matrix
        /// and a column vector. This method reduces considerably the
computational
        /// cost by replacing the diagonal with its equivalent column
vector
        /// <summary>
        public static ColumnVector Mult(ColumnVector Vector1,
ColumnVector Vector2)
        {
            ColumnVector theProduct = new
ColumnVector(Vector1.Dimension);
            for (int i = 0; i < Vector1.Dimension; i++)
            {
                theProduct[i] = Vector1[i] * Vector2[i];
            }
            return theProduct;
        }

        /// <summary>
        /// Public method that returns the first aggregation term of
the
        /// continuous Smoluchowski coagulation equation
        /// <summary>
        public double I1k(int k, int i, int j, ColumnVector Pvec)
        {
            double sum = 0;

            double c1 = A / (2 * AdvancedMath.Gamma(Pvec[i + nmodes]) *
AdvancedMath.Gamma(Pvec[j + nmodes]));

            for (int l = 0; l <= k; l++)
            {
                for (int m = 0; m <= p; m++)
                {
                    double i1 = k - l + r * (p - m);
                    double i2 = l + r * m;
                    double bc =
AdvancedIntegerMath.BinomialCoefficient(k, l) *
AdvancedIntegerMath.BinomialCoefficient(p, m);
                    double c2 = Math.Pow(Pvec[i + 2 * nmodes], i1) *
Math.Pow(Pvec[j + 2 * nmodes], i2);
```

```
                    double c3 = AdvancedMath.Gamma(i1 + Pvec[i + 
nmodes]) * AdvancedMath.Gamma(i2 + Pvec[j + nmodes]);
                    sum += bc * c2 * c3;
                }
            }
            return c1 * sum;
        }

        /// <summary>
        /// Public method that returns the second aggregation term of 
the
        /// continuous Smoluchowski coagulation equation
        /// <summary>
        public double I2k(int k, int i, int j, ColumnVector Pvec)
        {
            double sum = 0;

            double c1 = A / (AdvancedMath.Gamma(Pvec[i + nmodes]) * 
AdvancedMath.Gamma(Pvec[j + nmodes]));

            for (int m = 0; m <= p; m++)
            {
                double i1 = k + r * (p - m);
                double i2 = r * m;
                double bc = AdvancedIntegerMath.BinomialCoefficient(p, 
m);
                double c2 = Math.Pow(Pvec[i + 2 * nmodes], i1) * 
Math.Pow(Pvec[j + 2 * nmodes], i2);
                double c3 = AdvancedMath.Gamma(i1 + Pvec[i + nmodes]) * 
AdvancedMath.Gamma(i2 + Pvec[j + nmodes]);
                sum += bc * c2 * c3;
            }
            return c1 * sum;
        }

        /// <summary>
        /// Public method that returns the first fragmentation term of 
the
        /// continuous Smoluchowski coagulation equation
        /// <summary>
        public double J1k(int k, int i, ColumnVector Pvec)
        {
            double alpha = Pvec[i + nmodes];
            double beta = Pvec[i + 2 * nmodes];
            double c1 = B * Math.Pow(beta, lambda + k + a) * 
Math.Pow(b, k);
            double c2 = Math.Pow(beta + b, k + a) * 
AdvancedMath.Gamma(a) * AdvancedMath.Gamma(alpha);
            double c3 = AdvancedMath.Gamma(lambda + alpha - 1) * 
AdvancedMath.Gamma(k + a - 1);

            return c1 * c3 / c2;
        }

        /// <summary>
        /// Public method that returns the second fragmentation term of 
the
```

```
        /// continuous Smoluchowski coagulation equation
        /// <summary>
        public double J2k(int k, int i, ColumnVector Pvec)
        {
            double alpha = Pvec[i + nmodes];
            double beta = Pvec[i + 2 * nmodes];
            double c1 = B * Math.Pow(beta, k + lambda) /
(AdvancedMath.Gamma(alpha));
            double c2 = AdvancedMath.Gamma(k + lambda + alpha);

            return c1 * c2;
        }

        /// <summary>
        /// Private method that returns the derivative coefficients
        /// of the moments of the d_w/dt term
        /// <summary>
        private double Wi_Derivk(int k, int i, ColumnVector Pvec)
        {
            double product = 1;
            double c1 = Math.Pow(Pvec[i + 2 * nmodes], k - 1);

            for (int j = 2; j <= k; j++)
            {
                product = product * (k - j + Pvec[i + nmodes]);
            }
            return c1 * product;
        }
        /// <summary>
        /// Private method that returns the derivative coefficients
        /// of the moments of the d_alpha/dt term
        /// <summary>
        private double Alpha_Derivk(int k, int i, ColumnVector Pvec)
        {
            double sum = 0;
            double product = 1;
            double c1 = Pvec[i] * Math.Pow(Pvec[i + 2 * nmodes], k -
1);

            for (int q = 2; q <= k; q++)
            {
                product = 1;
                for (int j = 2; j <= k; j++)
                {
                    if (j != q)
                        product = product * (k - j + Pvec[i + nmodes]);
                }
                sum += product;
            }
            return c1 * sum;
        }

        /// <summary>
        /// Private method that returns the derivative coefficients
        /// of the moments of the d_beta/dt term
        /// <summary>
        private double Beta_Derivk(int k, int i, ColumnVector Pvec)
        {
            double product = 1;
            double c1 = (k - 1) * Pvec[i] * Math.Pow(Pvec[i + 2 *
nmodes], k - 2);

            for (int j = 2; j <= k; j++)
            {
                product = product * (k - j + Pvec[i + nmodes]);
            }

            return c1 * product;
        }

        /// <summary>
        /// Private method that returns the matrix containing
        /// the derivate coefficients of the moments
        /// <summary>
        private SquareMatrix Matrix_Deriv(ColumnVector Pvec)
        {
            SquareMatrix Deriv_Coeff_Mat = new SquareMatrix(3 *
nmodes);

            for (int rows = 0; rows < 3 * nmodes; rows++)
            {
                for (int cols = 0; cols < nmodes; cols++)
                {
                    Deriv_Coeff_Mat[rows, cols] = Wi_Derivk(rows + 2,
cols, Pvec);
                    Deriv_Coeff_Mat[rows, cols + nmodes] =
Alpha_Derivk(rows + 2, cols, Pvec);
                    Deriv_Coeff_Mat[rows, cols + 2 * nmodes] =
Beta_Derivk(rows + 2, cols, Pvec);
                }
            }
            return Deriv_Coeff_Mat;
        }

        /// <summary>
        /// Public method that returns the right hand side
        /// of the coagulation equation
        /// <summary>
        public ColumnVector RHS(ColumnVector VI, ColumnVector VJ,
ColumnVector Pvec)
        {
            ColumnVector I = new ColumnVector(3 * nmodes);
            ColumnVector J = new ColumnVector(3 * nmodes);

            for (int rows = 0; rows < 3 * nmodes; rows++)
            {
                double I_sum = 0;
                double J_sum = 0;
                for (int i = 0; i < nmodes; i++)
                {
                    for (int j = 0; j < nmodes; j++)
                    {
                        double I1ijk = I1k(rows + 2, i, j, Pvec);
                        double I2ijk = I2k(rows + 2, i, j, Pvec);
                        I_sum += Pvec[i] * Pvec[j] * (I1ijk - I2ijk);
                    }
                    double J1ik = J1k(rows + 2, i, Pvec);
                    double J2ik = J2k(rows + 2, i, Pvec);
                    J_sum += Pvec[i] * (J1ik - J2ik);
                }
                I[rows] = I_sum;
                J[rows] = J_sum;
            }
            return Mult(VI, I) + Mult(VJ, J);
        }

        /// <summary>
        /// Public method that returns the results from the
        /// RK4 method
        /// <summary>
        private ColumnVector RK4(SquareMatrix DerivCoeffs, ColumnVector
PreviousVector, ColumnVector Vi, ColumnVector Vj, double h,
ColumnVector Signs)
        {
            ColumnVector NextVector = new ColumnVector(3 * nmodes);
            ColumnVector k1 = new ColumnVector(3 * nmodes);
            ColumnVector k2 = new ColumnVector(3 * nmodes);
            ColumnVector k3 = new ColumnVector(3 * nmodes);
            ColumnVector k4 = new ColumnVector(3 * nmodes);
            ColumnVector slope = new ColumnVector(3 * nmodes);
            SquareMatrix Minv = DerivCoeffs.Inverse();

            k1 = Minv * RHS(Vi, Vj, PreviousVector);
            k2 = Minv * RHS(Vi, Vj, PreviousVector + 0.5 * h * k1);
            k3 = Minv * RHS(Vi, Vj, PreviousVector + 0.5 * h * k2);
            k4 = Minv * RHS(Vi, Vj, PreviousVector + h * k3);

            slope = (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);

            for (int i = 0; i < 3 * nmodes; i++)
            {
                if (Signs[i] * slope[i] < 0)
                    slope[i] = -slope[i];
                else if (Signs[i] * slope[i] == 0)
                    slope[i] = 0;
            }

            NextVector = PreviousVector + slope;

            return NextVector;
        }

        /// <summary>
        /// Internal function that provides the time evolution of the
        /// w, alpha and beta variables through the use of the RK4
method.
        /// <summary>
        private void Normalize(ColumnVector Pvec)
        {
            double sumwi = 0;
            for (int i = 0; i < nmodes; i++)
```

```
                sumwi += Pvec[i];

            if (sumwi != 1)
            {
                for (int i = 0; i < nmodes; i++)
                    Pvec[i] = Math.Pow(sumwi, -1) * Pvec[i];
            }
        }

        /// <summary>
        /// Public method that returns the time evolution
        /// of the coagulation equation
        /// <summary>
        public List<ColumnVector> TimeEvolution(ColumnVector Vi,
ColumnVector Vj, double h, int timesteps, ColumnVector Signs)
        {
            ColumnVector Vn = IniVec.Clone();
            List<ColumnVector> TEv = new List<ColumnVector>();
            TEv.Add(Vn);

            for (int t = 1; t <= timesteps; t++)
            {
                ColumnVector Vnp1 = new ColumnVector(3*nmodes);
                SquareMatrix Mderivs = new SquareMatrix(3*nmodes);
                Mderivs = Matrix_Deriv(Vn);
                Vnp1 = RK4(Mderivs, Vn, Vi, Vj, h, Signs);
                Normalize(Vnp1);
                TEv.Add(Vnp1);
                Vn = Vnp1.Clone();
            }
            return TEv;
        }
        /// <summary>
        /// Public method that returns the cumulative gamma
        /// distribution function
        /// <summary>
        public double GammaCFD(double Diameter, ColumnVector Pvec)
        {
            double CDF = 0;

            for (int i = 0; i < nmodes; i++)
            {
                double cdfi = AdvancedMath.LeftGamma(Pvec[i + nmodes],
Diameter / Pvec[i + 2 * nmodes]);
                CDF += Pvec[i] * cdfi;
            }
            return 100 * CDF;
        }

        //End of Multimodal Class_______________
    }
}
```

2) The Genetic Algorithm

```
namespace Chapter4_Code_Multimodal
{
    class Multimodal_GA
    {
        /// <summary>
        /// Static function that returns the index of the
        /// maximum value of a column vector
        /// </summary>
        public static int IndexofMax(ColumnVector theVec)
        {
            int max = 0;
            int rows = theVec.Dimension - 1;
            int j = 1;

            while (j <= rows)
            {
                if (theVec[j] > theVec[max])
                {
                    max = j;
                    j += 1;
                }
                else
                    j += 1;
            }
            return max;
        }

        /// <summary>
        /// Static function that returns the index of the
        /// minimum value of a column vector
        /// </summary>
        public static int IndexofMin(ColumnVector theVec)
        {
            int min = 0;
            int rows = theVec.Dimension - 1;
            int j = 1;

            while (j <= rows)
            {
                if (theVec[j] < theVec[min])
                {
                    min = j;
                    j += 1;
                }
                else
                    j += 1;
            }
            return min;
        }

        /// <summary>
        /// A class that defines the Chromosome type
        /// </summary>
        public class Chromosome
        {
            /// <summary>
            /// Members
            /// </summary>
            ColumnVector[] Chrom;

            /// <summary>
            /// Constructors
            /// </summary>
            public Chromosome(ColumnVector AggregationVector,
ColumnVector FragmentationVector)
            {
                Chrom = new ColumnVector[2];
                Chrom[0] = AggregationVector.Clone();
                Chrom[1] = FragmentationVector.Clone();
            }
            /// <summary>
            /// Properties
            /// </summary>
            public ColumnVector Aggregation_Vector
            {
                get
                { return Chrom[0]; }
            }
            public ColumnVector Fragmentation_Vector
            {
                get
                { return Chrom[1]; }
            }

            /// <summary>
            /// Methods
            /// </summary>
            public Chromosome Clone()
            {
                Chromosome cloned = new
Chromosome(this.Aggregation_Vector, this.Fragmentation_Vector);
                return cloned;
            }
            private double GaussianRandom()
            {
                Random rng = new Random();
                Distribution dist = new NormalDistribution(0.5, 1);
                double y = rng.NextDouble();
                double x = Math.Abs(dist.InverseLeftProbability(y));
                return x;
            }
            public Chromosome Mutate()
            {
                //Random x = new Random();
                int dim = this.Aggregation_Vector.Dimension;
                ColumnVector AggMutated = new ColumnVector(dim);
                ColumnVector FragMutated = new ColumnVector(dim);

                for (int i = 0; i < dim; i++)
                {
                    AggMutated[i] = GaussianRandom() *
this.Aggregation_Vector[i];
                    FragMutated[i] = GaussianRandom() *
this.Fragmentation_Vector[i];
                }
                Chromosome mutated = new Chromosome(AggMutated,
FragMutated);

                return mutated;
            }
            public Chromosome Crossover(Chromosome Parent2)
            {
                int rows = Parent2.Aggregation_Vector.Dimension;
                ColumnVector NewAgg = this.Aggregation_Vector.Clone();
                ColumnVector NewFrag =
this.Fragmentation_Vector.Clone();

                for (int i = 0; i < rows; i += 2)
                {
                    NewAgg[i] = Parent2.Aggregation_Vector[i];
                    NewFrag[i] = Parent2.Fragmentation_Vector[i];
                }
                Chromosome Crossed = new Chromosome(NewAgg, NewFrag);

                return Crossed;
            }
        }

        /// <summary>
        /// A class that defines the population type
        /// </summary>
        public class Population
        {
            /// <summary>
            /// Members
            /// </summary>
            List<Chromosome> theChromosomes = new List<Chromosome>();

            /// <summary>
            /// Properties
            /// </summary>
            public int Count
            {
                get
                { return theChromosomes.Count; }
            }

            /// <summary>
            /// Methods
            /// </summary>
            /// <summary>
            public void Add(Chromosome tobeadded)
            {
                theChromosomes.Add(tobeadded);
            }
            public void RemoveAt(int index)
            {
                theChromosomes.RemoveAt(index);
            }
            public Chromosome Chromosome(int index)
            {
                return theChromosomes[index];
            }
            public Chromosome Population_Average()
            {
                int PopLength = this.Count;
                int dim =
this.Chromosome(0).Aggregation_Vector.Dimension;

                ColumnVector sumAgg = new ColumnVector(dim);
                ColumnVector sumFrag = new ColumnVector(dim);

                for (int i = 0; i < PopLength; i++)
                {
                    sumAgg += this.Chromosome(i).Aggregation_Vector;
                    sumFrag += this.Chromosome(i).Fragmentation_Vector;
                }

                sumAgg = (1 / PopLength) * sumAgg;
                sumFrag = (1 / PopLength) * sumFrag;
                Chromosome Average = new Chromosome(sumAgg, sumFrag);

                return Average;
            }
            public Population Remove10Worst(ColumnVector
Fitness_Vector)
            {
                int d = Fitness_Vector.Dimension;
                ColumnVector fitaux = Fitness_Vector.Clone();
                int[] indices = new int[d];
                Population theBest = new Population();
                List<int> indcomp = new List<int>();

                for (int i = 0; i < 20; i++)
                {
                    indcomp.Add(i);
                }

                for (int j = 0; j < 10; j++)
                {
                    indices[j] =
Unimodal_SteadyState.IndexofMax(fitaux);
                    fitaux[indices[j]] = -1000;
                    indcomp.Remove(indices[j]);
                }

                for (int k = 0; k < 10; k++)
                {
                    theBest.Add(this.Chromosome(indcomp[k]));
                }
                return theBest;
            }
        }
        /// The Genetic Algorithm to obtain the optimum parameters
        /// for the steady-state unimodal condition
        /// <summary>
        /// Members
        /// </summary>
        int num_modes;
        Multimodal MultimodalObject;
        ColumnVector InitialVec;
        private List<double[]> TargetInput = new List<double[]>();
        ColumnVector SignsVec;
        int p_norm;
        double h;
        int timesteps;

        /// <summary>
        /// Constructor
        /// </summary>
        public Multimodal_GA(int numberofmodes, Multimodal 
aMultimodalObject, int pnormCoeff, ColumnVector ControlVector, 
List<double[]> InputFlocData, ColumnVector SignsVector, double 
time_interval, int time_steps)
        {
            num_modes = numberofmodes;
            MultimodalObject = aMultimodalObject;
            p_norm = pnormCoeff;
            TargetInput = InputFlocData;
            InitialVec = ControlVector.Clone();
            timesteps = time_steps;
            h = time_interval;
            SignsVec = SignsVector.Clone();
        }

        /// <summary>
        /// Methods
        /// </summary>
        private double LpNorm(ColumnVector Vector)
        {
            double norm = 0;
            int dim = Vector.Dimension;

            for (int i = 0; i < dim; i++)
            {
                norm += Math.Pow(Vector[i], p_norm);
            }

            return Math.Pow(norm, Math.Pow(p_norm, -1));
        }
        private double Fitness(ColumnVector Predicted_Vector, 
ColumnVector Observed_Vector)
        {
            return LpNorm(Predicted_Vector - Observed_Vector);
        }
        private ColumnVector GeneratePopFitness(List<double[]> 
InputFloc, Population thePopulation)
        {
            ColumnVector PopFitnesses = new 
ColumnVector(thePopulation.Count);
```

```
            for (int i = 0; i < thePopulation.Count; i++)
            {
                List<ColumnVector> ResList = new List<ColumnVector>();
                ResList =
MultimodalObject.TimeEvolution(thePopulation.Chromosome(i).Aggregation_
Vector, thePopulation.Chromosome(i).Fragmentation_Vector, h, timesteps,
SignsVec);
                //PopFitnesses[i] = Fitness(ResList[ResList.Count - 1],
TargetVec);
                ColumnVector Temp = ResList[ResList.Count - 1].Clone();
                bool isNAN = false;
                for (int k = 0; k < Temp.Dimension; k++)
                {
                    if (double.IsNaN(Temp[k]))
                    {
                        isNAN = true;
                        break;
                    }
                }
                bool isBad = false;
                for (int j = 0; j < num_modes; j++)
                {
                    if (Temp[j] < 0)
                    {
                        isBad = true;
                        break;
                    }
                }
                if (!isNAN && !isBad)
                    PopFitnesses[i] = MultimodalObject.SEE(InputFloc,
ResList[ResList.Count - 1]);
                else
                    PopFitnesses[i] = 100000000;

                if ((double.IsNaN(PopFitnesses[i])) && isBad)
                    PopFitnesses[i] = 100000000;
            }
            return PopFitnesses;
        }

        //Main Routine that searchs for the optimum results
        public Chromosome GetOptimum(string GApath, int MaxSteps)
        {
            //The StreamWriter that will capture the results
            StreamWriter sw = new StreamWriter(GApath);
            sw.WriteLine("GA Algorithm");
            sw.WriteLine();
            sw.WriteLine("Iteration No.,Min. Fitness Val.");

            ColumnVector Aini = new ColumnVector(3 * num_modes);
            ColumnVector Bini = new ColumnVector(3 * num_modes);

            Aini[0] = 1.07489181643306E-08;
            Aini[1] = 4.36386425971222E-08;
            Aini[2] = 1.07489181643306E-08;
            Aini[3] = 4.36386425971222E-08;
            Aini[4] = 1.07489181643306E-08;
            Aini[5] = 4.36386425971222E-08;

            Bini[0] = 3.1318876813257E-10;
            Bini[1] = 1.27148914049075E-09;
            Bini[2] = 3.1318876813257E-10;
            Bini[3] = 1.27148914049075E-09;
            Bini[4] = 3.1318876813257E-10;
            Bini[5] = 1.27148914049075E-09;

            //Creating initial population
            Population Population = new Population();
            int N = 20;
            //Generating N chromosomes
            Chromosome MainChromosome = new Chromosome(Aini, Bini);
            Population.Add(MainChromosome);
            ColumnVector fitnesses = new ColumnVector(N);

            //Creating the initial Chromosome population (N-1)
            for (int i = 1; i < N; i++)
            {
                Random rnd = new Random();
                Chromosome NextChrom = Population.Chromosome(i -
1).Mutate();
                Population.Add(NextChrom);
            }

            //Defining the stop criteria
            double epsilon = 1;
            int count = 0;
            double minfit = 2 * epsilon;
            Chromosome Optimum = MainChromosome.Clone();

            do
            {
                fitnesses = GeneratePopFitness(TargetInput,
Population);
                minfit = fitnesses.Min();
                int indmin = IndexofMin(fitnesses);
                Optimum = Population.Chromosome(indmin).Clone();

                sw.WriteLine(count + "," + minfit);

                //Updating the population:
                //Removing the 10 worst from the population
                Population = Population.Remove10Worst(fitnesses);

                //Proceeding to breed to obtain the offsprings
                int numoffs = Population.Count - 1;
                int noff = 0;
                while (noff < numoffs)
                {
                    Chromosome p1 = Population.Chromosome(noff);
                    Chromosome p2 = Population.Chromosome(noff + 1);
                    noff += 2;
                    Chromosome offspring1 = p1.Crossover(p2);
                    Population.Add(offspring1);
                    Chromosome offspring2 = offspring1.Mutate();
                    Population.Add(offspring2);
                }

                //Verifying that we have a new population
                int npop = Population.Count;
                //if not add the remaining members through averaging
                while (npop < N)
                {
                    Chromosome newmember =
Population.Population_Average();
                    Population.Add(newmember);
                }

                //The new population should now be complete

                count += 1;
            } while ((minfit > epsilon) && (count <= MaxSteps));

            //int themin = IndexofMin(fitnesses);

            sw.WriteLine("Optimum Results:");
            sw.WriteLine("Aggregation Parameters");
            for (int i = 0; i < 3 * num_modes; i++)
            {
                sw.WriteLine(Optimum.Aggregation_Vector[i]);
            }
            sw.WriteLine("Fragmentation Diagonal");
            for (int i = 0; i < 3 * num_modes; i++)
            {

                sw.WriteLine(Optimum.Fragmentation_Vector[i] + ",");
            }
            sw.Close();

            return Optimum;
        }
    }
}
```

APPENDIX J: Observed vs. Predicted Flocculated PSD curves

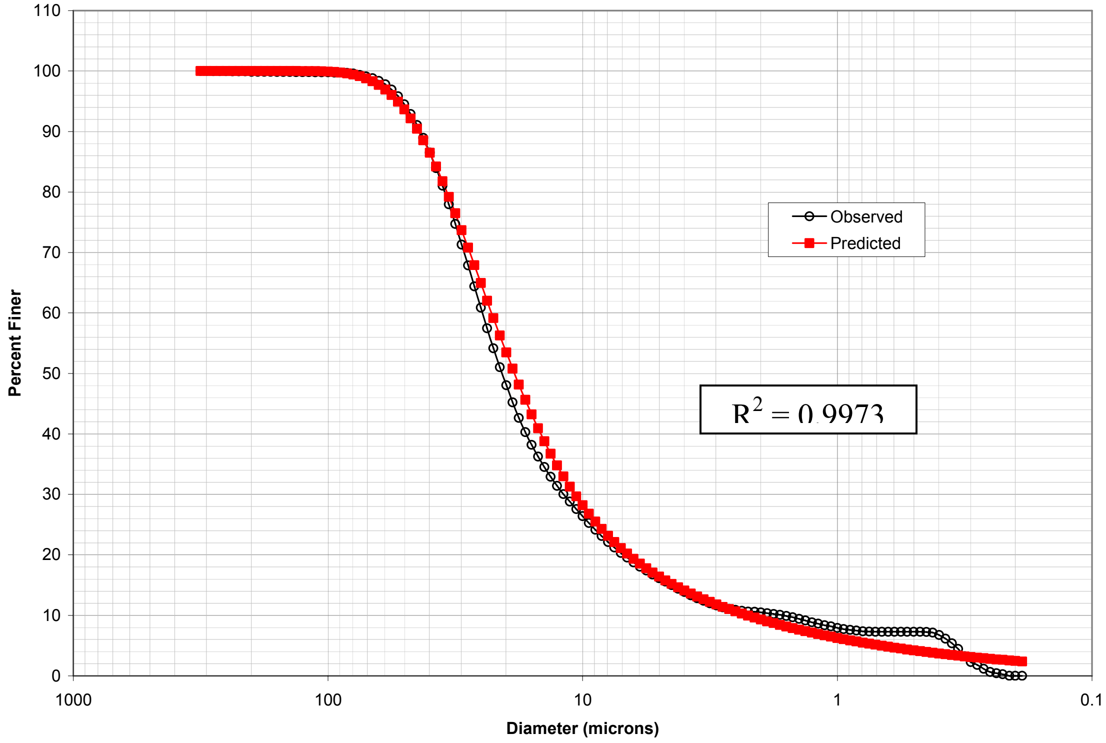

Figure 4 - 9: Observed vs. Predicted PSD for Sample A1 using Flocculant MF

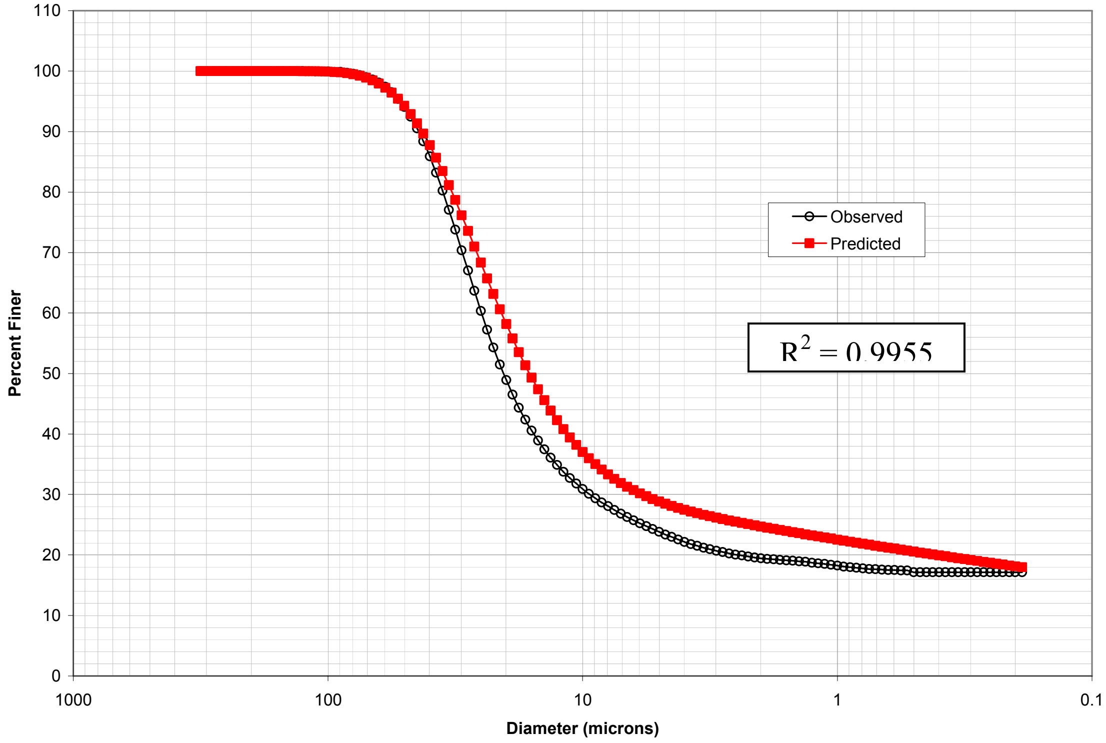

Figure 4 - 10: Observed vs. Predicted PSD for Sample A1 using Flocculant FP

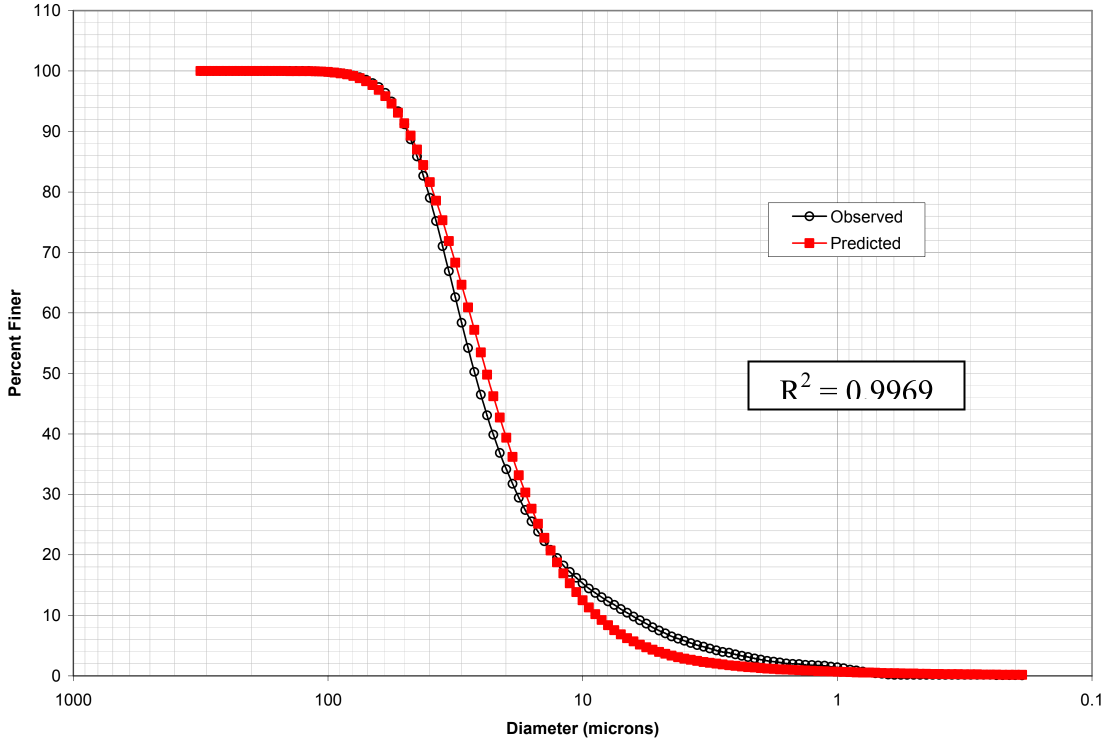

Figure 4 - 11: Observed vs. Predicted PSD for Sample A1 using Flocculant TF

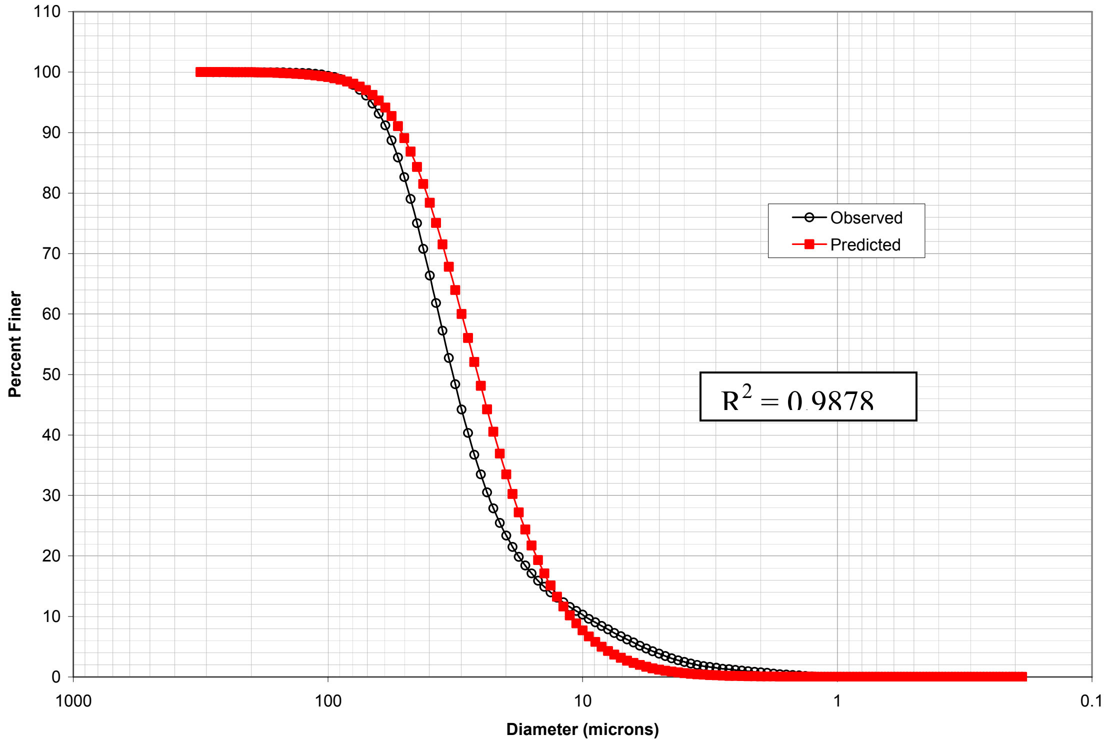

Figure 4 - 12: Observed vs. Predicted PSD for Sample A1 using Flocculant HF

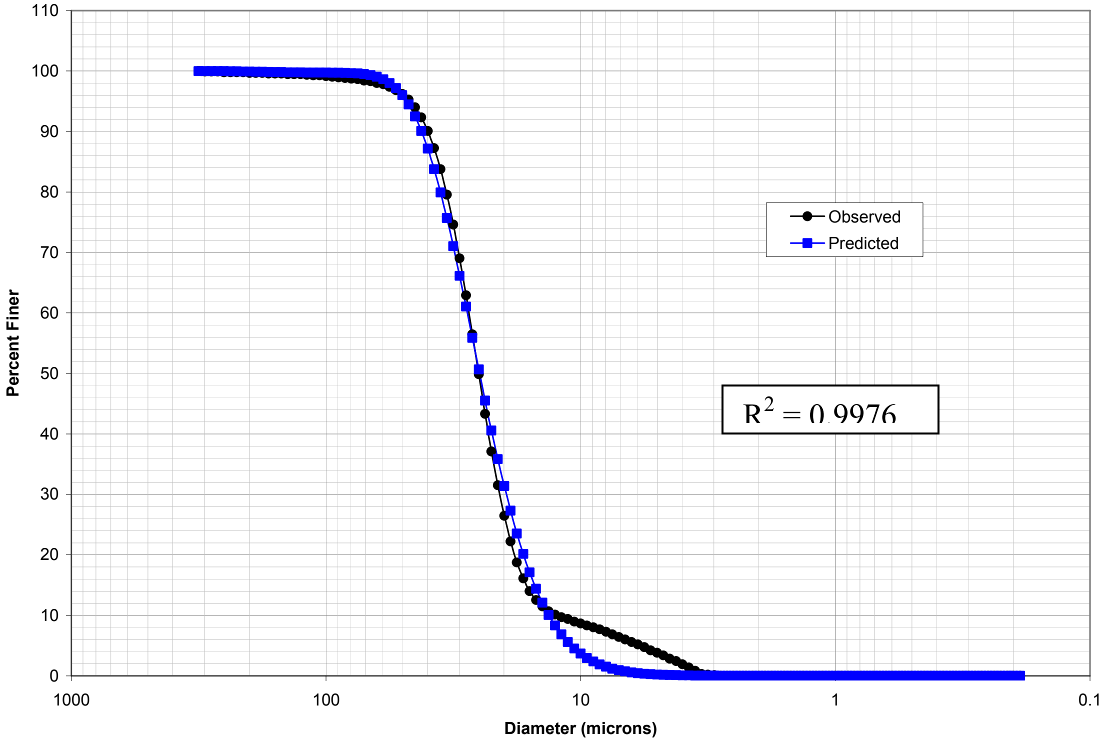

Figure 4 - 13: Observed vs. Predicted PSD for Sample A2 using Flocculant MF

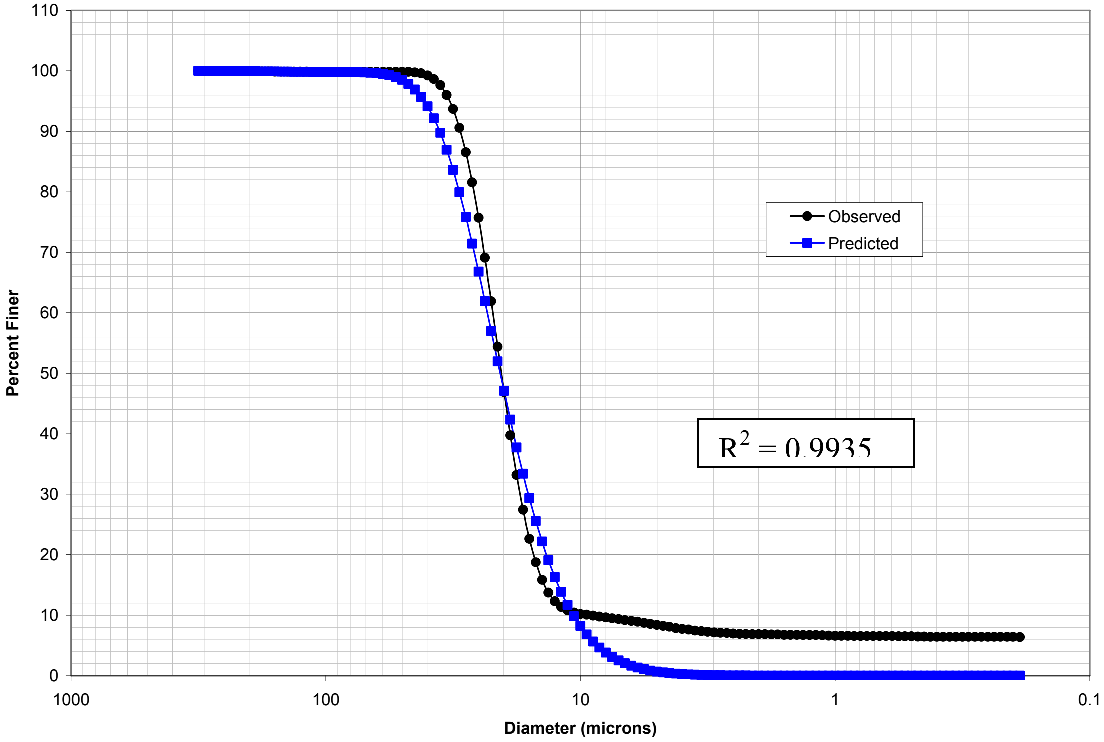

Figure 4 - 14: Observed vs. Predicted PSD for Sample A2 using Flocculant FP

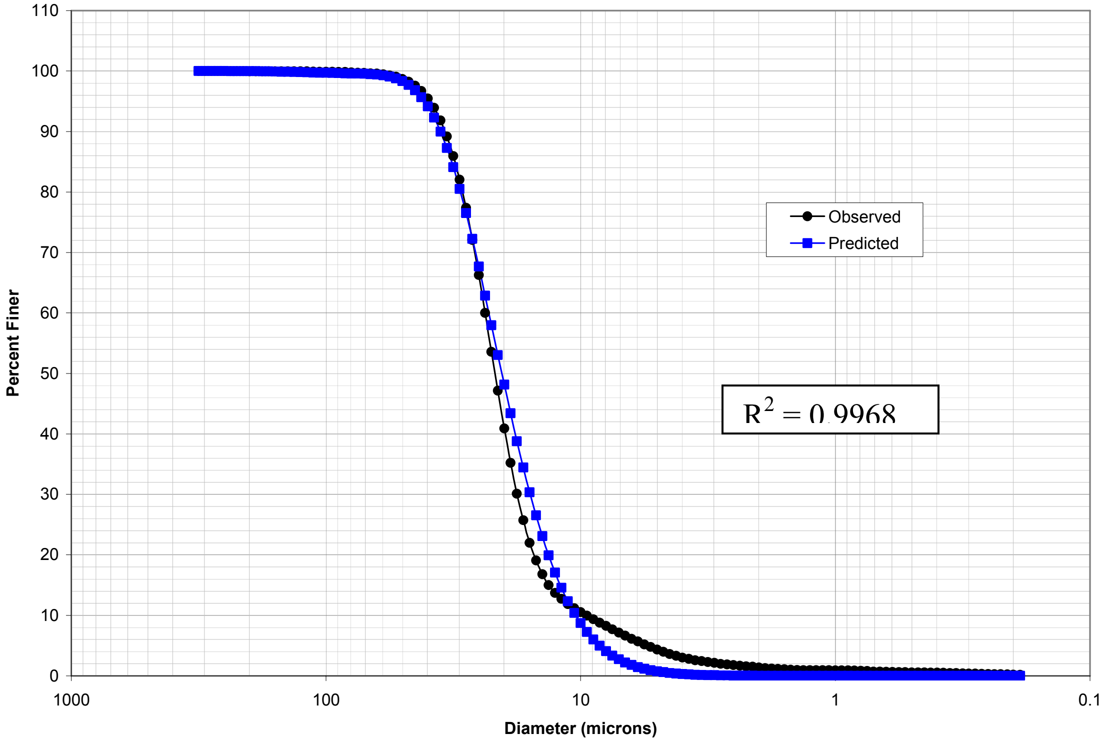

Figure 4 - 15: Observed vs. Predicted PSD for Sample A2 using Flocculant TF

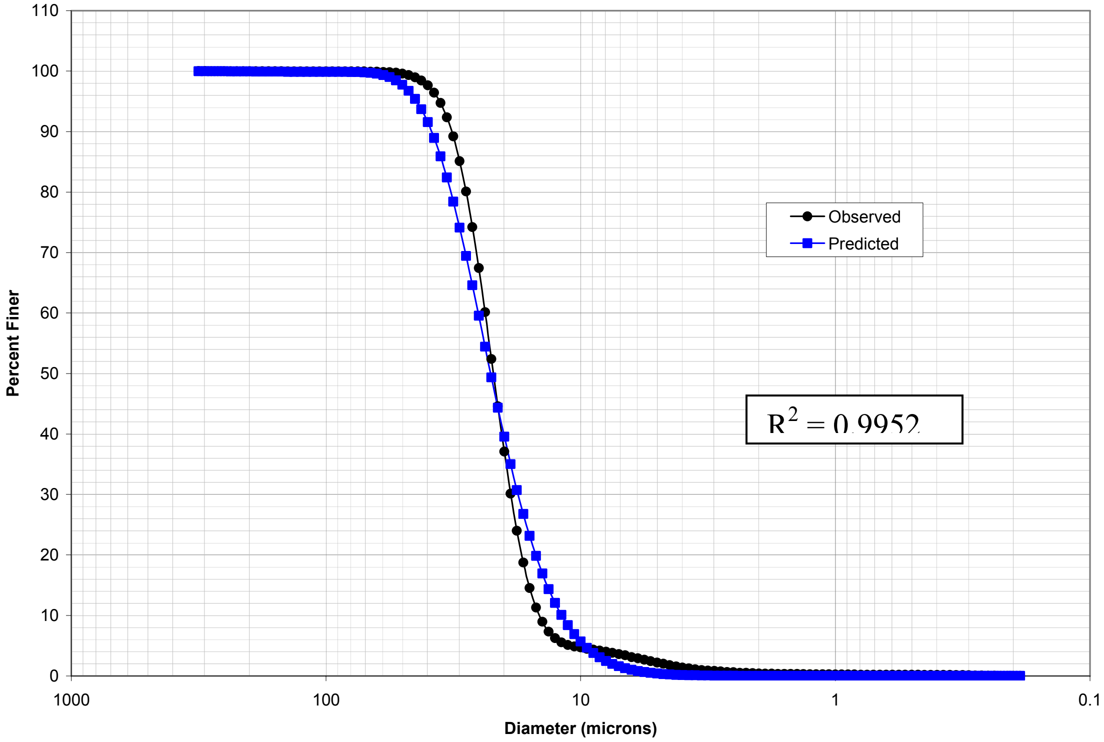

Figure 4 - 16: Observed vs. Predicted PSD for Sample A2 using Flocculant HF

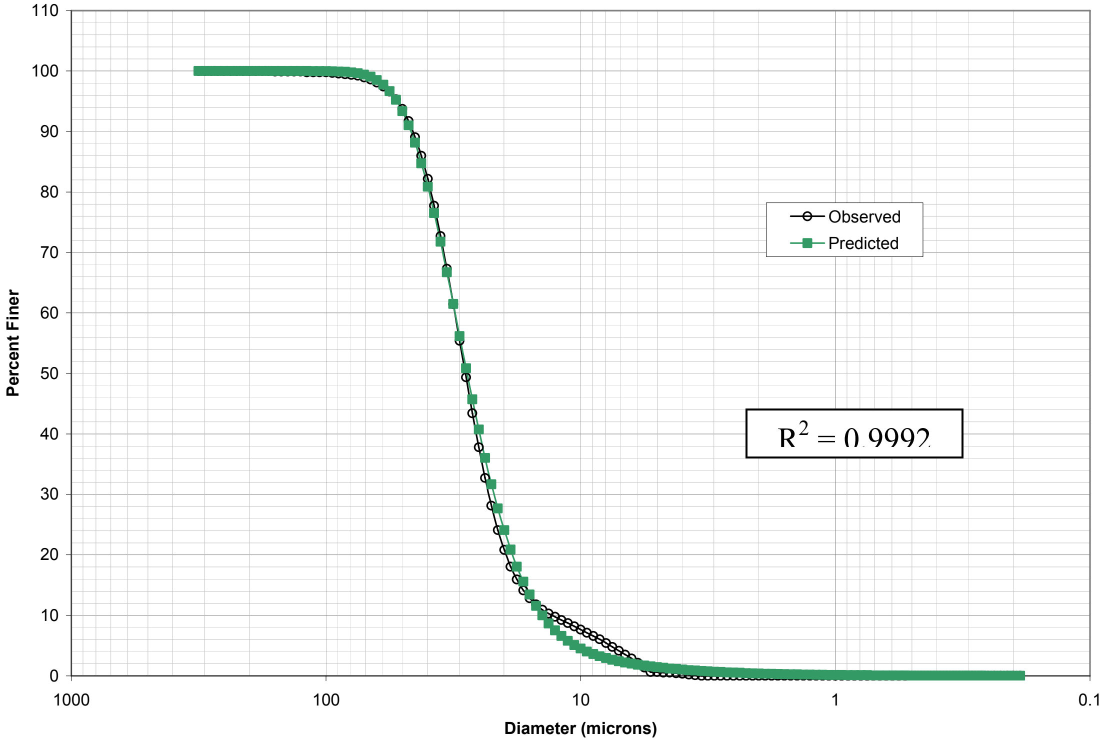

Figure 4 - 17: Observed vs. Predicted PSD for Sample A3 using Flocculant MF

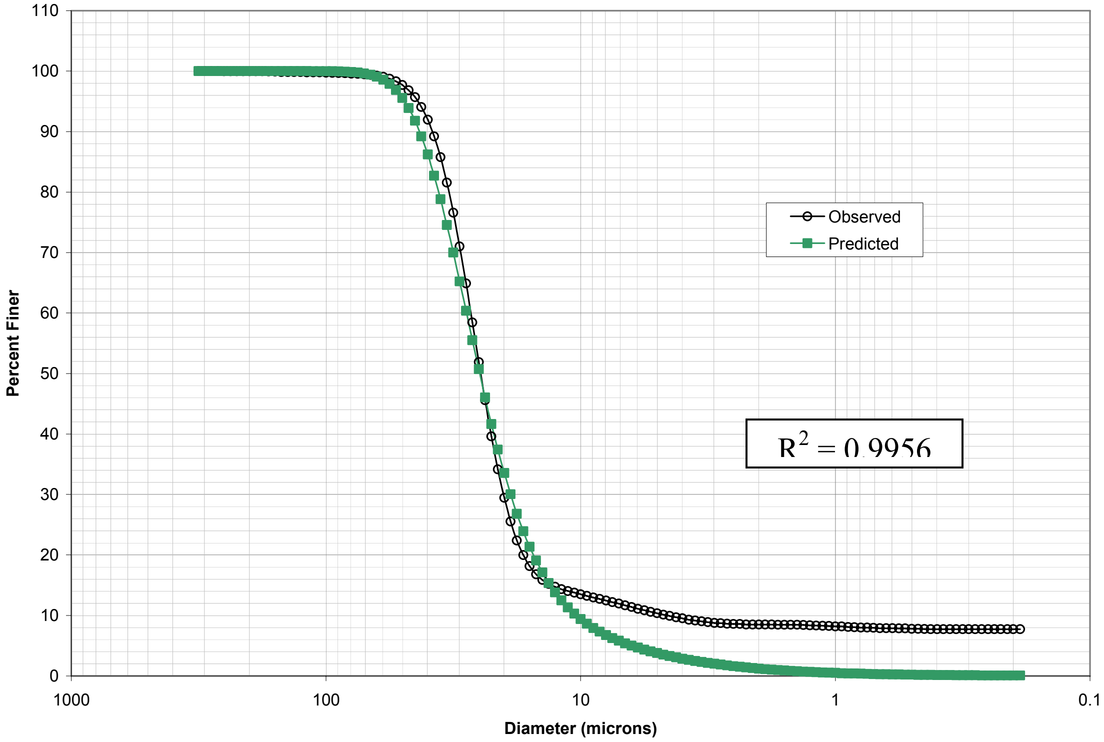

Figure 4 - 18: Observed vs. Predicted PSD for Sample A3 using Flocculant FP

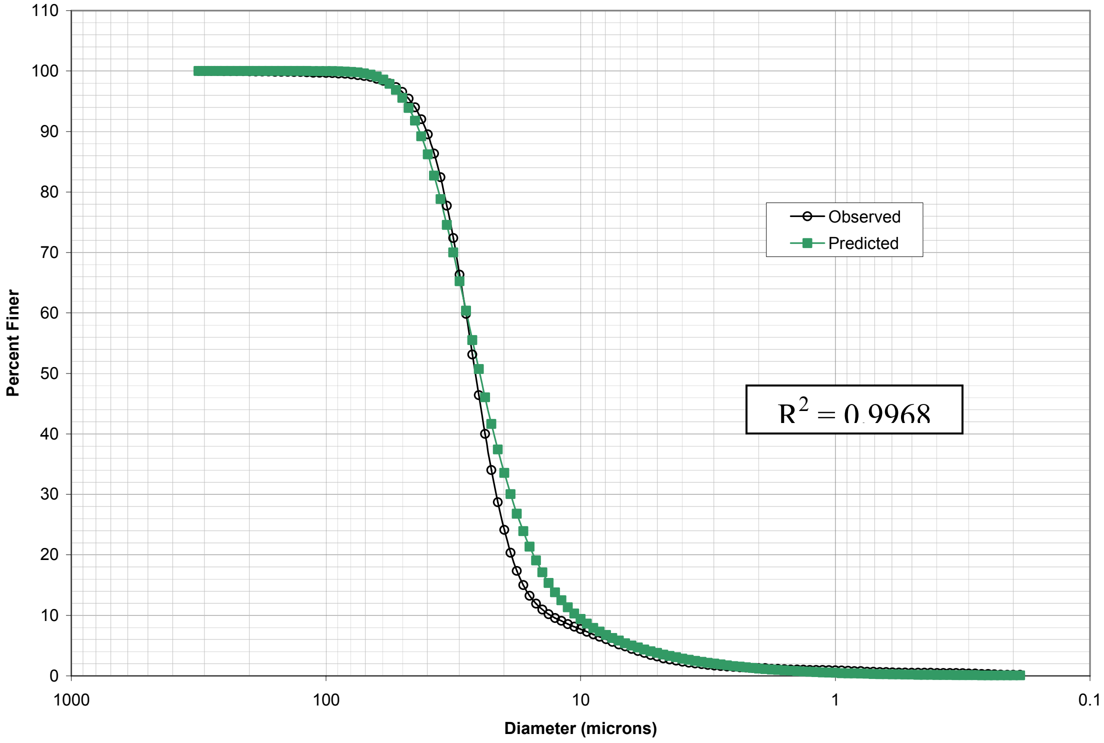

Figure 4 - 19: Observed vs. Predicted PSD for Sample A3 using Flocculant TF

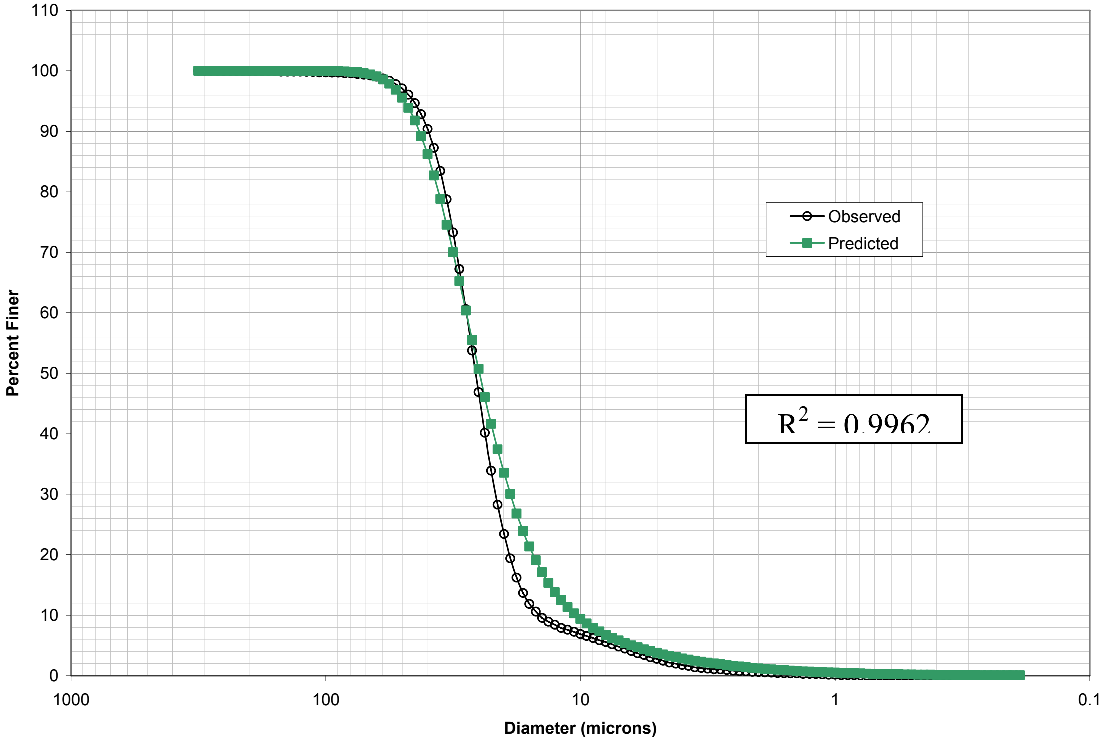

Figure 4 - 20: Observed vs. Predicted PSD for Sample A3 using Flocculant HF

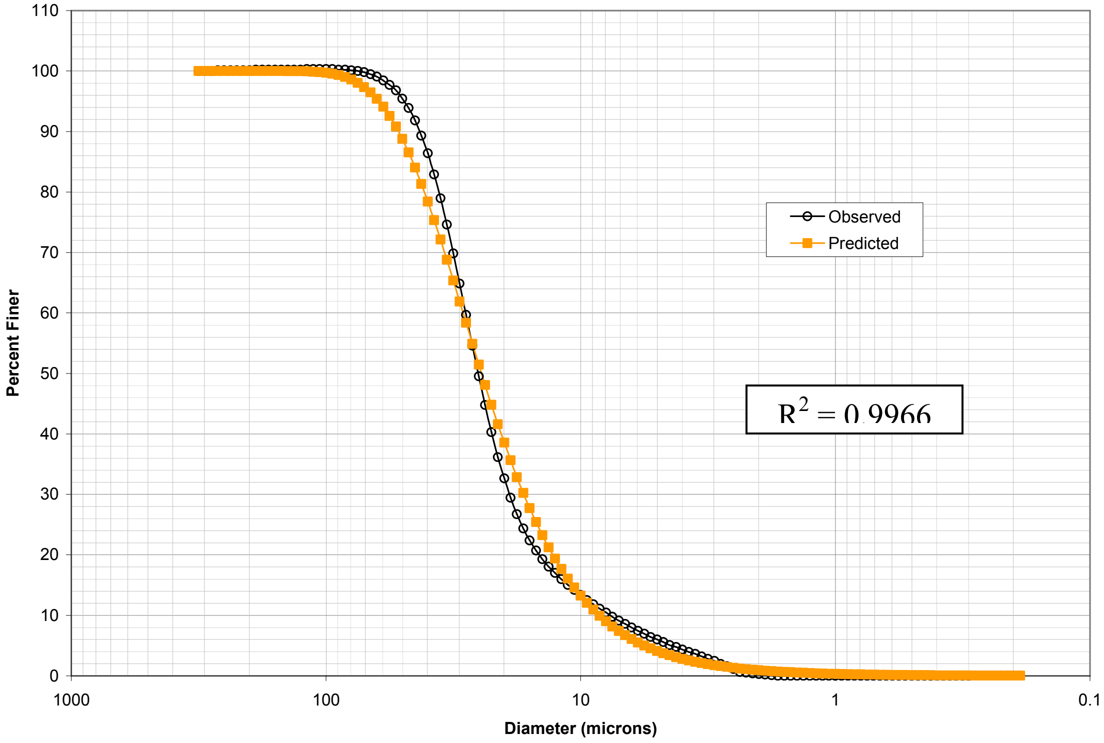

Figure 4 - 21: Observed vs. Predicted PSD for Sample A4 using Flocculant MF

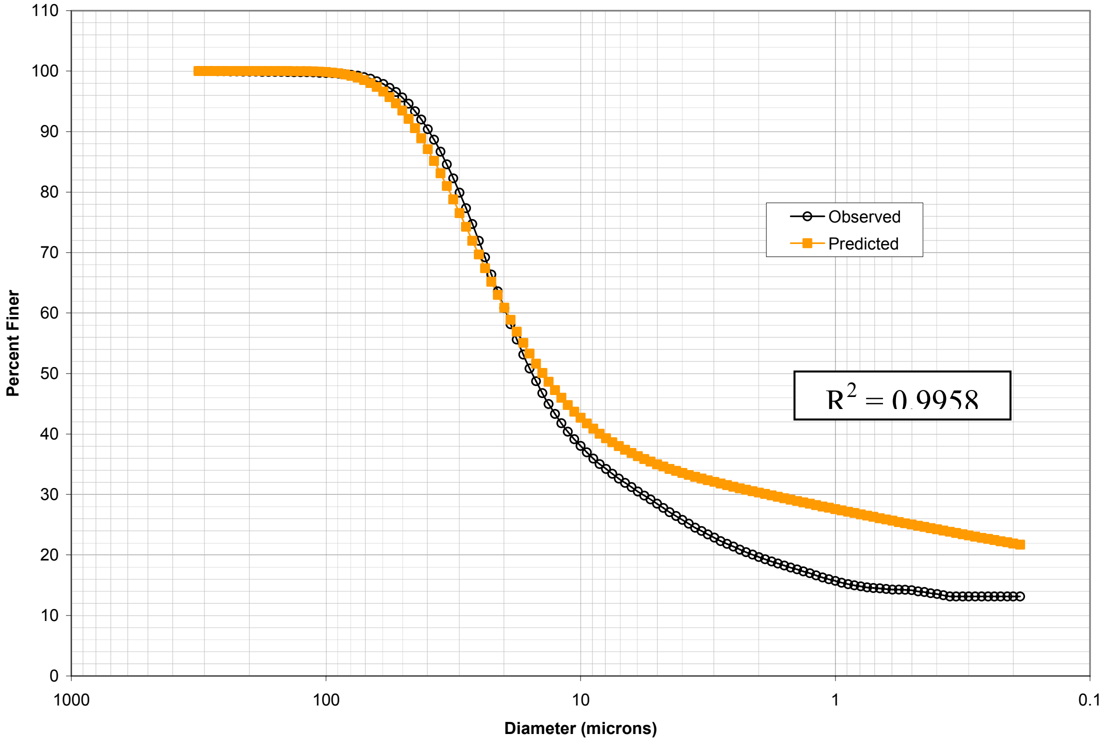

Figure 4 - 22: Observed vs. Predicted PSD for Sample A4 using Flocculant FP

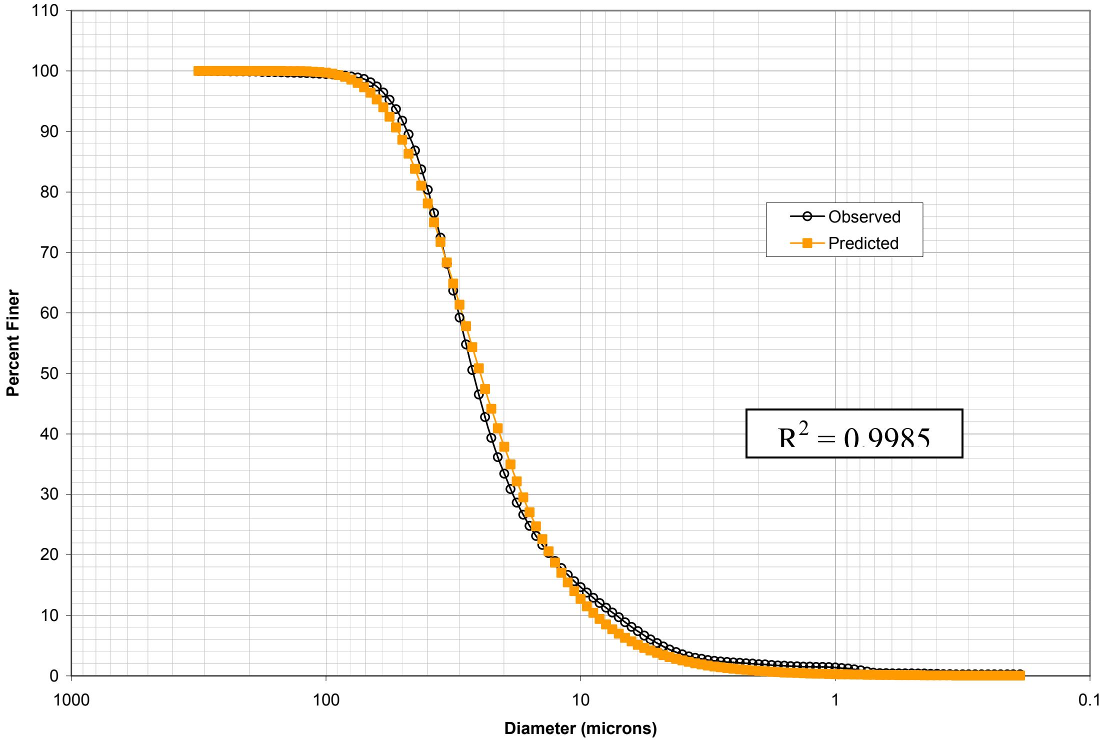

Figure 4 - 23: Observed vs. Predicted PSD for Sample A4 using Flocculant TF

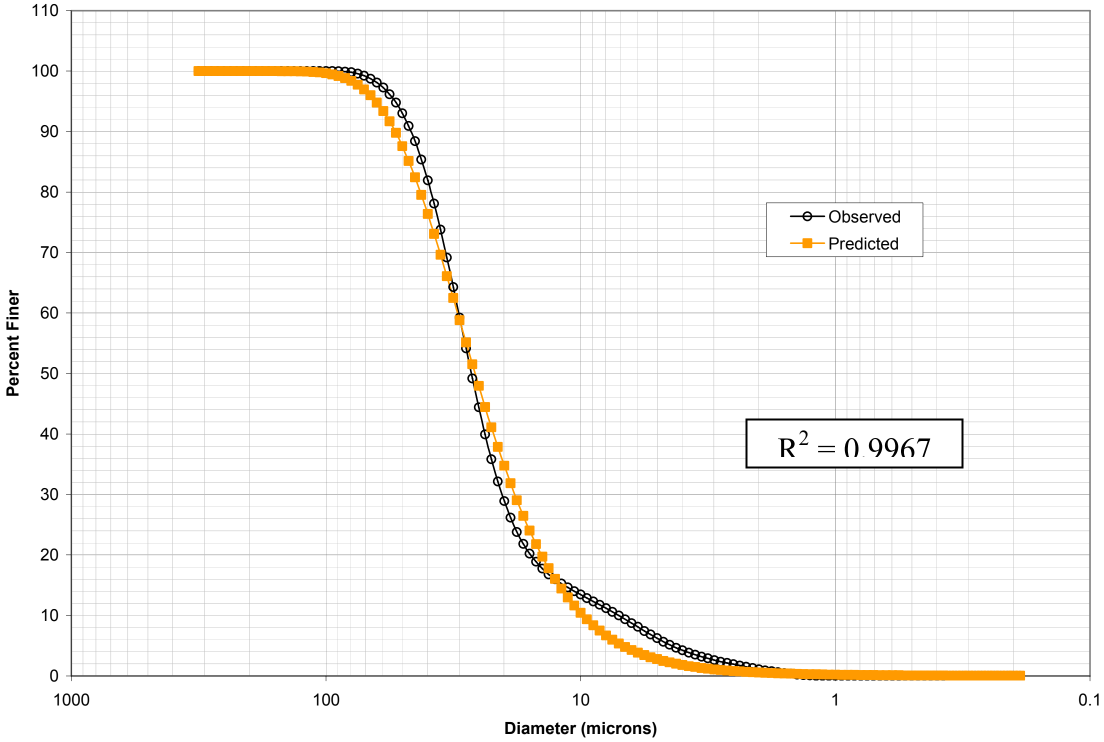

Figure 4 - 24: Observed vs. Predicted PSD for Sample A4 using Flocculant HF

APPENDIX K: Column Test Graphs – Small Glass Micro-Spheres

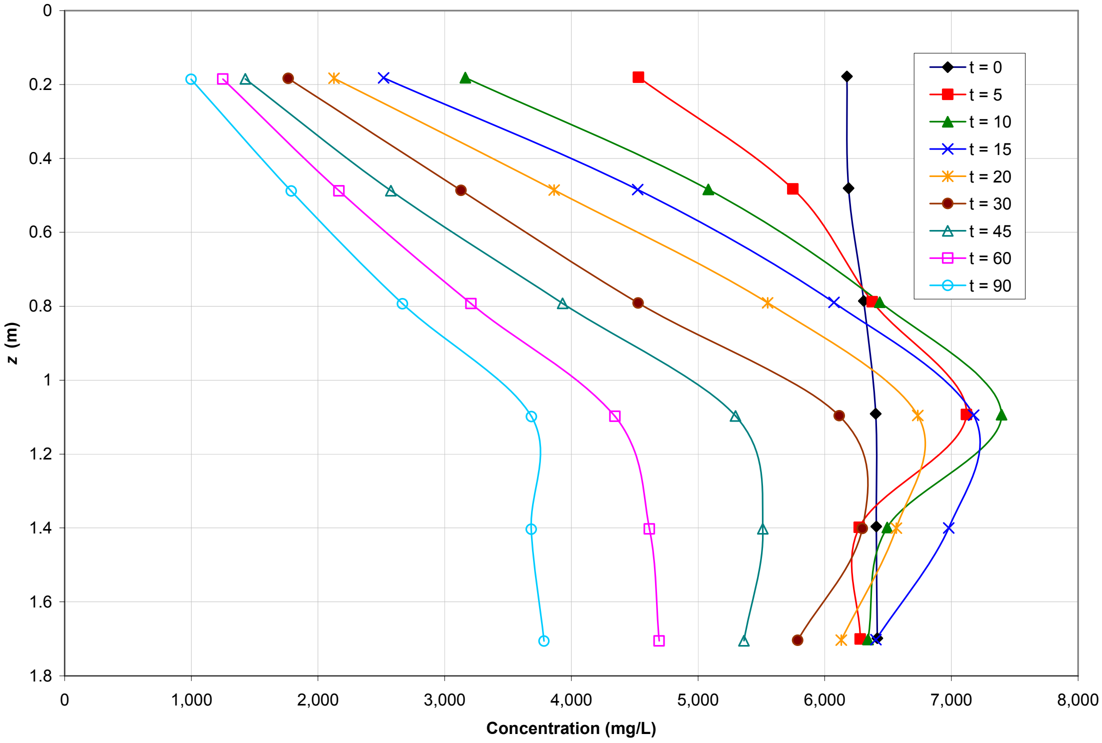

Figure 5 - 3: Small Micro-Spheres – Low Concentration – Replicate 1.

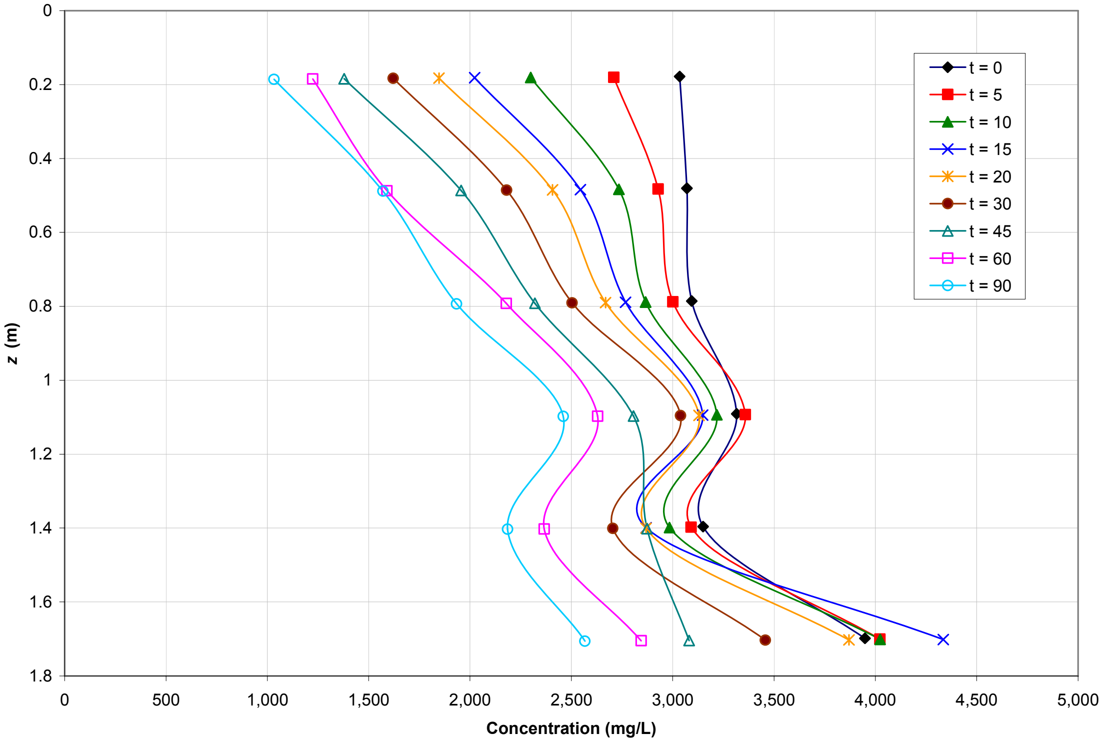

Figure 5 - 4: Small Micro-Spheres – Low Concentration – Replicate 2.

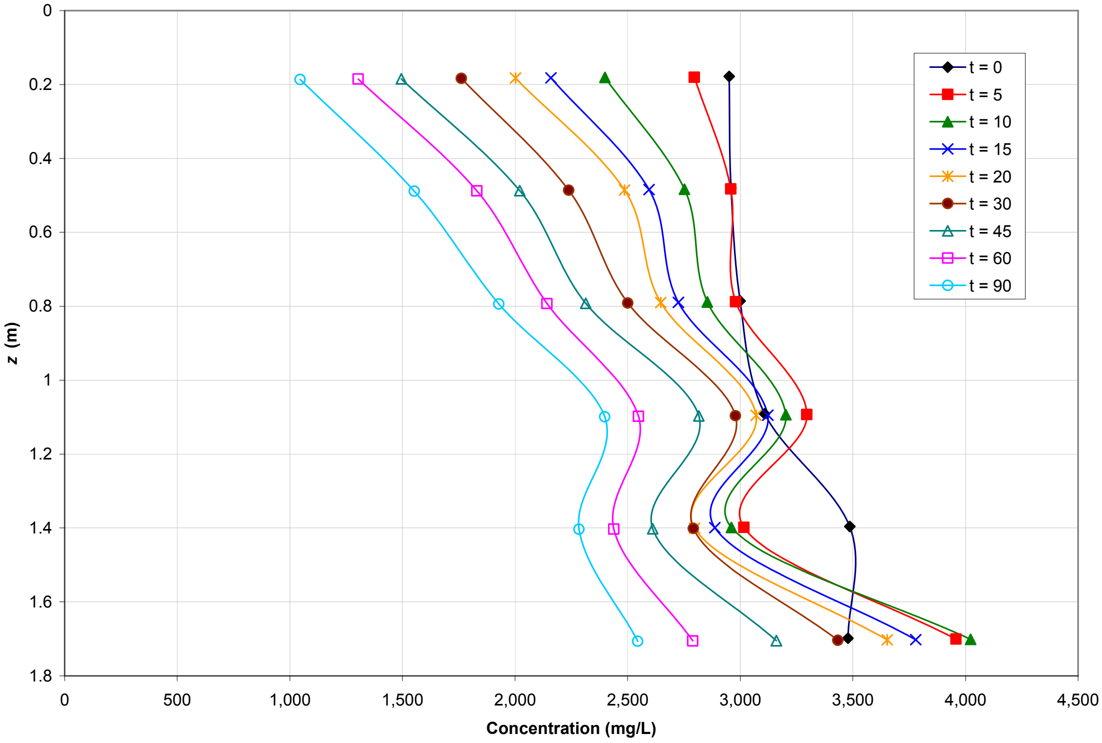

Figure 5 - 5: Small Micro-Spheres – Low Concentration – Replicate 3.

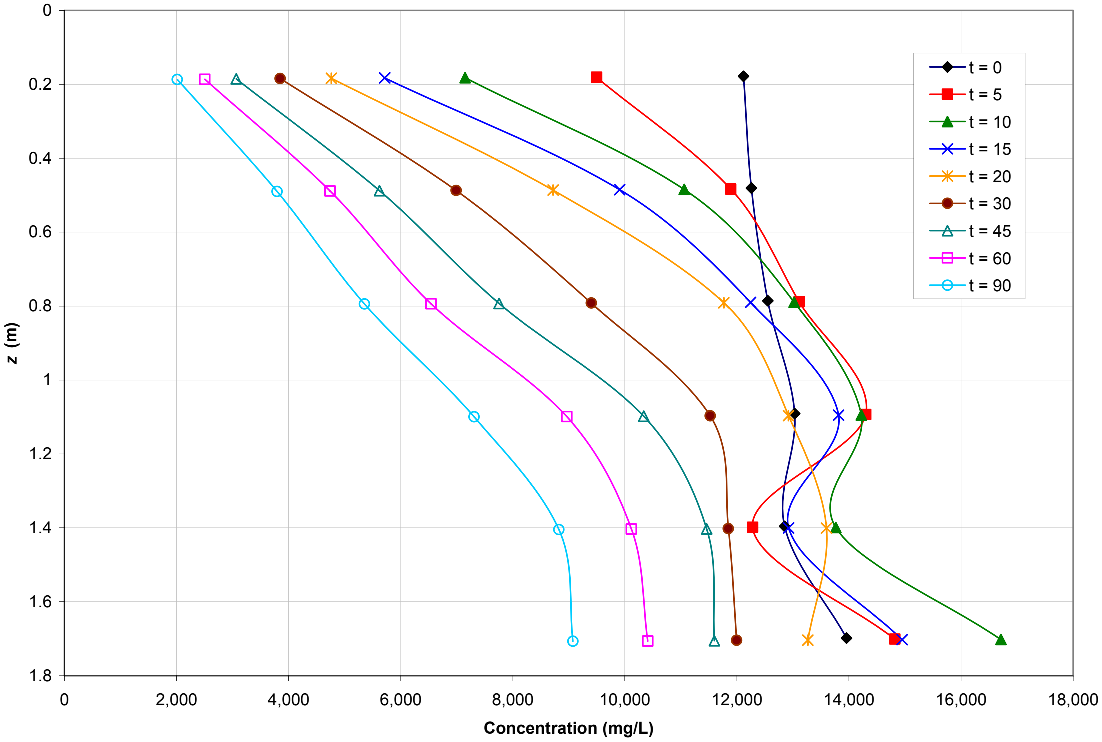

Figure 5 - 6: Small Micro-Spheres – Medium Concentration – Replicate 1.

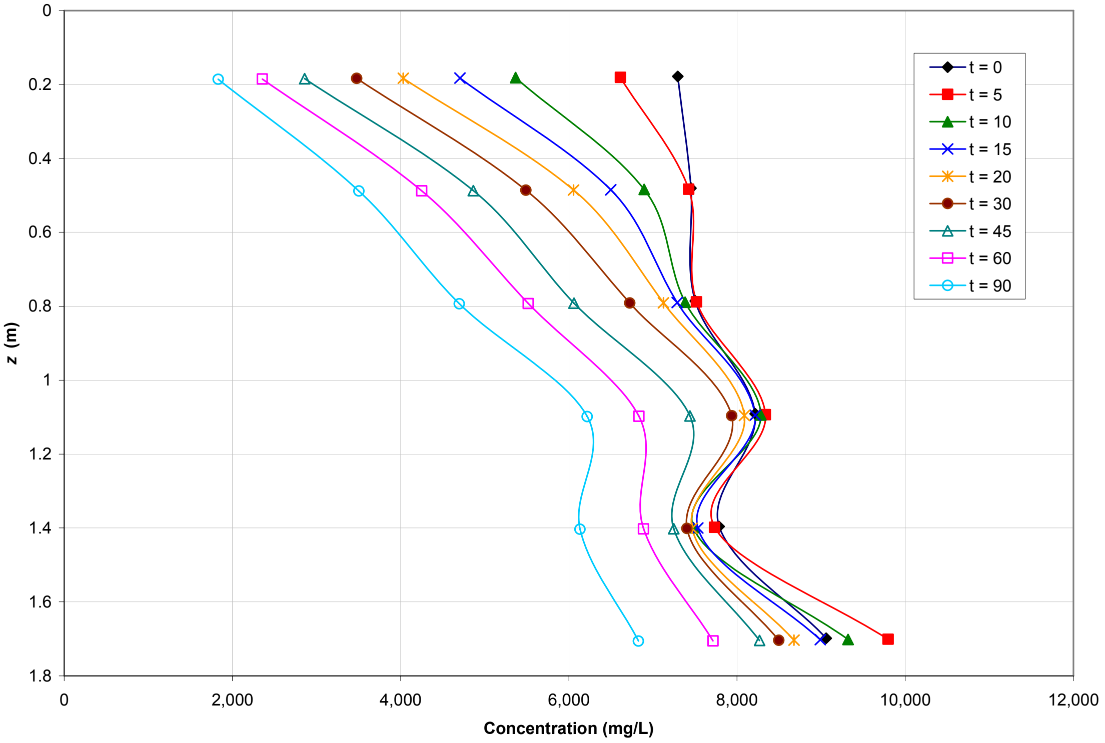

Figure 5 - 7: Small Micro-Spheres – Medium Concentration – Replicate 2.

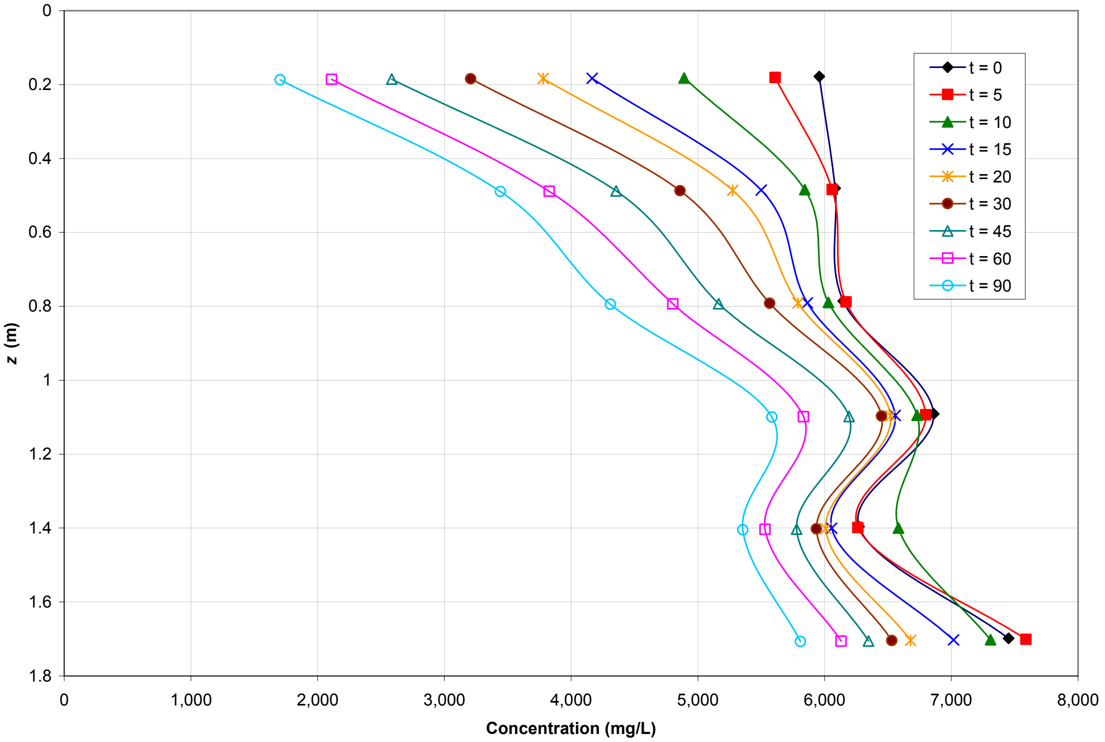

Figure 5 - 8: Small Micro-Spheres – Medium Concentration – Replicate 3.

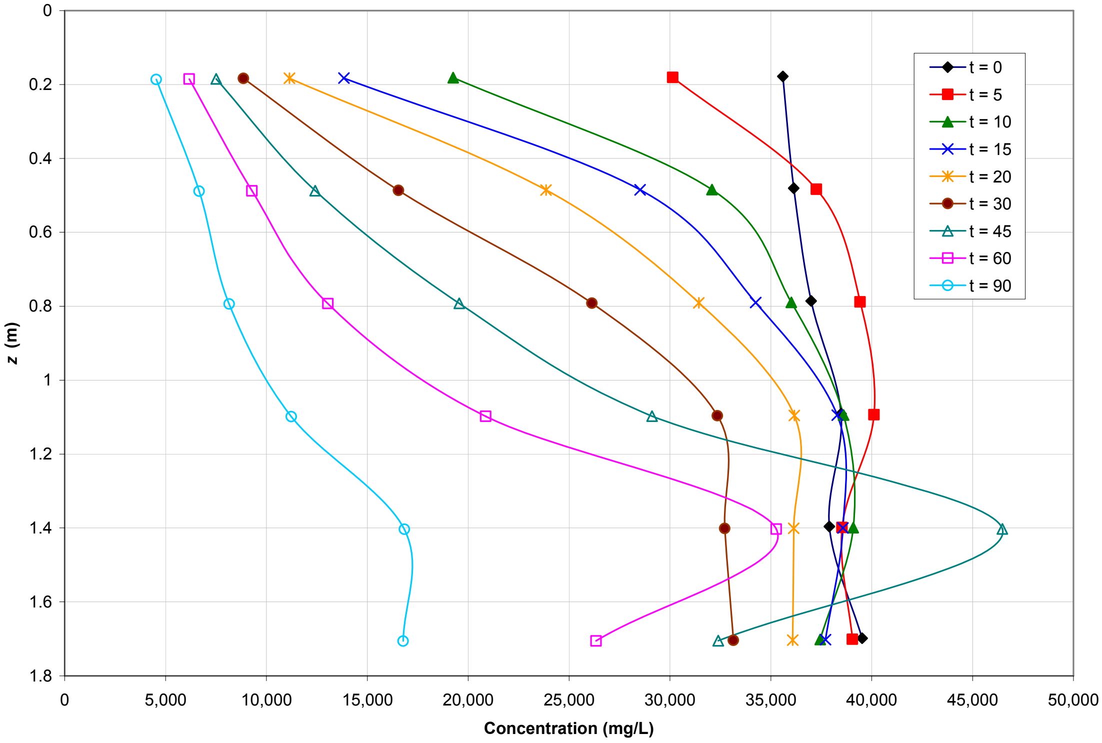

Figure 5 - 9: Small Micro-Spheres – High Concentration – Replicate 1.

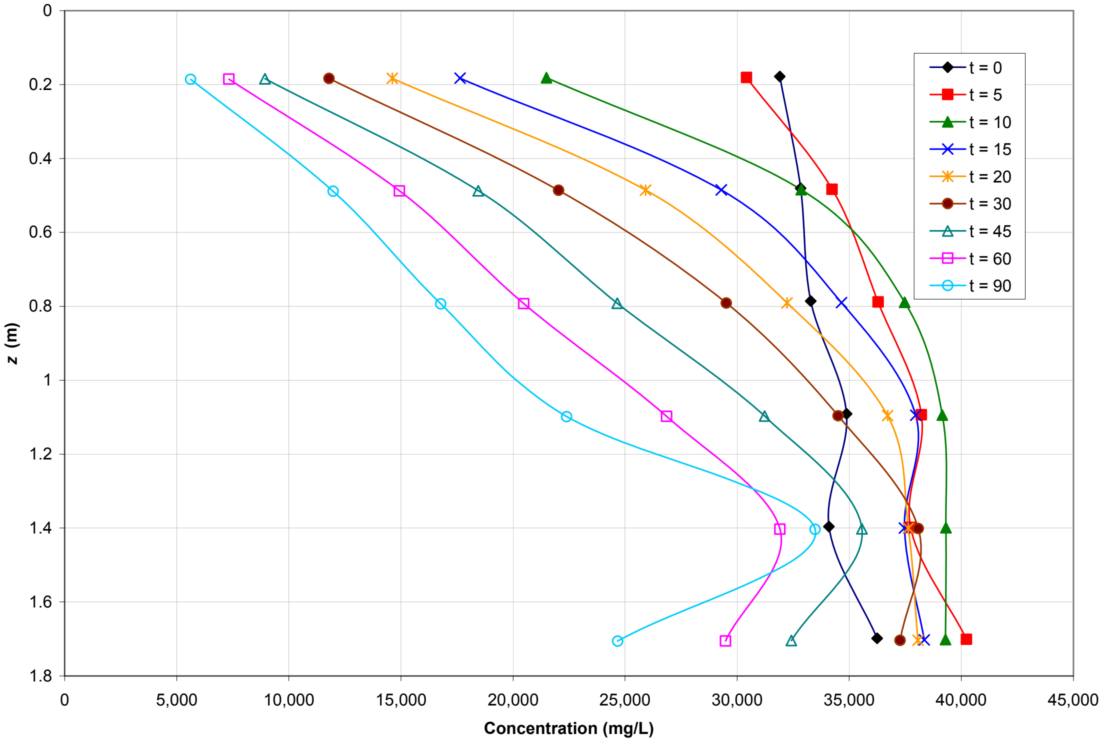

Figure 5 - 10: Small Micro-Spheres – High Concentration – Replicate 2.

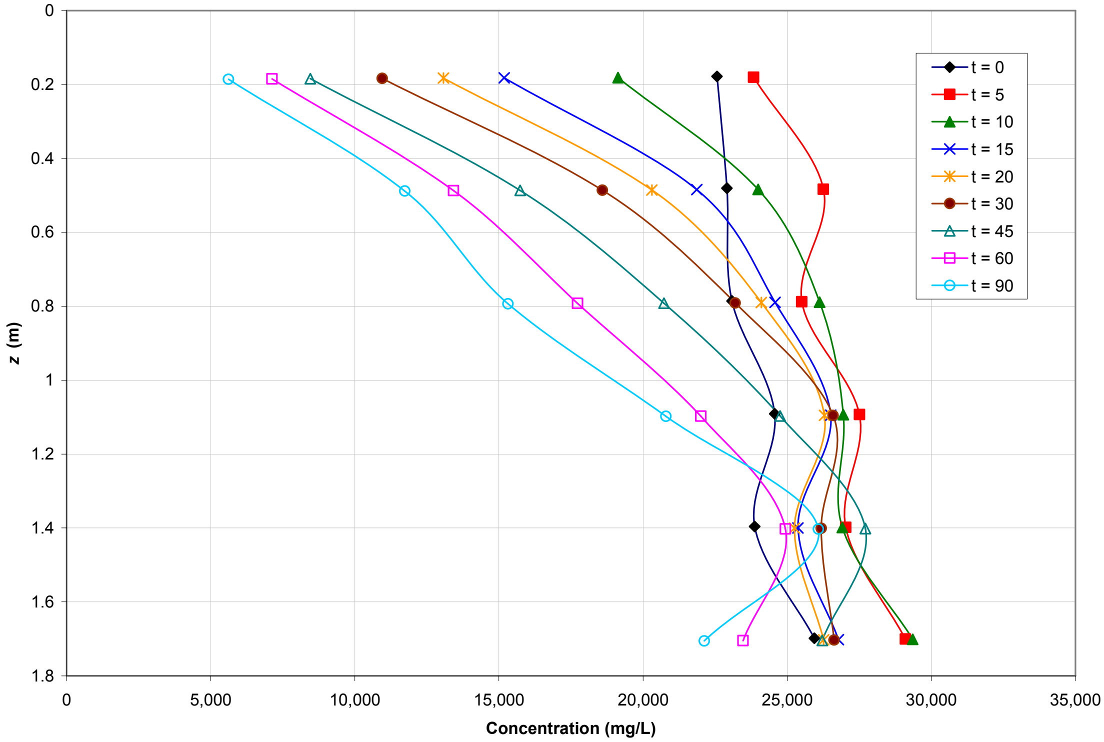

Figure 5 - 11: Small Micro-Spheres – High Concentration – Replicate 3.

APPENDIX L: Column Test Graphs – Large Glass Micro-Spheres

Replicate 1 for the low concentration revealed some errors and thus was not used in the analysis.

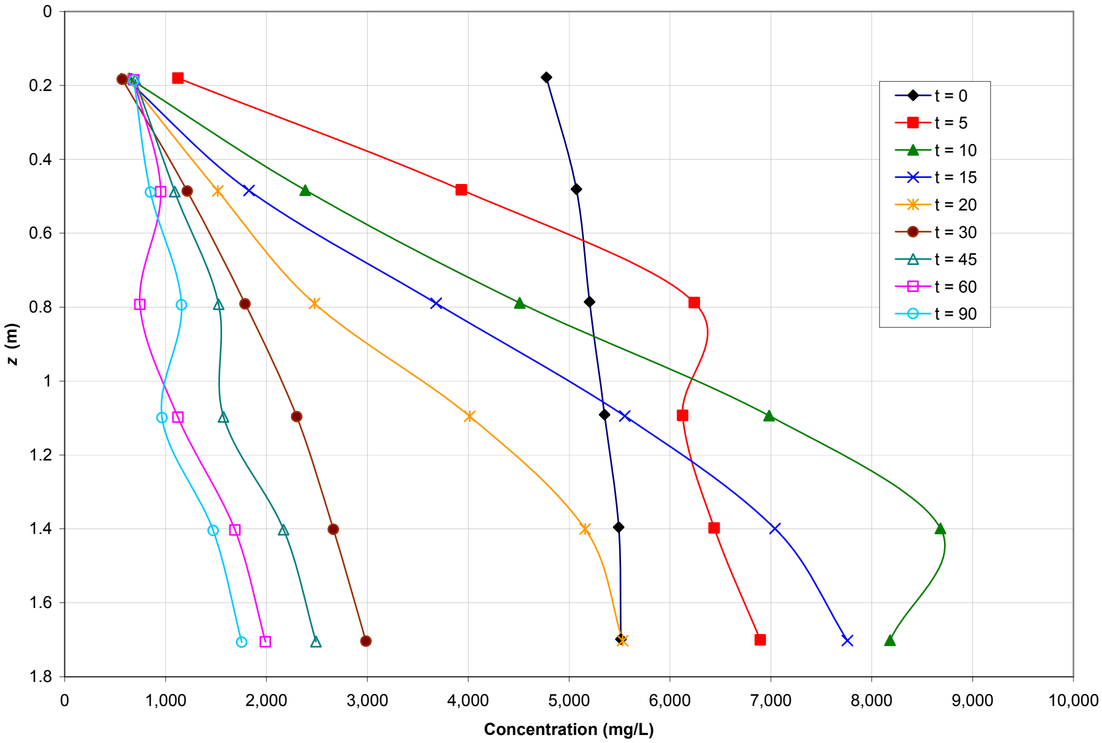

Figure 5 - 12: Large Micro-Spheres – Low Concentration – Replicate 2.

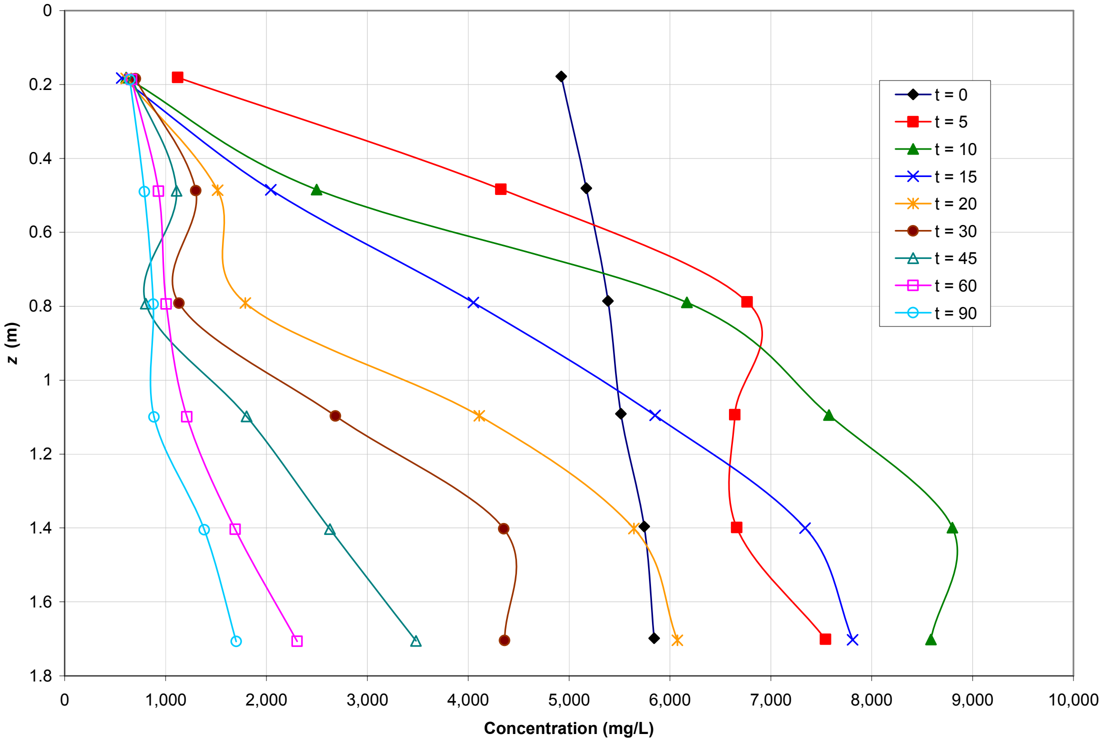

Figure 5 - 13: Large Micro-Spheres – Low Concentration – Replicate 3.

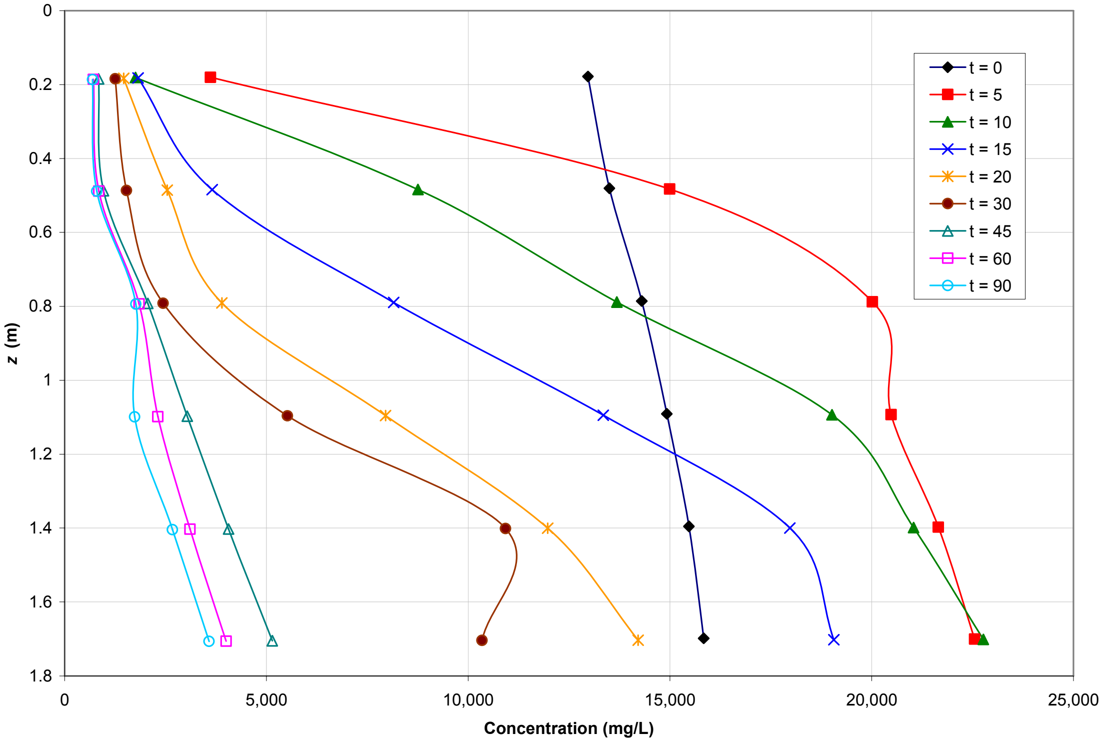

Figure 5 - 14: Large Micro-Spheres – Medium Concentration – Replicate 1.

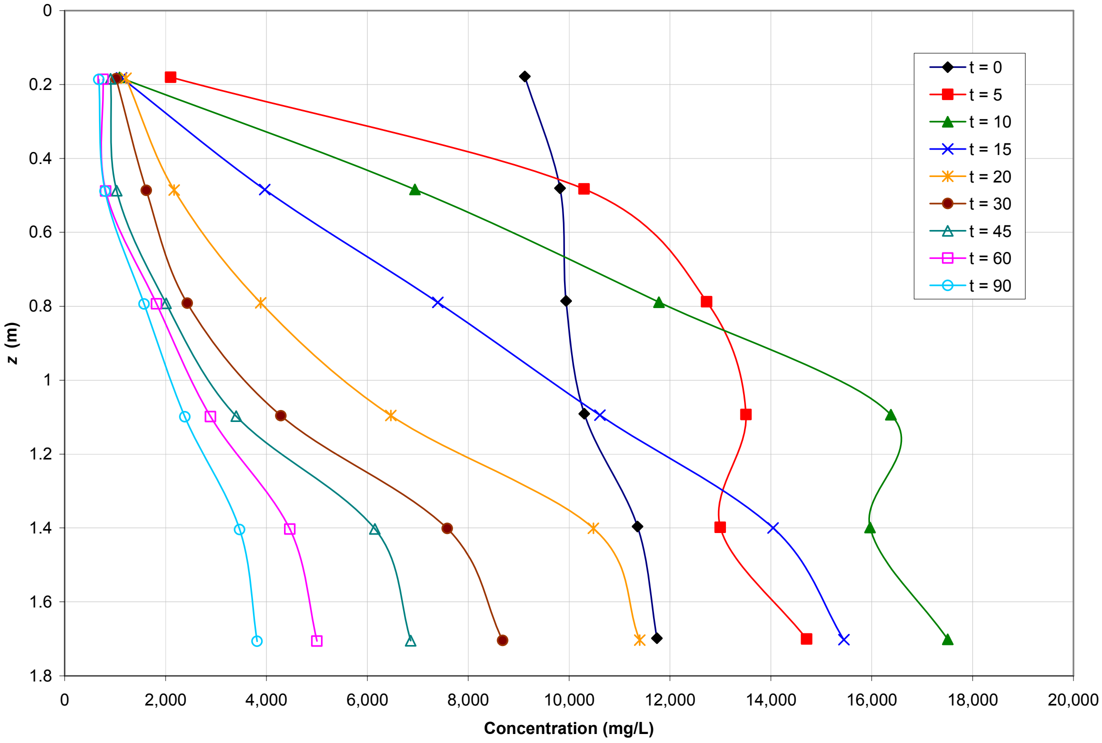

Figure 5 - 15: Large Micro-Spheres – Medium Concentration – Replicate 2.

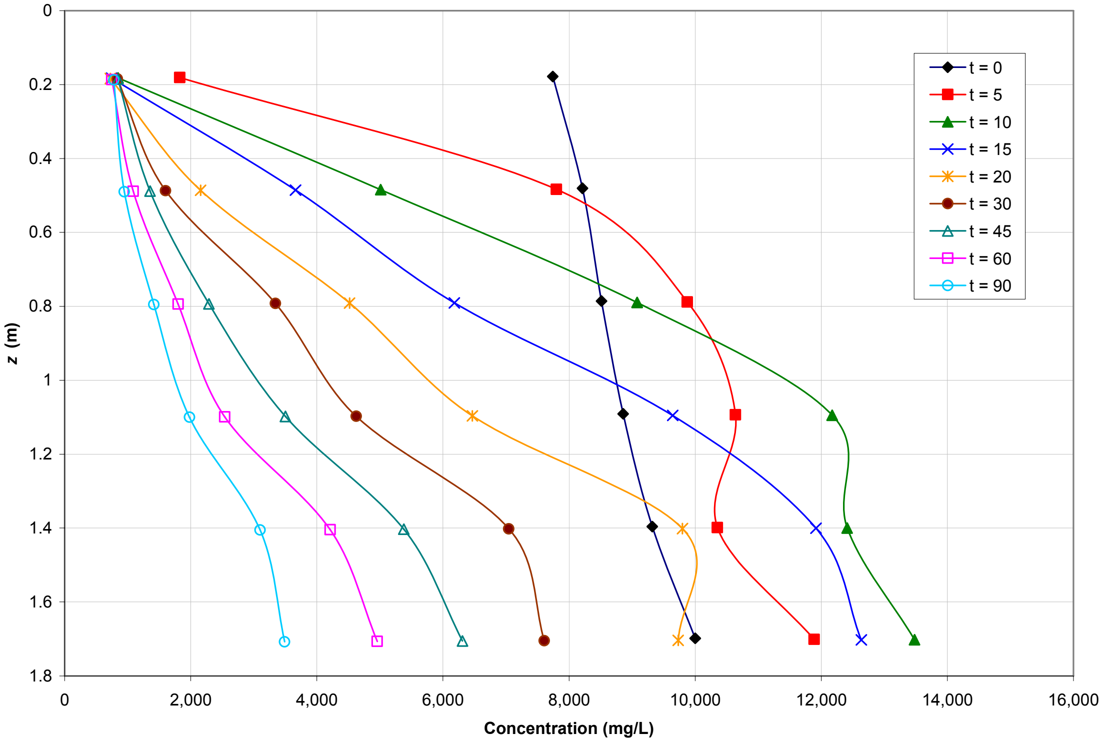

Figure 5 - 16: Large Micro-Spheres – Medium Concentration – Replicate 3.

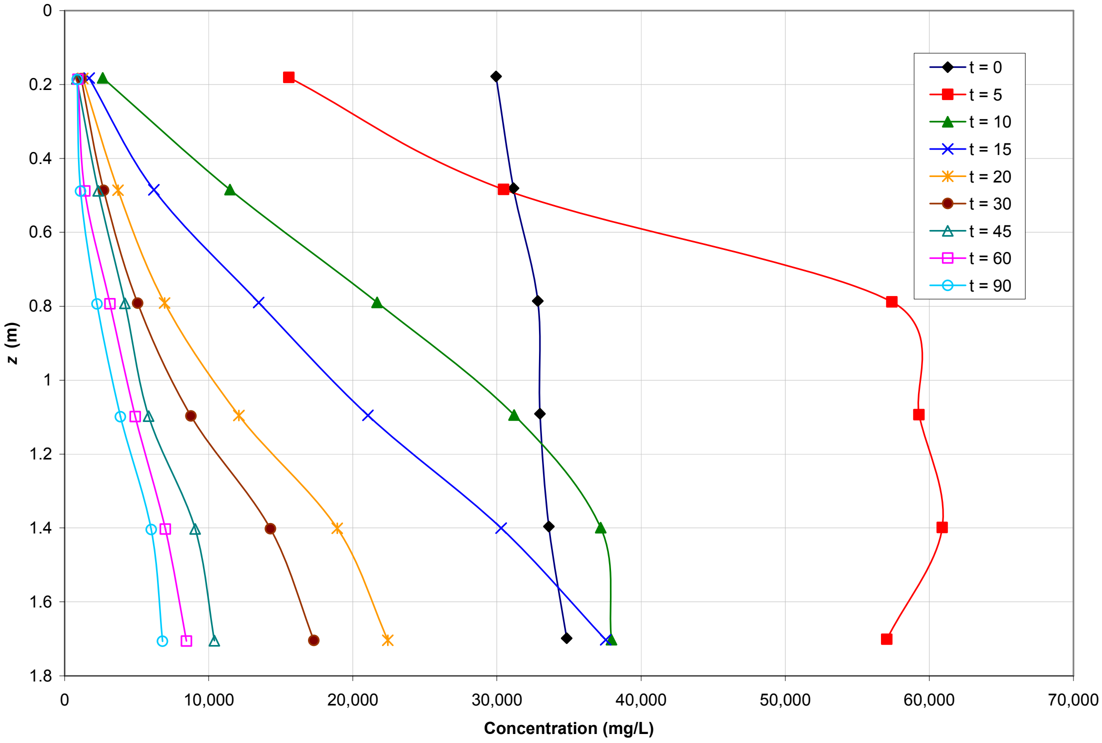

Figure 5 - 17: Large Micro-Spheres – High Concentration – Replicate 1.

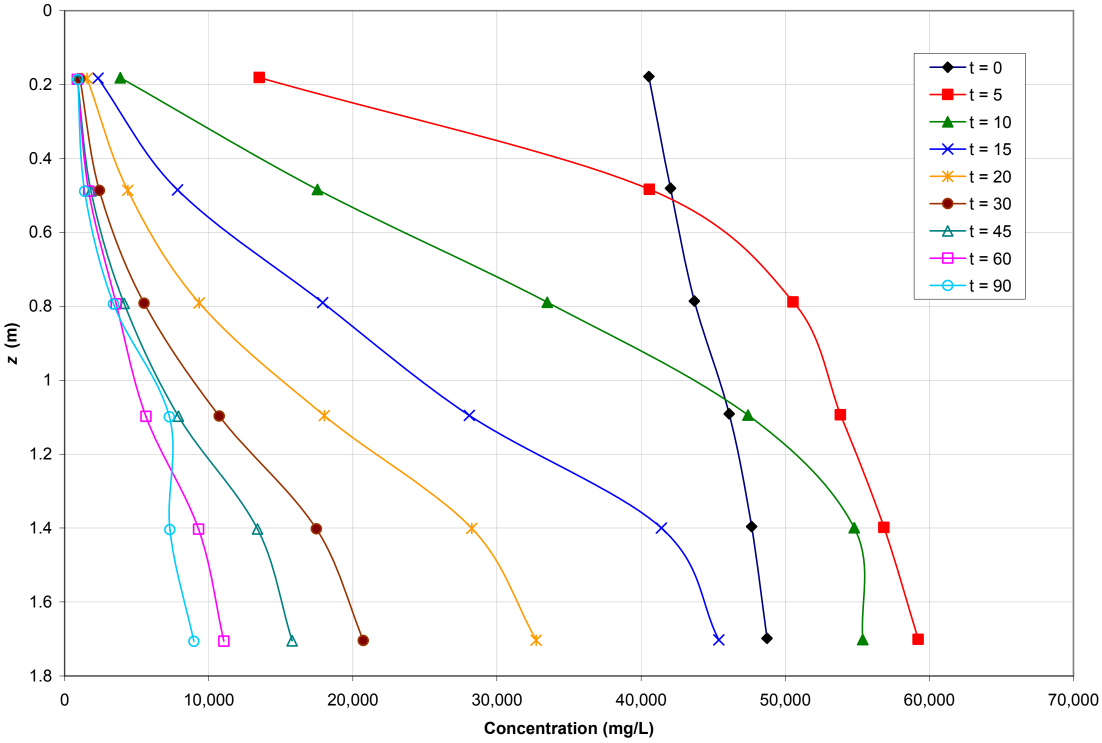

Figure 5 - 18: Large Micro-Spheres – High Concentration – Replicate 2.

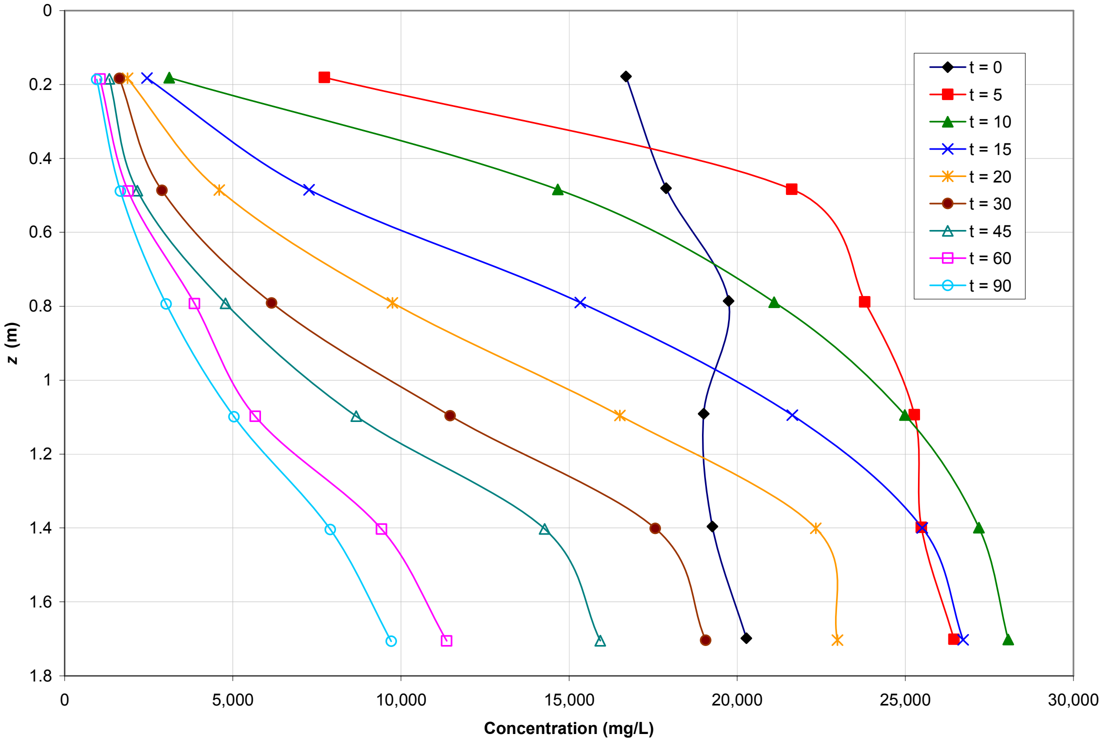

Figure 5 - 19: Large Micro-Spheres – High Concentration – Replicate 3.

APPENDIX M: C# Code of the Non-Staggered NT Numerical Method

```
class NTscheme
    {
        ///<summary>
        ///Class Members/Attributes
        ///<summary>
        int nmodes;
        ColumnVector GammaVec;
        ColumnVector d;
        double binf;
        double h, dz;
        int numT, numJ;
        double lambda;
        double theta;
        double phimax;
        double M1;
        double M2;
        ColumnVector tcoeff;
        ///<summary>
        ///Static method that returs M1
        ///<summary>
        private static double M_1(int n, ColumnVector Pvec)
        {
            double average = 0;

            for (int i = 0; i < n; i++)
            {
                average += Pvec[i] * Pvec[i + n] * Pvec[i + 2 * n];
            }

            return average;
        }
        ///<summary>
        ///Static method that returs M2
        ///<summary>
        private static double M_2(int n, ColumnVector Pvec)
        {
            double variance = 0;

            for (int i = 0; i < n; i++)
            {
                variance += Pvec[i] * Pvec[i + n] * (Pvec[i + n] + 1) *
Math.Pow(Pvec[i + 2 * n], 2);
            }

            return variance;
        }
        ///<summary>
        ///Constructor
        ///<summary>
        public NTscheme(int num_modes, ColumnVector Coeffs,
ColumnVector Time_Coeffs, ColumnVector Initial_Gamma_Parameters, double
Stokes_Coefficient, int numtsteps, int numjsteps, double deltat, double
deltaz, double PhiMax)
        {
            nmodes = num_modes;
            GammaVec = Initial_Gamma_Parameters.Clone();
            binf = Stokes_Coefficient;
            h = deltat;
            dz = deltaz;
            numT = numtsteps;
            numJ = numjsteps;
            lambda = deltat / deltaz;
            theta = 2;
            phimax = PhiMax;
            M1 = M_1(2, GammaVec);
            M2 = M_2(2, GammaVec);
            d = Coeffs.Clone();
            tcoeff = Time_Coeffs.Clone();
            theta = 2;
        }
        ///<summary>
        ///Method that returns true if all the elements of a vector are
positive or
        ///false otherwise
        ///<summary>
        private bool IsAllPositive(ColumnVector Pvec)
        {
            bool iap = false;
            int d = Pvec.Dimension;

            for (int i = 0; i < d; i++)
            {
                if (Pvec[i] <= 0)
                {
                    iap = false;
                    break;
                }
                else iap = true;
            }

            return iap;
        }
        ///<summary>
        ///Method that returns true if all the elements of a vector are
negative or
        ///false otherwise
        ///<summary>
        private bool IsAllNegative(ColumnVector Pvec)
        {
            bool ian = false;
            int d = Pvec.Dimension;

            for (int i = 0; i < d; i++)
            {
                if (Pvec[i] >= 0)
                {
                    ian = false;
                    break;
                }
                else ian = true;
            }

            return ian;
        }
        ///<summary>
        ///Method that returns the numerical derivative using the
        ///MinMod delimiter
        ///<summary>
        private double MM(ColumnVector theVec)
        {
            double mm = 0;
            if (IsAllPositive(theVec))
                mm = theVec.Min();
            else if (IsAllNegative(theVec))
                mm = theVec.Max();

            return mm;
        }
        ///<summary>
        ///Flux Function
        ///<summary>
        public double flux(double phi)
        {
            double theres = phi * binf * (M2 + phi * (d[0] *
Math.Pow(M1, 2) + d[1] * M2));

            return theres;
        }
        public double flux(int j, int time, double phi)
        {
            double M1max = M1 * (2 - Math.Exp(-tcoeff[0] * time * h));
            double M1min = 2 * M1 - M1max;
            M1max = M1;
            double M2max = M2 * (2 - Math.Exp(-tcoeff[1] * time * h));
            double M2min = 2 * M2 - M2max;
            M2max = M2;
            double m1 = (M1max - M1min) / (1.75E6);
            double m2 = (M2max - M2min) / (1.75E6);

            double M1zt = m1 * j * dz + M1min;
            double M2zt = m2 * j * dz + M2min;

            double theres = phi * binf * (M2zt + phi * (d[0] *
Math.Pow(M1zt, 2) + d[1] * M2zt));

            return theres;
        }
        ///<summary>
        ///Derivative of the flux
        ///<summary>
        public double flux_prime(double phi)
        {
            double theres = binf * (M2 + 2 * phi * (d[0] * Math.Pow(M1,
2) + d[1] * M2));

            return theres;
        }
        public double flux_prime(int j, int time, double phi)
        {
            double M1max = 2 * M1 * (1 - Math.Exp(-tcoeff[0] * time *
h));
```

```
            double M1min = 2 * M1 - M1max;
            M1max = M1;
            double M2max = 2 * M2 * (1 - Math.Exp(-tcoeff[1] * time *
h));
            double M2min = 2 * M2 - M2max;
            M2max = M2;
            double m1 = (M1max - M1min) / (1.75E6);
            double m2 = (M2max - M2min) / (1.75E6);

            double M1zt = m1 * j * dz + M1min;
            double M2zt = m2 * j * dz + M2min;

            double theres = binf * (M2zt + 2 * phi * (d[0] *
Math.Pow(M1zt, 2) + d[1] * M2zt));

            return theres;
        }
        ///<summary>
        ///Discrete derivative function
        ///<summary>
        public double phi_prime_j(int j, ColumnVector Phi)
        {
            ColumnVector Elements = new ColumnVector(3);

            double deltajm = Phi[j] - Phi[j - 1];
            double deltajp = Phi[j + 1] - Phi[j];
            Elements[0] = theta * deltajp;
            Elements[1] = 0.5 * (deltajm + deltajp);
            Elements[2] = theta * deltajm;

            return MM(Elements);
        }
        ///<summary>
        ///Predictor form
        ///<summary>
        public double phi_j_nphalf(int j, ColumnVector Phi)
        {
            double theres = 0;

            if ((j < 3) || (j > numJ + 3))
                theres = Phi[j];
            else
                theres = Phi[j] - 0.5 * lambda * flux_prime(Phi[j]) *
phi_prime_j(j, Phi);

            return theres;
        }
        public double phi_j_nphalf(int j, int time, ColumnVector Phi)
        {
            double theres = 0;

            if ((j < 3) || (j > numJ + 3))
                theres = Phi[j];
            else
                theres = Phi[j] - 0.5 * lambda * flux_prime(j, time,
Phi[j]) * phi_prime_j(j, Phi);
```

```
            return theres;
        }
        ///<summary>
        ///Corrector form
        ///<summary>
        public double phi_jphalf_np1(int j, ColumnVector Phi)
        {
            double theres;
            double p1 = 0.5 * (Phi[j] + Phi[j + 1]);
            double p2 = 0.125 * (phi_prime_j(j, Phi) - phi_prime_j(j +
1, Phi));
            double p3 = 0;
            if (j == 2)
                p3 = lambda * flux(phi_j_nphalf(j + 1, Phi));
            else if (j == numJ + 3)
                p3 = lambda * (-flux(phi_j_nphalf(j, Phi)));
            else if ((j > 2) || (j < numJ + 4))
                p3 = lambda * (flux(phi_j_nphalf(j + 1, Phi)) -
flux(phi_j_nphalf(j, Phi)));

            theres = p1 + p2 - p3;

            return theres;
        }
        public double phi_jphalf_np1(int j, int time, double tcoeff1,
double tcoeff2, ColumnVector Phi)
        {
            double theres;
            double p1 = 0.5 * (Phi[j] + Phi[j + 1]);
            double p2 = 0.125 * (phi_prime_j(j, Phi) - phi_prime_j(j +
1, Phi));
            double p3 = 0;
            if (j == 2)
                p3 = lambda * flux(j, time, phi_j_nphalf(j + 1, Phi));
            else if (j == numJ + 3)
                p3 = lambda * (-flux(j, time, phi_j_nphalf(j, Phi)));
            else if ((j > 2) || (j < numJ + 4))
                p3 = lambda * (flux(j, time, phi_j_nphalf(j + 1, Phi))
- flux(j, time, phi_j_nphalf(j, Phi)));

            theres = p1 + p2 - p3;

            return theres;
        }
        ///<summary>
        ///Predictor/Corrector form
        ///<summary>
        public double delta_phi_i_np1(int i, ColumnVector Phi)
        {
            double theres;
            double p1 = 0.5 * (Phi[i + 1] - Phi[i - 1]);
            double p2 = 0.125 * (phi_prime_j(i - 1, Phi) - 2 *
phi_prime_j(i, Phi) + phi_prime_j(i + 1, Phi));
            double p3 = 0;
            if (i == 2)
                p3 = p3 = lambda * (flux(phi_j_nphalf(i + 1, Phi)));
            else if (i == 3)
```

```
                p3 = lambda * (flux(phi_j_nphalf(i + 1, Phi)) - 2 *
flux(phi_j_nphalf(i, Phi)));
            else if (i == numJ + 3)
                p3 = lambda * (flux(phi_j_nphalf(i - 1, Phi)) - 2 *
flux(phi_j_nphalf(i, Phi)));
            else if (i == numJ + 4)
                p3 = lambda * (flux(phi_j_nphalf(i - 1, Phi)));
            else if ((i > 3) && (i < numJ + 3))
                p3 = lambda * (flux(phi_j_nphalf(i - 1, Phi)) - 2 *
flux(phi_j_nphalf(i, Phi)) + flux(phi_j_nphalf(i + 1, Phi)));
            theres = p1 - p2 - p3;

            return theres;
        }
        public double delta_phi_i_np1(int i, int time, ColumnVector
Phi)
        {
            double theres;
            double p1 = 0.5 * (Phi[i + 1] - Phi[i - 1]);
            double p2 = 0.125 * (phi_prime_j(i - 1, Phi) - 2 *
phi_prime_j(i, Phi) + phi_prime_j(i + 1, Phi));
            double p3 = 0;
            if (i == 2)
                p3 = p3 = lambda * (flux(i, time, phi_j_nphalf(i + 1,
Phi)));
            else if (i == 3)
                p3 = lambda * (flux(i, time, phi_j_nphalf(i + 1, Phi))
- 2 * flux(i, time, phi_j_nphalf(i, Phi)));
            else if (i == numJ + 3)
                p3 = lambda * (flux(i, time, phi_j_nphalf(i - 1, Phi))
- 2 * flux(i, time, phi_j_nphalf(i, Phi)));
            else if (i == numJ + 4)
                p3 = lambda * (flux(i, time, phi_j_nphalf(i - 1,
Phi)));
            else if ((i > 3) && (i < numJ + 3))
                p3 = lambda * (flux(i, time, phi_j_nphalf(i - 1, Phi))
- 2 * flux(i, time, phi_j_nphalf(i, Phi)) + flux(i, time,
phi_j_nphalf(i + 1, Phi)));
            theres = p1 - p2 - p3;

            return theres;
        }
        ///<summary>
        ///Staggered discrete derivatives
        ///<summary>
        public double phi_prime_jmhalf(int j, ColumnVector Phi)
        {
            ColumnVector Deltaj = new ColumnVector(2);
            Deltaj[0] = delta_phi_i_np1(j, Phi);
            Deltaj[1] = delta_phi_i_np1(j - 1, Phi);
            double theres = MM(Deltaj);

            return theres;
        }
        public double phi_prime_jphalf(int j, ColumnVector Phi)
        {
            ColumnVector Deltajp1 = new ColumnVector(2);
            Deltajp1[0] = delta_phi_i_np1(j + 1, Phi);
            Deltajp1[1] = delta_phi_i_np1(j, Phi);
            double theres = MM(Deltajp1);

            return theres;
        }
        ///<summary>
        ///Non-Staggered corrector scheme
        ///<summary>
        public double phi_j_np1(int j, ColumnVector Phi)
        {
            double p1 = 0.25 * (Phi[j - 1] + 2 * Phi[j] + Phi[j + 1]);
            double p2 = 0.0625 * (phi_prime_j(j + 1, Phi) - 
phi_prime_j(j - 1, Phi));
            double p3 = 0;
            if ((j == 2) || (j == 3))
                p3 = 0.5 * lambda * (flux(phi_j_nphalf(j + 1, Phi)));
            else if ((j == numJ + 3) || (j == numJ + 4))
                p3 = 0.5 * lambda * (-flux(phi_j_nphalf(j - 1, Phi)));
            else if ((j > 3) && (j < numJ + 3))
                p3 = 0.5 * lambda * (flux(phi_j_nphalf(j + 1, Phi)) - 
flux(phi_j_nphalf(j - 1, Phi)));
            double p4 = 0.125 * (phi_prime_jphalf(j, Phi) - 
phi_prime_jmhalf(j, Phi));
            double theres = p1 - p2 - p3 - p4;

            if (theres < 0)
                theres = 0;
            else if (theres > phimax)
                theres = phimax;

            return theres;
        }
        public double phi_j_np1(int j, int time, ColumnVector Phi)
        {
            double p1 = 0.25 * (Phi[j - 1] + 2 * Phi[j] + Phi[j + 1]);
            double p2 = 0.0625 * (phi_prime_j(j + 1, Phi) - 
phi_prime_j(j - 1, Phi));
            double p3 = 0;
            if ((j == 2) || (j == 3))
                p3 = 0.5 * lambda * (flux(j, time, phi_j_nphalf(j + 1, 
Phi)));
            else if ((j == numJ + 3) || (j == numJ + 4))
                p3 = 0.5 * lambda * (-flux(j, time, phi_j_nphalf(j - 1, 
Phi)));
            else if ((j > 3) && (j < numJ + 3))
                p3 = 0.5 * lambda * (flux(j, time, phi_j_nphalf(j + 1, 
Phi)) - flux(j, time, phi_j_nphalf(j - 1, Phi)));
            double p4 = 0.125 * (phi_prime_jphalf(j, Phi) - 
phi_prime_jmhalf(j, Phi));
            double theres = p1 - p2 - p3 - p4;

            if (theres < 0)
                theres = 0;
            else if (theres > phimax)
                theres = phimax;
            return theres;
        }
        ///<summary>
        ///Time-Evolution
        ///<summary>
        public List<ColumnVector> TimeEvolution(ColumnVector iniPhi)
        {
            List<ColumnVector> Results = new List<ColumnVector>();
            ColumnVector Phi_n = iniPhi.Clone();
            Results.Add(Phi_n);

            for (int n = 1; n <= numT; n++)
            {
                ColumnVector Phi_np1 = new ColumnVector(numJ + 7);
                int maxj = numJ + 3;
                for (int j = 3; j <= maxj; j++)
                {
                    Phi_np1[j] = phi_j_np1(j, Phi_n);
                }

                //Piece-wise linear interpolation for the inflow
section
                double s = (Phi_np1[4] - Phi_np1[3]) * Math.Pow(dz, -
1);
                double nlin = Phi_np1[3];
                double x1 = -1 * dz;
                double x2 = -2 * dz;
                double x3 = -3 * dz;
                Phi_np1[0] = s * x1 + nlin;
                Phi_np1[1] = s * x2 + nlin;
                Phi_np1[2] = s * x3 + nlin;

                //Piece-wise linear interpolation for the outflow
section
                s = (Phi_np1[numJ + 3] - Phi_np1[numJ + 2]) *
Math.Pow(dz, -1);
                nlin = Phi_np1[numJ + 3] - s * numJ * dz; ;
                x1 = (numJ + 1) * dz;
                x2 = (numJ + 2) * dz;
                x3 = (numJ + 3) * dz;
                Phi_np1[numJ + 4] = s * x1 + nlin;
                Phi_np1[numJ + 5] = s * x2 + nlin;
                Phi_np1[numJ + 6] = s * x3 + nlin;

                Results.Add(Phi_np1);
                Phi_n = Phi_np1.Clone();
            }

            return Results;
        }
        public List<ColumnVector> TimeEvolution_zt(ColumnVector iniPhi)
        {
            List<ColumnVector> Results = new List<ColumnVector>();
            ColumnVector Phi_n = iniPhi.Clone();
            Results.Add(Phi_n);

            for (int n = 1; n <= numT; n++)
            {
                ColumnVector Phi_np1 = new ColumnVector(numJ + 7);
                int maxj = numJ + 3;
                for (int j = 3; j <= maxj; j++)
                {
                    Phi_np1[j] = phi_j_np1(j, n, Phi_n);
                }

                //Piece-wise linear interpolation for the inflow
section
                double s = (Phi_np1[4] - Phi_np1[3]) * Math.Pow(dz, -
1);
                double nlin = Phi_np1[3];
                double x1 = -1 * dz;
                double x2 = -2 * dz;
                double x3 = -3 * dz;
                Phi_np1[0] = s * x1 + nlin;
                Phi_np1[1] = s * x2 + nlin;
                Phi_np1[2] = s * x3 + nlin;

                //Piece-wise linear interpolation for the outflow
section
                s = (Phi_np1[numJ + 3] - Phi_np1[numJ + 2]) *
Math.Pow(dz, -1);
                nlin = Phi_np1[numJ + 3] - s * numJ * dz; ;
                x1 = (numJ + 1) * dz;
                x2 = (numJ + 2) * dz;
                x3 = (numJ + 3) * dz;
                Phi_np1[numJ + 4] = s * x1 + nlin;
                Phi_np1[numJ + 5] = s * x2 + nlin;
                Phi_np1[numJ + 6] = s * x3 + nlin;

                Results.Add(Phi_np1);
                Phi_n = Phi_np1.Clone();
            }

            return Results;
        }
    }
```

APPENDIX N: C# Code of the Cubic Spline Method

```
public static double CubicSpline(double Height, ColumnVector x,
ColumnVector y)
        {
            ColumnVector xnew = new ColumnVector(x.Dimension);
            for (int i = 0; i < x.Dimension; i++)
                xnew[i] = x[i];
            int N = xnew.Dimension;
            xnew[N - 1] = 10 * x[N - 2];

            ColumnVector h = new ColumnVector(N - 1);
            for (int k = 0; k < N - 1; k++)
                h[k] = xnew[k + 1] - xnew[k];

            ColumnVector d = new ColumnVector(N - 1);
            for (int k = 0; k < N - 1; k++)
                d[k] = (y[k + 1] - y[k]) / h[k];

            ColumnVector u = new ColumnVector(N - 2);
            for (int k = 1; k < N - 1; k++)
                u[k - 1] = 6 * (d[k] - d[k - 1]);

            ColumnVector m = new ColumnVector(N);
            ColumnVector maux = new ColumnVector(N - 2);
            SquareMatrix A = new SquareMatrix(N - 2);
            A[0, 0] = 3 * h[0] + 2 * h[1] + Math.Pow(h[0], 2) / h[1];
            A[0, 1] = h[1] - Math.Pow(h[0], 2) / h[1];
            for (int k = 1; k < N - 3; k++)
            {
                A[k, k - 1] = h[k - 1];
                A[k, k] = 2 * (h[k - 1] + h[k]);
                A[k, k + 1] = h[k];
            }
            A[N - 3, N - 4] = h[N - 3] - Math.Pow(h[N - 2], 2) / h[N -
3];
            A[N - 3, N - 3] = 2 * h[N - 3] + 3 * h[N - 2] +
Math.Pow(h[N - 2], 2) / h[N - 3];

            maux = A.Inverse() * u;
            m[0] = maux[0] - h[0] * (maux[1] - maux[0]) / h[1];
            m[N - 1] = maux[N - 3] + h[N - 2] * (maux[N - 3] - maux[N -
4]) / h[N - 3];
            for (int k = 1; k < N - 1; k++)
            {
                m[k] = maux[k - 1];
            }

            Matrix S = new Matrix(N - 1, 4);
            for (int k = 0; k < N - 1; k++)
            {
                S[k, 0] = y[k];
                S[k, 1] = d[k] - h[k] * (2 * m[k] + m[k + 1]) / 6;
                S[k, 2] = m[k] / 2;
                S[k, 3] = (m[k + 1] - m[k]) / (6 * h[k]);
            }

            double res = 0;
            double Sx;
            int Ik;

            if (Height < xnew[0])
            {
                double w = Height - xnew[0];
                res = ((S[0, 3] * w + S[0, 2]) * w + S[0, 1]) * w +
S[0, 0];
            }
            else if (Height > xnew[N - 1])
            {
                double w = Height - xnew[N - 1];
                res = ((S[N - 2, 3] * w + S[N - 2, 2]) * w + S[N - 2,
1]) * w + S[N - 2, 0];
            }
            else
            {
                for (int k = 0; k < N - 1; k++)
                {
                    if ((Height < xnew[k]) || (Height > xnew[k + 1]))
                        Ik = 0;
                    else Ik = 1;
                    double w = Height - xnew[k];
                    Sx = ((S[k, 3] * w + S[k, 2]) * w + S[k, 1]) * w +
S[k, 0];
                    res += Sx * Ik;
                }
            }
            return res;
        }
```

APPENDIX O: Observed and Predicted Concentration Profiles of the Small Glass Micro-Spheres after 60 Minutes of Settling

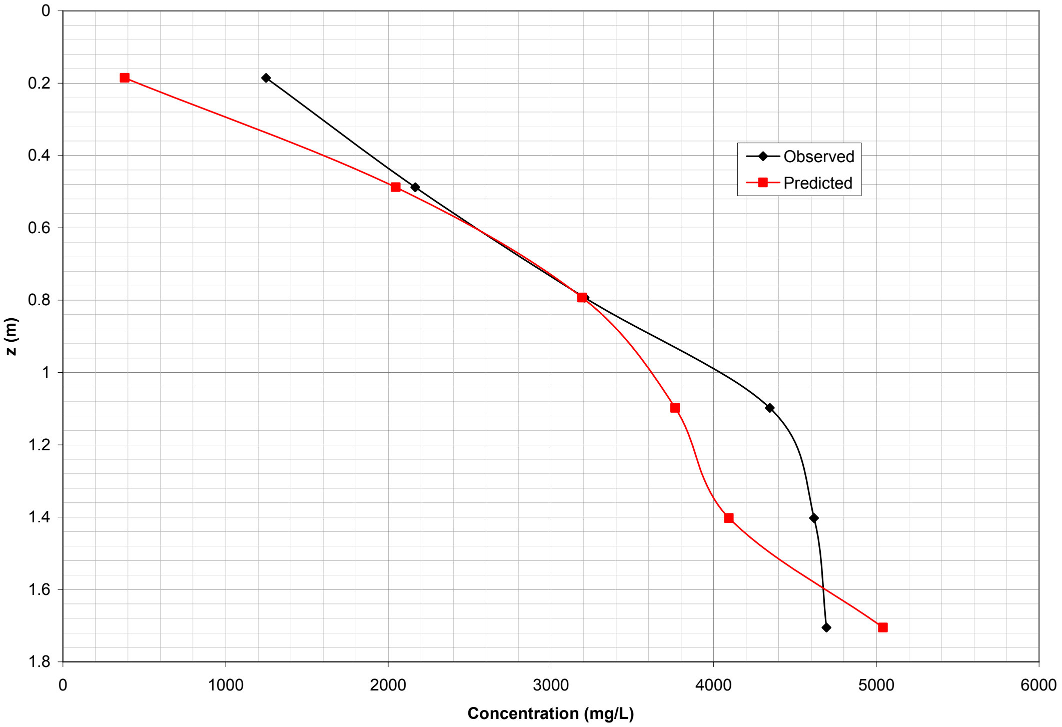

Figure 5 - 20: Observed vs. predicted concentration profiles after 60 minutes of settling for low initial concentration and replicate 1.

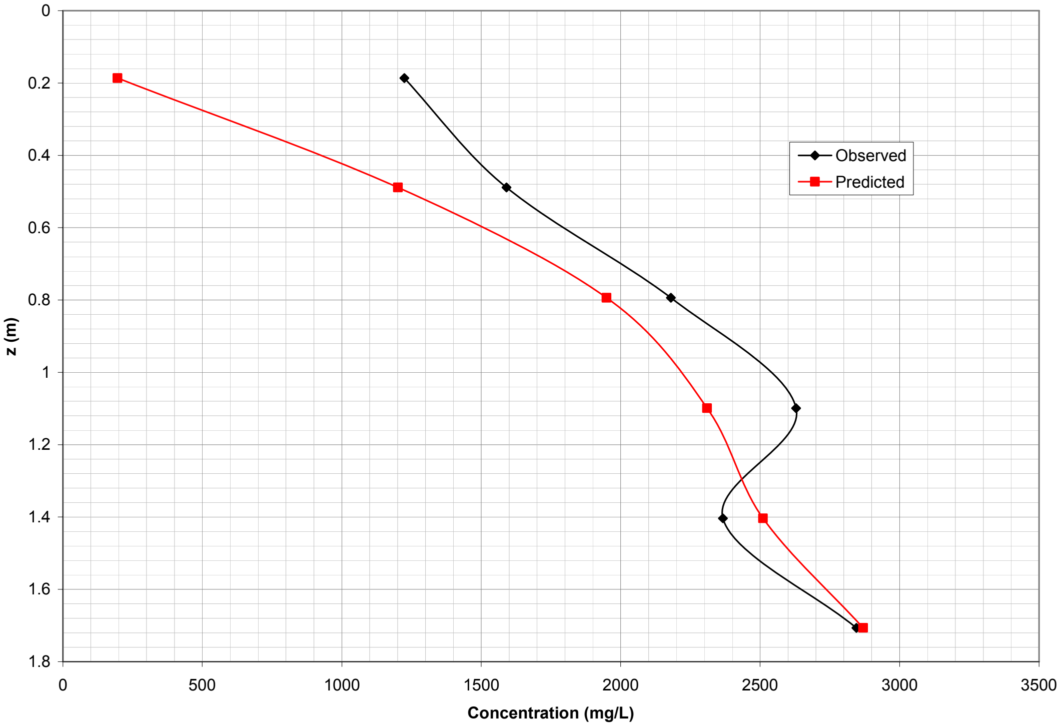

Figure 5 - 21: Observed vs. predicted concentration profiles after 60 minutes of settling for low initial concentration and replicate 2.

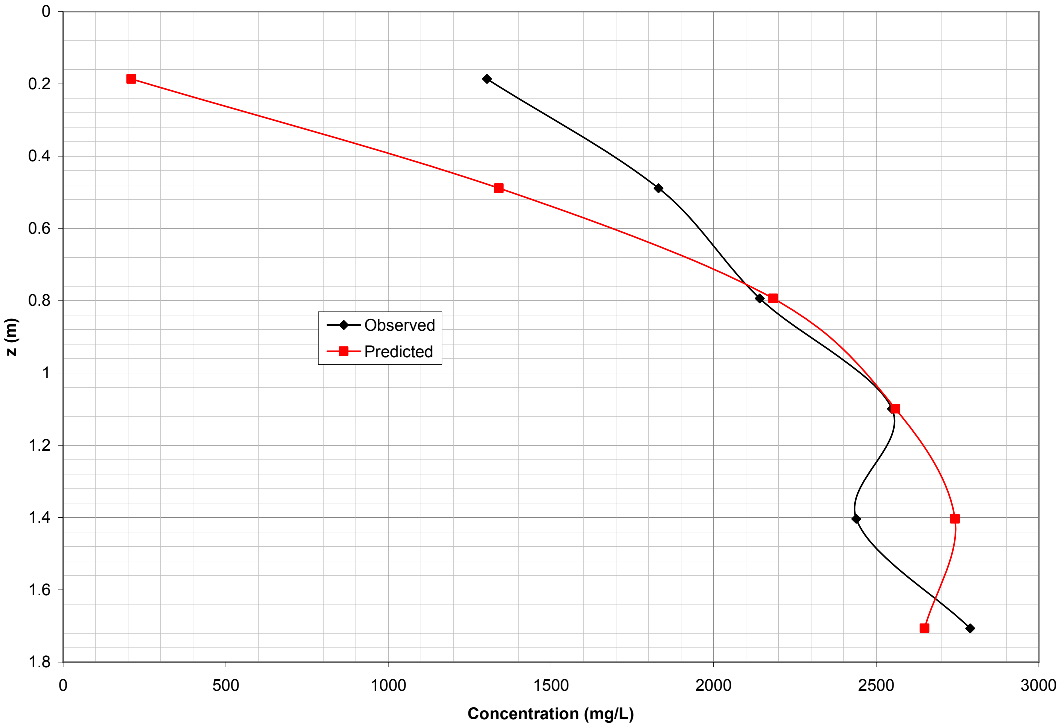

Figure 5 - 22: Observed vs. predicted concentration profiles after 60 minutes of settling for low initial concentration and replicate 3.

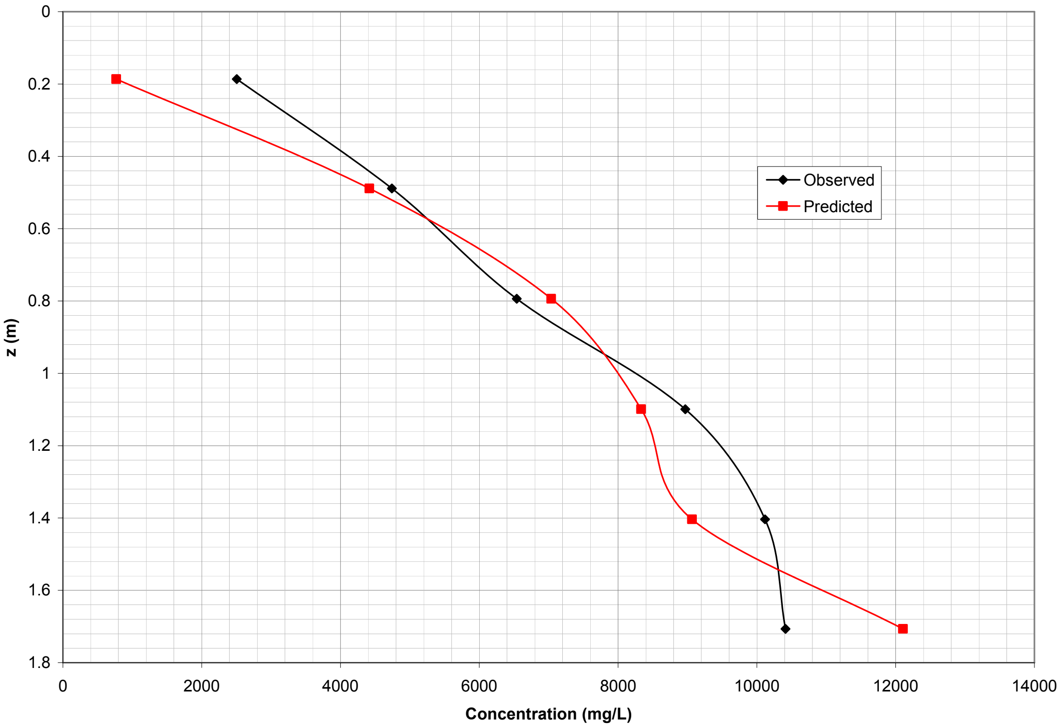

Figure 5 - 23: Observed vs. predicted concentration profiles after 60 minutes of settling for medium initial concentration and replicate 1.

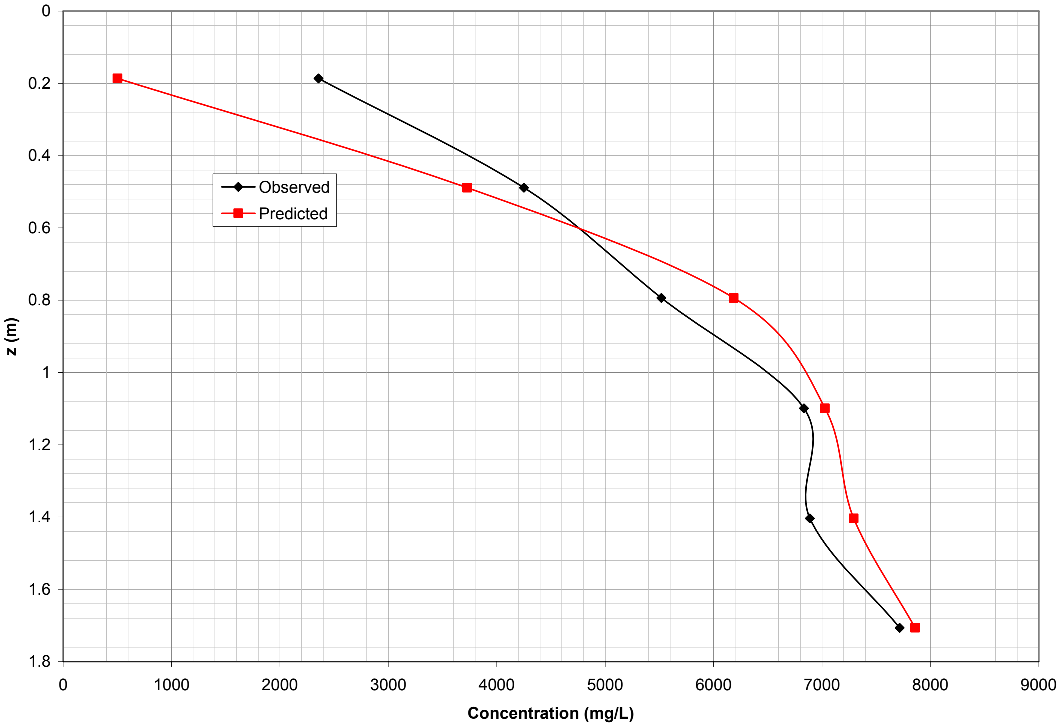

Figure 5 - 24: Observed vs. predicted concentration profiles after 60 minutes of settling for medium initial concentration and replicate 2.

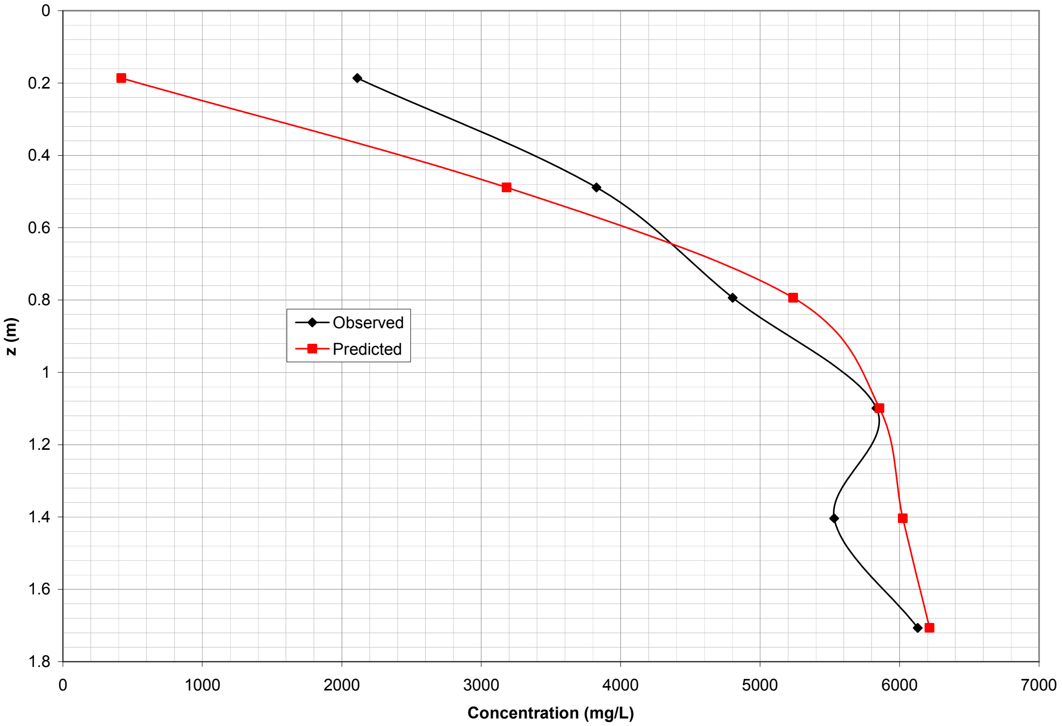

Figure 5 - 25: Observed vs. predicted concentration profiles after 60 minutes of settling for medium initial concentration and replicate 3.

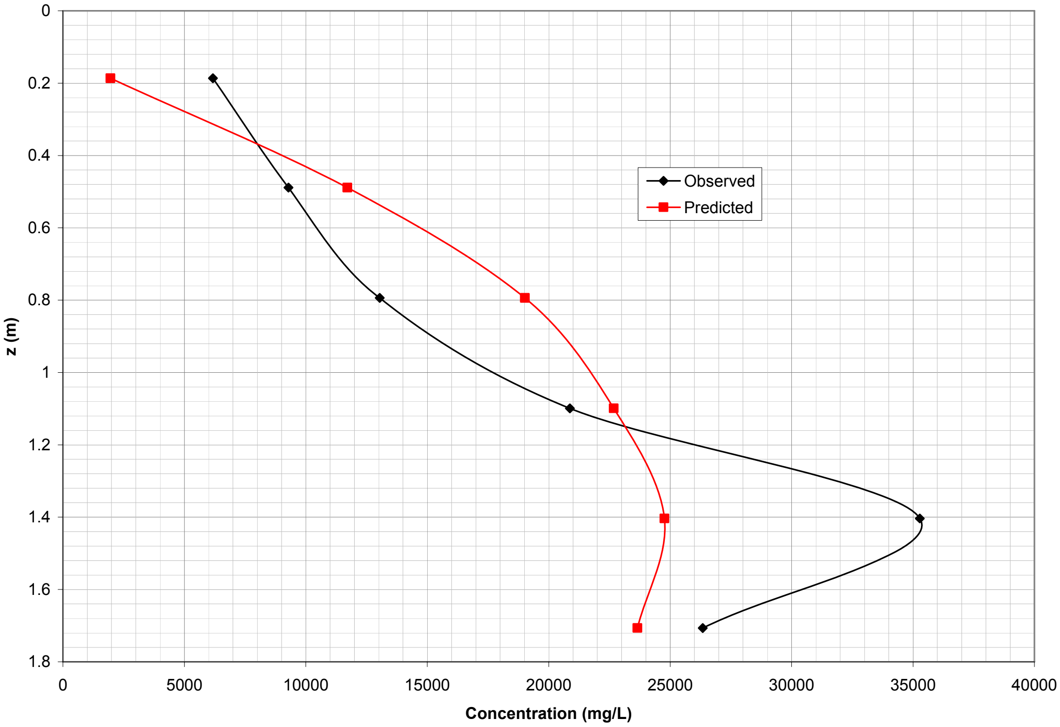

Figure 5 - 26: Observed vs. predicted concentration profiles after 60 minutes of settling for high initial concentration and replicate 1.

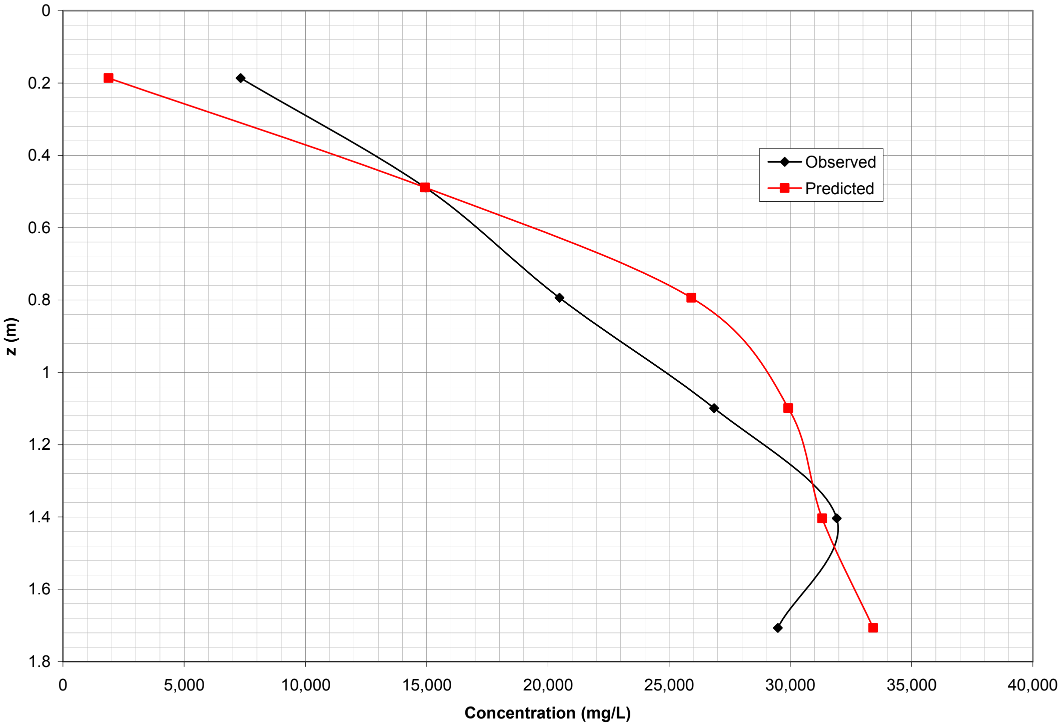

Figure 5 - 27: Observed vs. predicted concentration profiles after 60 minutes of settling for high initial concentration and replicate 2.

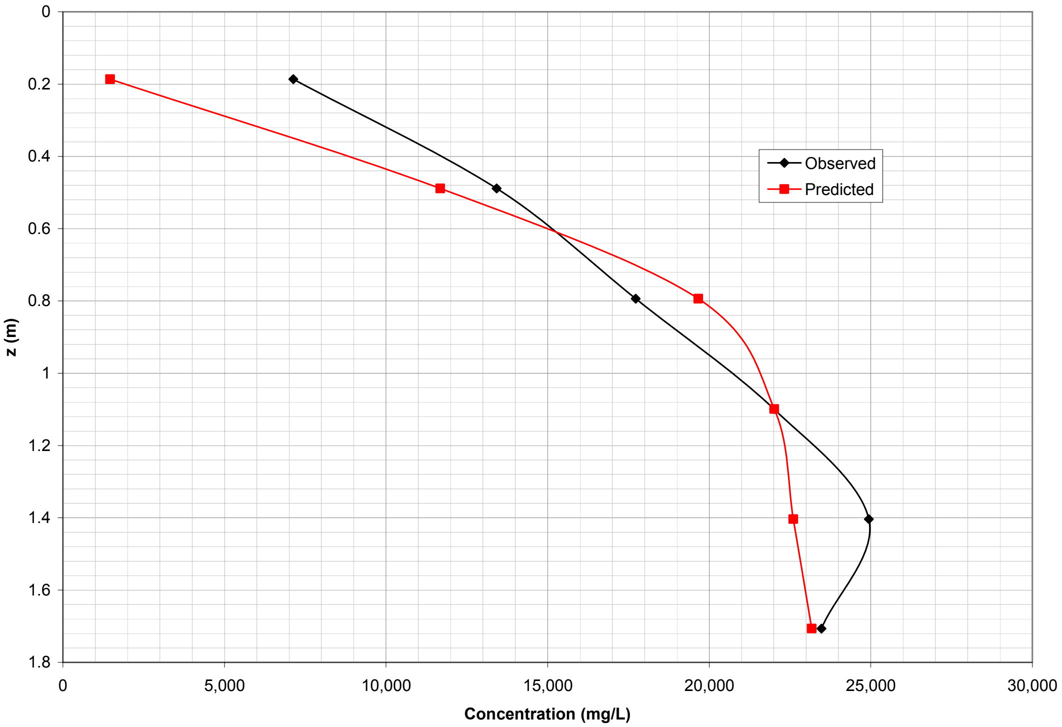

Figure 5 - 28: Observed vs. predicted concentration profiles after 60 minutes of settling for high initial concentration and replicate 3.

APPENDIX P: Observed and Predicted Concentration Profiles of the Large Glass Micro-Spheres after 60 Minutes of Settling

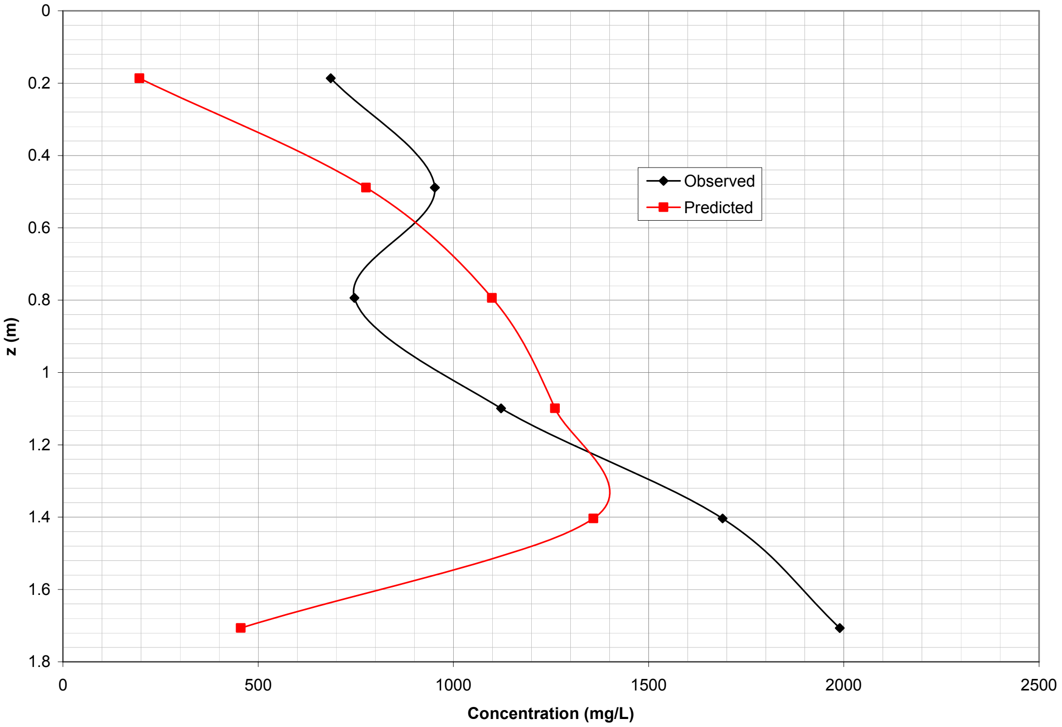

Figure 5 - 29: Observed vs. predicted concentration profiles after 60 minutes of settling for low initial concentration and replicate 2.

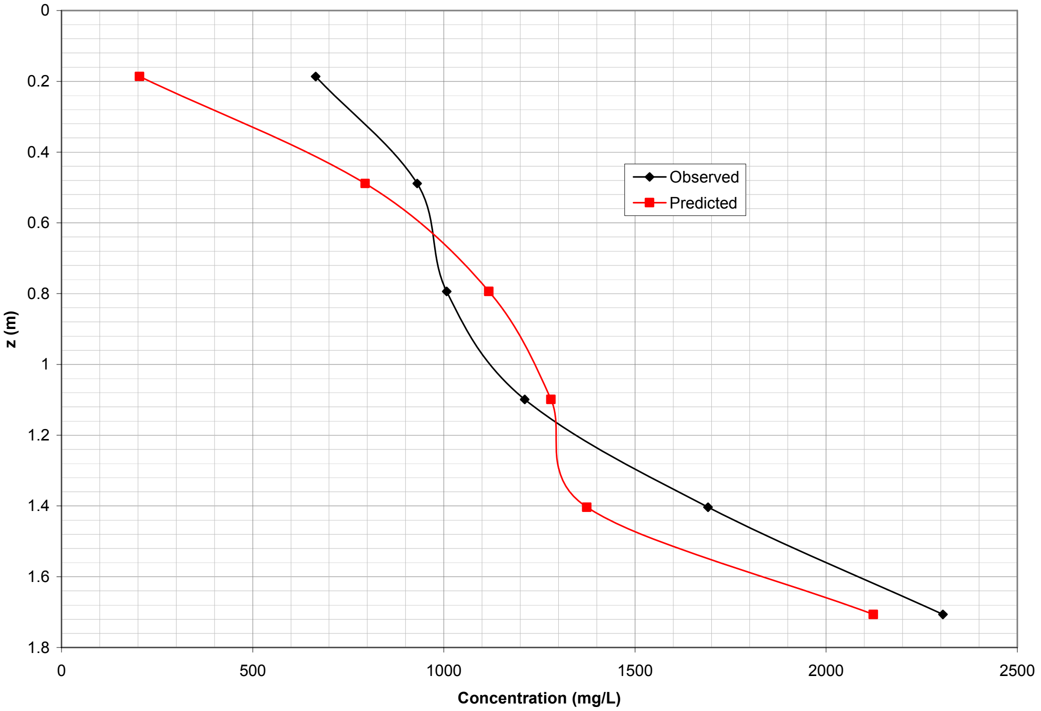

Figure 5 - 30: Observed vs. predicted concentration profiles after 60 minutes of settling for low initial concentration and replicate 3.

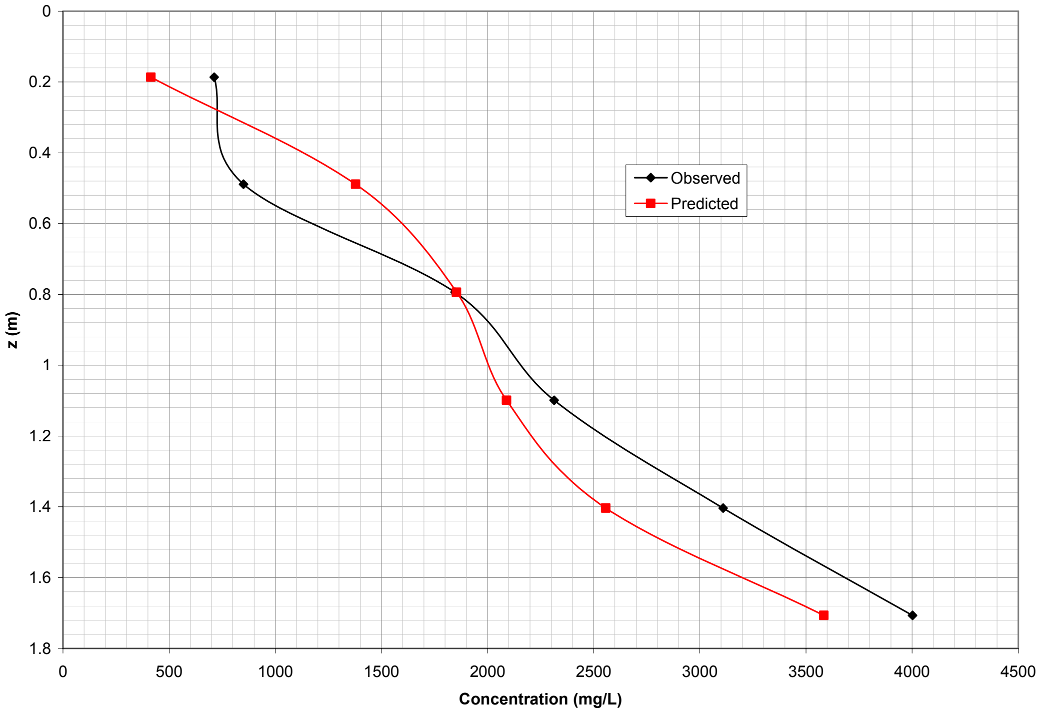

Figure 5 - 31: Observed vs. predicted concentration profiles after 60 minutes of settling for medium initial concentration and replicate 1.

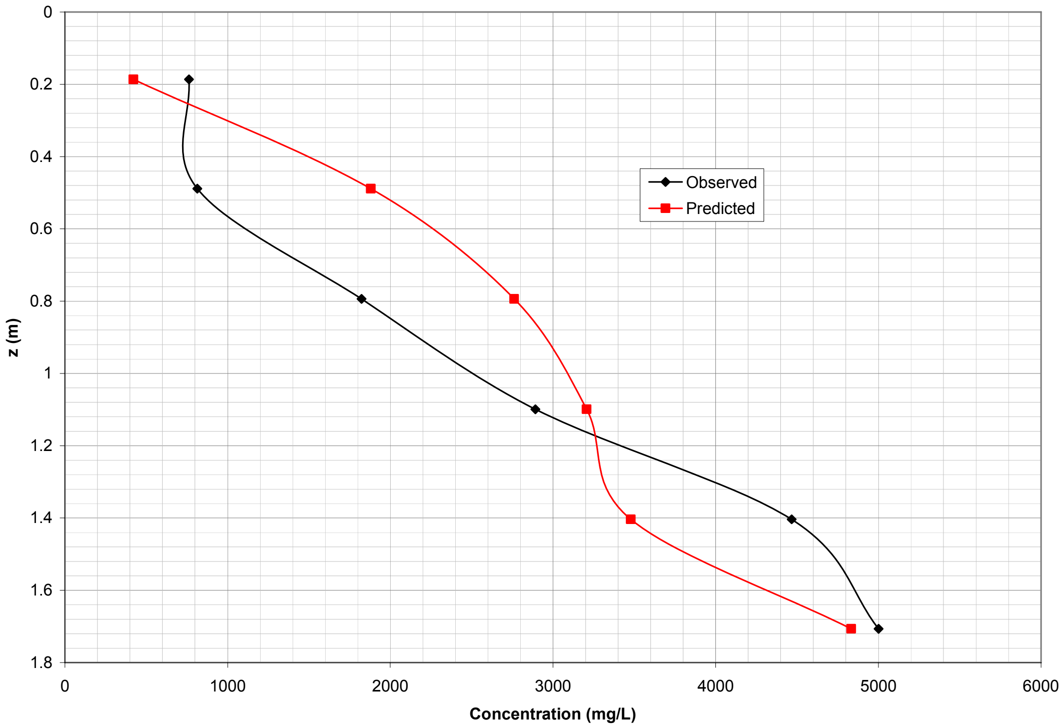

Figure 5 - 32: Observed vs. predicted concentration profiles after 60 minutes of settling for medium initial concentration and replicate 2.

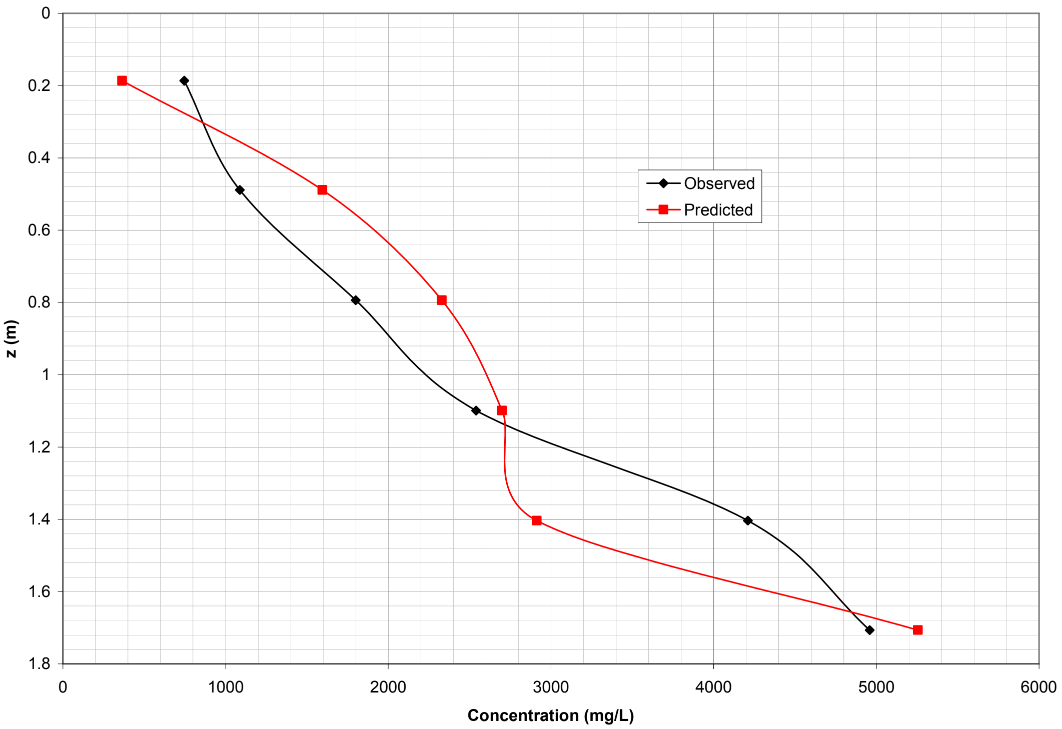

Figure 5 - 33: Observed vs. predicted concentration profiles after 60 minutes of settling for medium initial concentration and replicate 3.

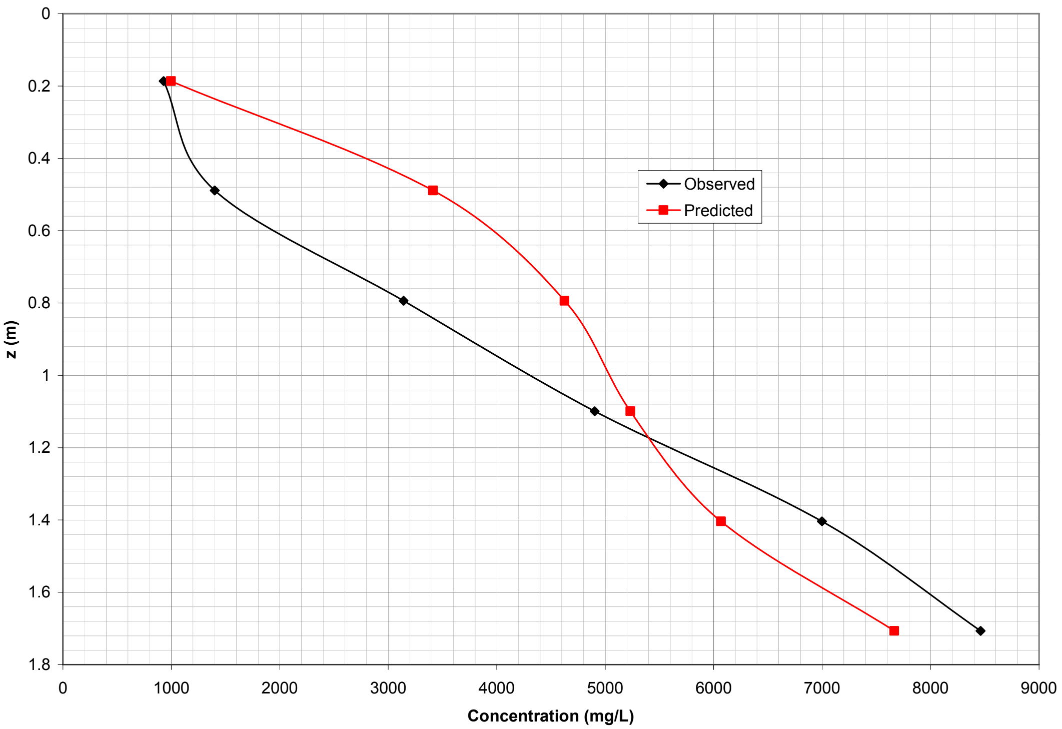

Figure 5 - 34: Observed vs. predicted concentration profiles after 60 minutes of settling for high initial concentration and replicate 1.

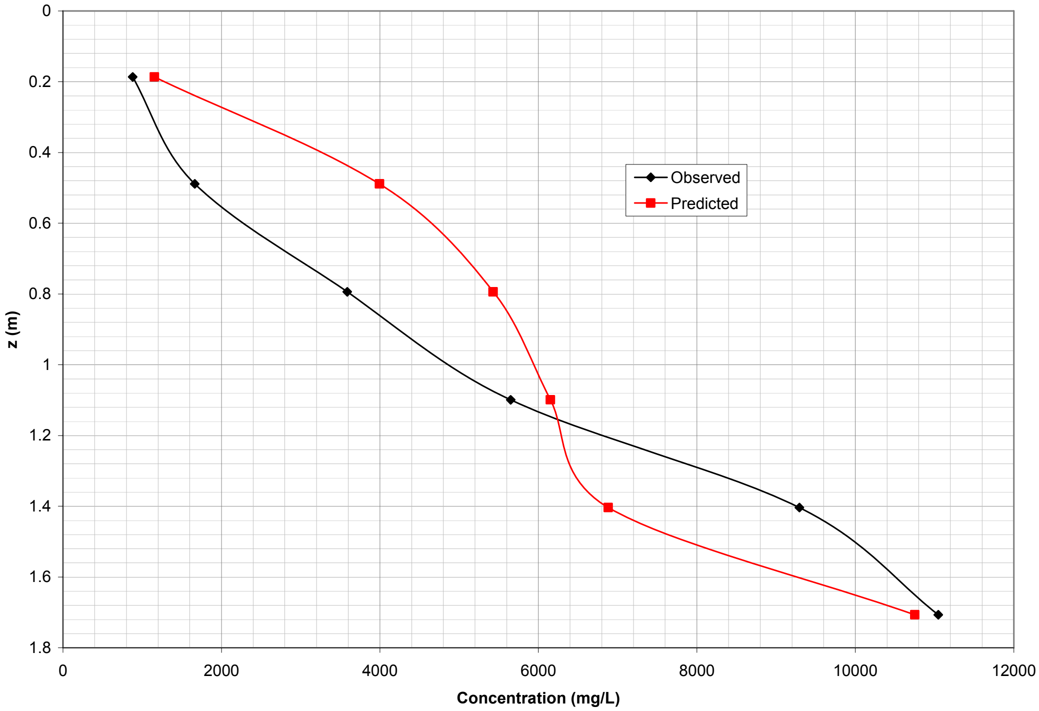

Figure 5 - 35: Observed vs. predicted concentration profiles after 60 minutes of settling for high initial concentration and replicate 2.

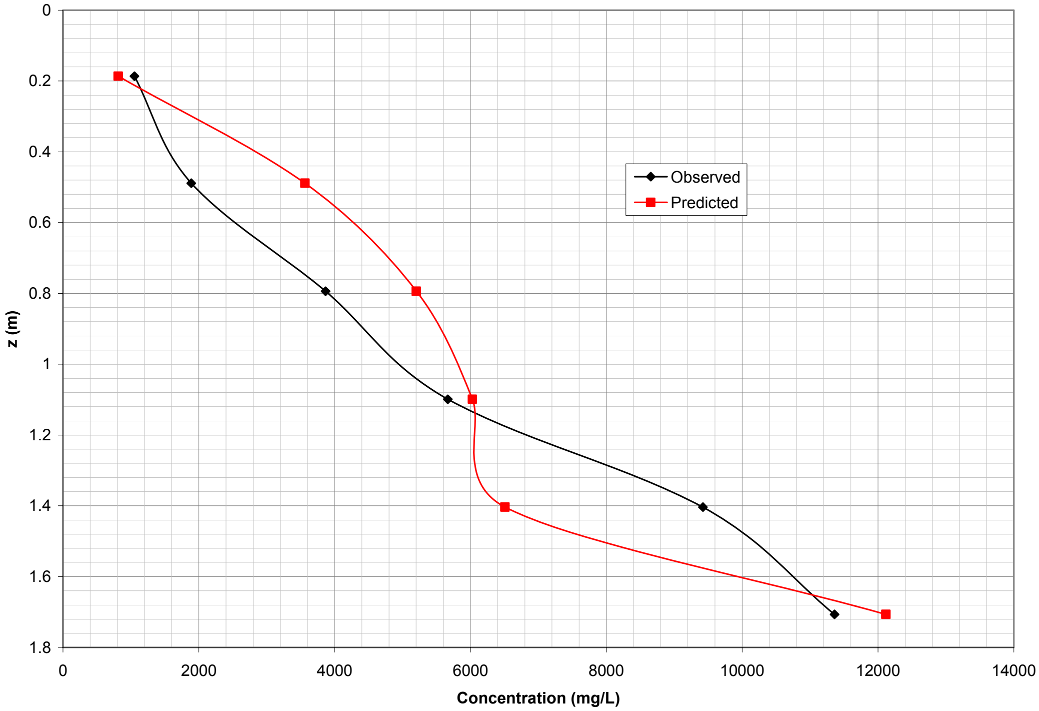

Figure 5 - 36: Observed vs. predicted concentration profiles after 60 minutes of settling for high initial concentration and replicate 3.

REFERENCES

[1.1] Federal Register, December 1, 2009 (Volume 74, Number 229) - Rules and Regulations. Pages 62995-63058.

[1.2] B.G. Krishnappan and J. Marsalek. Modeling of flocculation and transport of cohesive sediment from an on-stream stormwater detention pond. Water Res. 36: 3849-3859 (2002).

[1.3] N. Tambo and Y. Watanabe. Physical aspects of flocculation process-I. Fundamental Treatise. Water Res. 13: 429-439 (1979).

[1.4] I.A. Vialoulis and E.J. List. Numerical simulation of a sedimentation basin. 1. Model Development. Environ. Sci. Technol. 18(4): 242-247 (1984).

[2.1] S. Assouline, D. Tessier, A. Bruand. A conceptual model of the soil water retention curve. Water Resour. Res. 34: 223-231 (1998).

[2.2] L.M. Ayra, F.J. Leij, P.J. Shouse M.T. van Genuchten. Relationship between the hydraulic conductivity function and the particle-size distribution. Soil Sci. Soc. Am. J. 63: 1063-1070 (1999).

[2.3] M. Bittelli, G. S. Campbell, M. Flury. Characterization of particle-size distribution in soils with a fragmentation model. Soil Sci. Soc. Am. J. 63: 782-788 (1999).

[2.4] G.D. Buchan. Applicability of the simple lognormal model to particle-size distribution in soils. Soil Sci. 147: 155-161 (1989).

[2.5] G.D. Buchan, K.S. Grewal, A.B. Robson. Improved models of particle-size distribution: an illustration of model comparison techniques. Soil Sci. Soc. Am. J. 57: 901-908 (1993).

[2.6] W. Durner. Hydraulic conductivity estimation for soils with heterogeneous pore structure. Water Resour. Res. 30:211-223 (1994).

[2.7] M.D. Fredlund, D.G. Fredlund, G.W. Wilson. An equation to represent grain-size distribution. Can. Geotech. J. 37: 817-827 (2000).

[2.8] S.I. Hwang. Effect of texture on the performance of soil particle-size distribution models. Geoderma 123: 363-371 (2004).

[2.9] S.I. Hwang, K.P. Lee, D.S. Lee, S.E. Powers. Models for estimating soil particle-size distributions. Soil Sci. Soc. Am. J. 66: 1143-1150 (2002).

[2.10] S.I. Hwang, S.E. Powers. Using particle-size distribution models to estimate soil hydraulic properties. Soil Sci. Soc. Am. J. 67:1103-1112 (2003).

[2.11] A. Kravchenko, R. Zhang. Estimating the soil water retention from particle-size distributions: a fractal approach. Soil Sci. 163: 171-179 (1998).

[2.12] M. Matsushita. Fractal viewpoint of fracture and accretion. J. Phys. Soc. Jpn. 54: 857-860 (1985).

[2.13] A.B. Nesbitt, W. Breytenbach. A particle size distribution model for manufactured particulate solids of narrow and intermediate size ranges. Powder Technology 164: 117-123 (2006).

[2.14] A.N.P. Posadas, D. Gimenez, M. Bittelli, C.M.P. Vaz, M. Flury. Multifractal characterization of soil particle-size distributions. Soil Sci. Soc. Am. J. 65: 1361-1367 (2001).

[2.15] P. Rosin, E. Rammler. The laws governing the fineness of powdered coal. Journal of the Institute of Fuel 7: 29-36 (1933).

[2.16] S. Shiozawa, G.S. Campbell. On the calculation of mean particle diameter and standard deviation from sand, silt, and clay fractions. Soil Sci. 152: 427-431 (1991).

[2.17] V.P. Singh, K. Singh. Derivation of the Gamma distribution by using the principle of maximum entropy (POME). Water Resour. Bull. 21: 941- 952 (1985).

[2.18] T.H. Skaggs, L.M. Ayra, P.J. Shouse, B.P. Mohanty. Estimating particle-size distribution from limited soil texture data. Soil Sci. Soc. Am. J. 65: 1038-1044 (2001).

[3.1] Aldous, D.J. 1999. Deterministic and stochastic models for coalescence (aggregation and coagulation): a Review of the mean-field theory for probabilists. Bernoulli 5 (1):3-48.

[3.2] Bertoin, J. 2002. Eternal solutions to Smoluchowski's coagulation equation with additive kernel and their probabilistic interpretations. The Annals of Applied Probability 12 (2):547-564.

[3.3] Cheng, Z. and Redner, S. 1988. Scaling theory of fragmentation. Phys. Rev. Let. 60: 2450-2453.

[3.4] Cheng, Z. and Redner, S. 1990. Kinetics of fragmentation. J. Phys. A: Math. Gen. 23: 1223-1258.

[3.5] Dasgupta, S. 2002. A model of aggregation and dissociation. J. Phys. A: Math. Gen. 33: 339-344.

[3.6] Deaconu, M. and E. Tanre. 2000. Smoluchowski's coagulation equation: Probabilistic interpretation of solutions for constant, additive and multiplicative kernels. Annali della Scuola Normale Superiore di Pisa, Classe di Scienza 29 (3): 549-579.

[3.7] Dobias, B. 1993. Coagulation and flocculation: Theory and Applications. Marcel Dekker.

[3.8] Drake, R. 1972. A general mathematical survey of the coagulation equation. International Reviews in Aerosol Physics and Chemistry, Pergamon Press vol. 3: 201-376.

[3.9] Estrada, P.R and J. N. Cuzzi. 2008. Solving the coagulation equation by the Moments method. The Astrophysical Journal 682: 515-526.

[3.10] Flesch, J. C., P.T. Spicer and S.E. Pratsinis. 1999. Laminar and turbulent shear-induced flocculation of fractal aggregates. AIChE Journal 45 (5): 1114-1124.

[3.11] Hounslow, M.J. 1990. A discretized population balance for continuous systems at a steady state. AIChE Journal 36 (1): 106-116.

[3.12] Jlang, Q and B.E. Logan. 1991. Fractal dimensions of aggregates determined from steady state size distributions. Environ. Sci. Technol. 25: 2031-2038.

[3.13] Kostoglou, M. and Karabelas, A.J. 1999. On the self-similarity of the aggregation-fragmentation equilibrium particle size distribution. J. Aerosol Sci. 30 (2): 157-162.

[3.14] McCoy, B.J. 2002. A population balance framework for nucleation, growth, and aggregation. Chemical Engineering Science 57: 2279-2285.

[3.15] Melzac, Z.A. 1957. A scholar transport equation. Trans. Amer. Math. Soc. 85: 547-560.

[3.16] Peterson, T. W. 1986. Similarity Solutions for the Population Balance Equation Describing Particle Fragmentation. Aerosol Science and Technology 5: 93-101.

[3.17] Saffman, P. and J. Turner. 1956. On the collision of drops in turbulent clouds. J. Fluid Mech: 1 (16): PAGES

[3.18] Spicer, P.T. and S.E. Pratsinis. 1996. Coagulation and fragmentation: Universal steady-state particle-size distribution. AIChE Journal 42 (6): 1612-1620.

[3.19] Smoluchowski, M. 1916. Drei vortage uber diffusion, Brownshe molekularbewegung und koagulation vor kolloidteilchen. Physikalische Zeitschrift 17: 557-571.

[3.20] Spouge, J.L. 1983. Solutions and critical times for the polydisperse coagulation equation when a(x,y) = A+B(x+y) + Cxy. J. Phys. A: Math. Gen. 16: 3127-3132.

[3.21] Tambo, N. and Y. Watanabe. 1979. Physical aspect of flocculation process-I: Fundamental treatise. Water Research 13: 429-439.

[3.22] Tambo, N. and H. Hozumi. 1979. Physical aspect of flocculation process-II: Contact flocculation. Water Research 13: 441-448.

[3.23] Tanaka, H., S. Inaba and K. Nakazawa. 1996. Steady-state size distribution for the self-similar collision cascade. Icarus 123: 450-455.

[3.24] Yu, M., J. Lin and T. Chan. 2008. A new moment method for solving coagulation equation for particles in Brownian motion. Aerosol Science and Technology 42: 705-713.

[3.25] Zhang, J. and X. Li. 2003. Modeling particle-size distribution dynamics in a flocculation system. AIChE Journal 49 (7): 1870-1882.

[5.1] Aldous, D.J. 1999. Deterministic and stochastic models for coalescence (aggregation and coagulation): a Review of the mean-field theory for probabilists. Bernoulli 5 (1):3-48.

[5.2] Cheng, Z. and Redner, S. 1988. Scaling theory of fragmentation. Phys. Rev. Let. 60: 2450-2453.

[5.3] Cheng, Z. and Redner, S. 1990. Kinetics of fragmentation. J. Phys. A: Math. Gen. 23: 1223-1258.

[5.4] Deaconu, M. and E. Tanre. 2000. Smoluchowski's coagulation equation: Probabilistic interpretation of solutions for constant, additive and multiplicative kernels. Annali della Scuola Normale Superiore di Pisa, Classe di Scienza 29 (3): 549-579.

[5.5] Dobias, B. 1993. Coagulation and flocculation: Theory and Applications. Marcel Dekker.

[5.6] Drake, R. 1972. A general mathematical survey of the coagulation equation. International Reviews in Aerosol Physics and Chemistry, Pergamon Press vol. 3: 201-376.

[5.7] Estrada, P.R and J. N. Cuzzi. 2008. Solving the coagulation equation by the Moments method. The Astrophysical Journal 682: 515-526.

[5.8] Filbet F. and Laurencot P. 2004. Numerical simulation of the Smoluchowski coagulation equation. Siam Journal on Scientific Computing 25(6): 2004-2028.

[5.9] Flesch, J. C., P.T. Spicer and S.E. Pratsinis. 1999. Laminar and turbulent shear-induced flocculation of fractal aggregates. AIChE Journal 45 (5): 1114-1124.

[5.10] Hounslow, M.J. 1990. A discretized population balance for continuous systems at a steady state. AIChE Journal 36 (1): 106-116.

[5.11] Kostoglou, M. and Karabelas, A.J. 1999. On the self-similarity of the aggregation-fragmentation equilibrium particle size distribution. J. Aerosol Sci. 30 (2): 157-162.

[5.12] Krivitsky D.S. 1995. Numerical solution of the Smoluchowski kinetic equation and asymtotics of the distribution function. J. Phys. A: Math. Gen. 28: 2025-2039.

[5.13] Melzac, Z.A. 1957. A scholar transport equation. Trans. Amer. Math. Soc. 85: 547-560.

[5.14] Natural Resources Conservation Service. Erosion Control: Land Application of Anionic Polyacrylamide – Code 1050. Department of Natural Resources - Conservation Practice Standard.

[5.15] Reich I. and Vold R.D. 1959. Flocculation-deflocculation in agitated suspensions. I. Carbon and ferric oxide in water. J. Phys. Chem. 63: 1497-1501.

[5.16] Roa-Espinosa, A., Bubenzer, G.D. and Miyashita, E. 2000. Sediment and runoff control on construction sites using four application methods of polyacrylamide mix. National Conference on Tools for Urban Water Resource Management and Protection, Chicago, February 7-10, 2000, pp. 278- EPA.

[5.17] Saffman, P. and J. Turner. 1956. On the collision of drops in turbulent clouds. J. Fluid Mech: 1: 16-30

[5.18] Spicer P.T., Pratsinis S.E., Willemse A.W., Merkus H.G., and Scarlett B. 1999. Monitoring the dynamics of concentrated suspensions by enhanced backward light scattering. Part. Part. Syst. Charact. 16: 201-206.

[5.19] Smoluchowski, M. 1916. Drei vortage uber diffusion, Brownshe molekularbewegung und koagulation vor kolloidteilchen. Physikalische Zeitschrift 17: 557-571.

[5.20] Torrealba, S.F. 2004. Development and testing of a methodology to assess the potential field performance of flocculants". M.Sc. Thesis. Biosystems and Agricultural Department, University of Kentucky, Lexington, KY.

[5.21] Yu, M., J. Lin and T. Chan. 2008. A new moment method for solving coagulation equation for particles in Brownian motion. Aerosol Science and Technology

[5.1] G.K. Batchelor. Sedimentation in a dilute polydisperse system of interacting spheres. Part1. General theory, J. Fluid Mech. 119 (1982) 379-408.

[5.2] G.K. Batchelor and C.S. Wen. Sedimentation in a dilute polydisperse system of interacting spheres. Part 2. Numerical results, J. Fluid Mech. 124 (1982) 495-528.

[5.3] R. Bürger, K.-K. Fjelde, K. Höfler, and K.H. Karlsen. Central difference solutions of the kinematic model of settling of polydisperse suspensions and three-dimensional particle-scale simulations. J. Engrg. Math., 41 (2001) 167–187.

[5.4] L.C. Evans. Partial differential equations. American Mathematical Society graduate studies in mathematics, Vol. 19 (1998).

[5.5] K. Höfler and S. Schwarzer. The structure of bidisperse suspensions at low Reynolds numbers, in W.L. Wendland, W. Ehlers, H. Herrmann, C. Miehe, A.-M. Sändig and W. Schiehlen (eds.), Multifield Problems in Solid and Fluid Mechanics. Berlin: Springer Verlag, to appear.

[5.6] G.S. Jiang, D. Levy, C.T. Lin, S. Osher, and E. Tadmor. High-resolution nonoscilatory central schemes with nonstaggered grids for hyperbolic conservation laws. J. Num. Anal. Society for Industrial and Applied Mathematics, 35 (1998) 2147-2168.

[5.7] G.J. Kynch. A theory of sedimentation. Trans. Faraday Soc. 48 (1952) 166-176

[5.8] R. J. LeVeque. Numerical methods for conservation laws. Basel: Birkhäuser Verlag, Second Ed., (1992).

[5.9] H. Nessyahu and E. Tadmor. Nonoscillatory central differencing for hyperbolic conservation laws. J. Comput. Phys. 87 (1990) 408-463.

[5.10] J.F. Richardson and W.N. Zaki, Sedimentation and fluidization I, Trans. Instn. Chem. Engrs. (London) 32 (1954) 35-53.

[5.11] E.M. Tory, R. Bürger, F. Concha, and M.C. Bustos. Sedimentation and thickening: phenomenological foundation and mathematical theory (mathematical modeling: theory and applications). Kluwer Academic Publishers (1999).

VITA

Name: Sebastian F. Torrealba

Date of Birth: October 7th, 1973

Place of Birth: Santiago, Chile.

Current Position

Research Assistant (2007 - present)

Dept. Biosystems & Agricultural Engineer

University of Kentucky, Lexington, KY.

Supervisor: Dr. Richard C. Warner.

Education

B.Sc. in Civil Engineering

University of Chile

Santiago, Chile (1991-1995)

Emphasis: Structural Analysis, Hydraulics and Hydrology

Professional Civil Engineer (Chile)

University of Chile

Santiago, Chile (1995-1999)

Emphasis: Geotechnical and Environmental Engineering

Research: “Design of Geosynthetic Liner Systems for Municipal Solid Waste Landfill Applications.”

Advisor: Dr. Pedro Acevedo

Master of Science

Biosystems & Agricultural Engineering

University of Kentucky

Lexington, Kentucky (2002-2004)

Emphasis: Sediment control, water resources engineering and surface water quality modeling

Research: “Development and Testing of a Methodology to Assess the Potential Field Performance of Flocculants”

Advisor: Dr. Richard C. Warner

Master of Science

Applied Mathematics

Department of Mathematics

University of Kentucky

Lexington, Kentucky (2006 – 2008)

Emphasis: Numerical methods for hyperbolic systems of conservation laws applied the sedimentation model.

Advisor: Dr. Sung A. Kang

Professional Certification

Professional Engineering Chilean Certificate from the University of Chile.

____

(Sebastian F. Torrealba)

____

(Date)