An Experimental and Numerical Investigation of Resin Transfer Molding

by

Thomas Nowak

B.S., Mechanical Engineering
Worcester Polytechnic Institute, May 1987

M.S., Mechanical Engineering
Worcester Polytechnic Institute, May 1990

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Signature of Author: _____
Department of Mechanical Engineering
June 1996

Certified by: _____
Professor Jung-Hoon Chun
Chairman, Doctoral Thesis Committee
l Engineering

Accepted by: _____
Professor Ain A. Sonin
Chairman, Graduate Committee
Department of Mechanical Engineering

An Experimental and Numerical Investigation of Resin Transfer Molding

by
Thomas Nowak

Submitted to the Department of Mechanical Engineering
on June 24, 1996 in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy in Mechanical Engineering

ABSTRACT

Impregnation of dry fiber preforms is a common step in many composites manufacturing processes. In resin transfer molding (RTM), for example, a dry fiber preform is produced and then impregnated with a polymer matrix. Complex preforms are fabricated using techniques such as weaving and braiding. Since designers typically do not have the means to efficiently and correctly account for variability in the preform structure, tooling and process design are commonly accomplished by trial-and-error, hindering cost-effective production. Mold filling problems are manifested as voids and resin rich regions occurring at three distinct length scales: the fiber, tow, and part scales. Current mold-filling software typically focuses on part-scale flows and relies on empirically determined preform properties. Analyses based on first principles and scaling up of results from the fiber scale can be computationally intensive. In addition, the current state of the art in experimental investigations does not provide the information on local variation of the preform structure that is needed to refine existing computational models.

The objective of this thesis was to reduce the trial-and-error in fiber impregnation tooling and process design by developing tools with which designers can investigate the effects of variability within the fiber preform. A method to directly observe and quantify the flow within the preform was developed. Based on matching the refractive indices of the fibers and a simulated resin, this flow visualization method can be combined with photometric measurements and image processing to quantitatively describe the impregnation process at the various length scales within the preform. Examples of flow visualization results at the fiber and tow scales are presented to illustrate the method. In order to generalize the experimental results, a computer model of preform impregnation at the fiber and tow scales was developed. The model, based on a three-dimensional network of control volumes, provides a design tool that can be used to study assumptions on the causes and effects of variability at this scale. Results of a study of void entrapment during uni-directional flow at the tow scale are presented as an example.

Thesis Committee:

Jung-Hoon Chun, Chairman
Edgerton Associate Professor, Department of Mechanical Engineering
Prof. Timothy G. Gutowski
Professor, Department of Mechanical Engineering
Prof. Anthony T. Patera
Professor, Department of Mechanical Engineering

For Carolyn

Acknowledgments

This thesis was made possible, in part, by support from Lord Corporation and Caterpillar, Incorporated.

There are many people who need to be thanked. So many, in fact, that I will surely miss a few. Therefore, I would like to thank in advance all those people who helped me get those assignments, presentations, papers, and this thesis done. Thanks.

My advisor, Professor Jung-Hoon Chun, taught me many lessons during my time at MIT. The concern for his students and hard work that he showed every day will always serve as a model in whatever career I follow.

The many talented individuals with whom I was privileged to work during this time helped make the days and nights in the lab both fun and stimulating. Doctor Greg ("I hate to be called Doctor") Dillon was and is a great friend. Greg gave me valuable professional and personal advice and introduced me to the beauty of "Sport". Doctor Pyongwon Yim was always ready with words of encouragement, from the bad old days in CP^3 to the present. Mr. Steve Ueda, whose research complemented mine, brought some of the aloha spirit to the lab and taught me how to "talk story". Mr. Stuart Muter could always make us laugh with his quick wit and, even more importantly, introduced me to *Seinfeld.* I would also like to thank: Mr. and Mrs. Christian and Honor Passow, Dr. Chen-An Chen, Mr. Sukyoung Chey, Mr. Paul Acquaviva (bound for that elusive No. 1 spot), Mr. Wesley Williams, Mr. Mark Hytros (a fellow Polska Boy), and Ms. Jeanie Cherng.

Like any other thesis done in the Laboratory for Manufacturing and Productivity, the acknowledgments would be incomplete if they did not include mention of Mr. Fred Cote, Mr. Gerry Wentworth, Mr. Bob Kane and, Mr. Kevin Baron. Fred was always available with advice on how to tackle tough problems...and good conversation while I was chained to the Xerox machine. My duties as a teaching assistant for the graduate and undergraduate manufacturing classes, 2.810 and 2.86, become less of a chore and more fun because of Gerry's good-natured approach to work.

I thank my family and friends for their understanding and patience while I pursued yet another degree. Although they might not have understood exactly what it was that I did

with my time, my parents were always supportive (if not a bit bemused at times). My brother and his wife were nearby and always willing to help. My sister not only sent me juicy tidbits from the Big City, but was also a great roommate when she stayed with me.

Finally, my most sincere and greatest thanks go to Carolyn Sasaoka. She not only waited patiently for me to finish my Ph.D., but she also gave the support, advice, and love that I needed to accomplish this goal. Thank you, darling.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

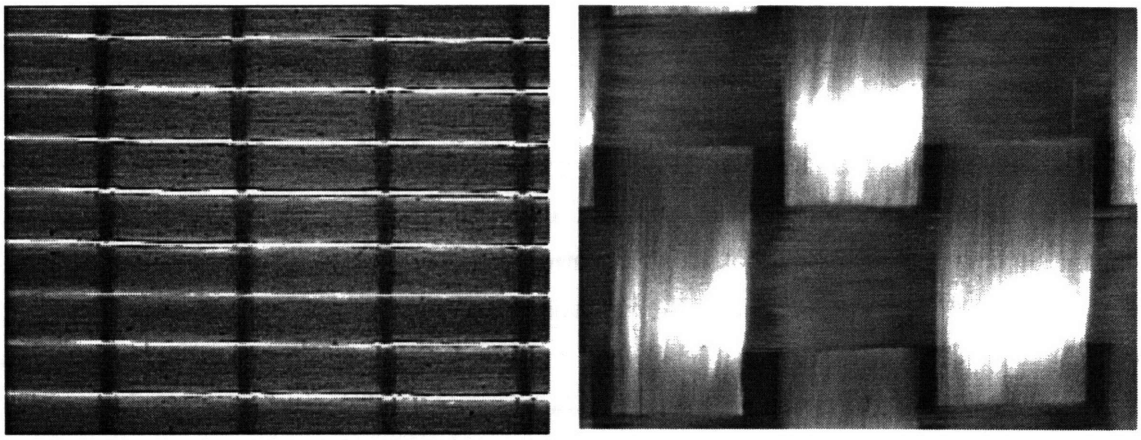
Impregnation of dry fiber preforms is a common step in many composites manufacturing processes. In structural reaction injection molding (SRIM) and resin transfer molding (RTM), for example, a dry fiber preform is produced and then impregnated with a polymer matrix. Utilization of textile technology (e.g., weaving and braiding) makes possible the fabrication of complex, 3-dimensional preforms. Examples of common preform materials are shown in Figure 1.1. Because of an incomplete knowledge of the resin infiltration process, however, tool design and process parameter selection is commonly accomplished by trial-and-error. As a result, cost-effective production is difficult. As shown in Figure 1.2, mold filling problems are usually manifested as voids and resin rich regions occurring at three distinct length scales: the fiber, tow and part scales.

Effective process simulation has long been recognized as a key to more efficient tooling and process development. The existence, however, of significant quality drivers at disparate length scales raises the question of how much detail is enough for a process model to be meaningful without being too cumbersome.

The objective of this thesis was to develop a framework for preform impregnation tooling and process design. Using the proposed framework, industrial users should be able to understand and predict (with the ultimate goal of avoiding) defect (e.g., void and resin rich region) formation at the various length scales within the preform. In addition, knowledge of the scaling behavior of the physical phenomena within the process can aid in the identification of appropriate averaging lengths for the the determination of effective permeability coefficients for use in efficient process modeling.

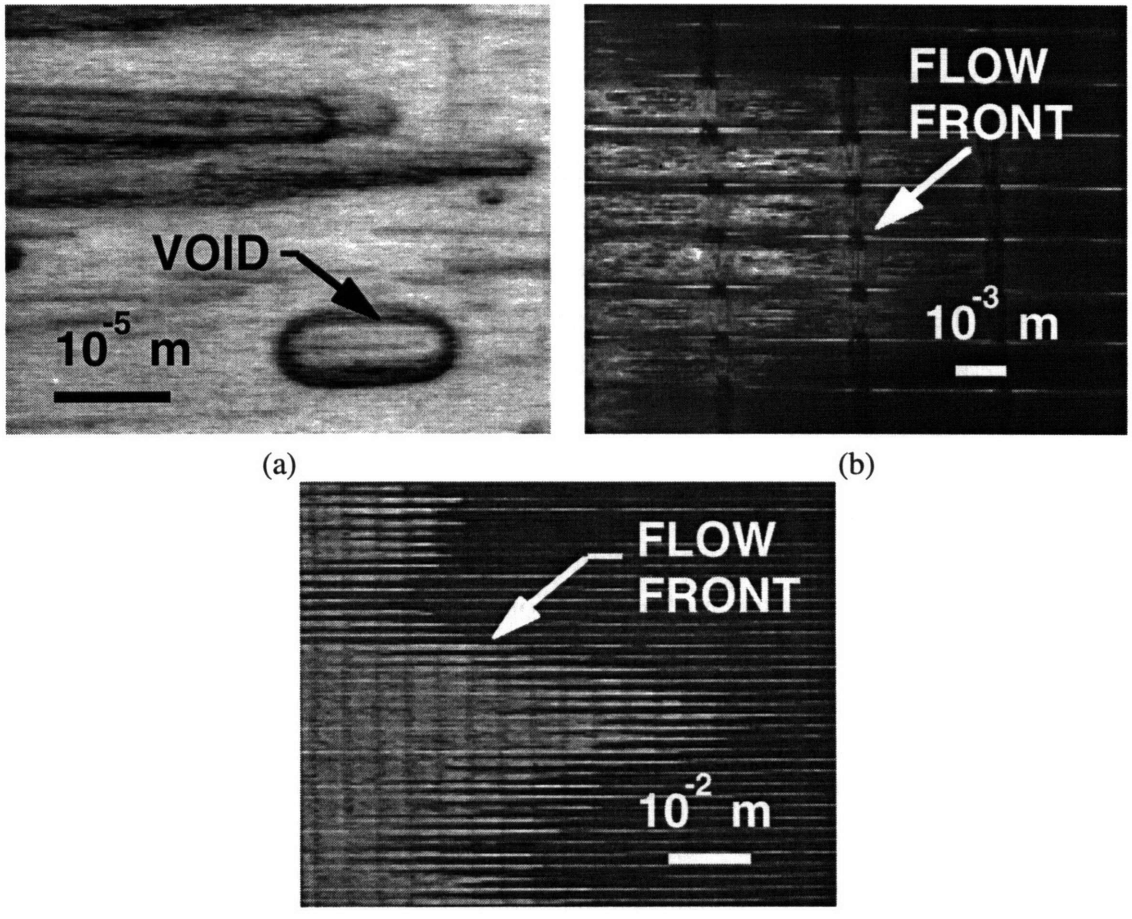
Although much effort has been devoted to simulation of preform impregnation at the part scale, very little modeling work has been done relating fiber and tow-scale flows to processing conditions and resulting part quality. One of the main quality problems formed at this length scale is the presence of voids. Voids can occur either in the part or on its surface. Both types of voids can cause degradation of mechanical properties. In addition, surface voids can result in costly secondary operations before the part is ready for subsequent processes such as painting.

Consequently, this thesis concentrated on the first of the two goals stated above. In particular, fiber and tow-scale flows were related to processing conditions and resulting part quality as quantified by void content. The issues of permeability calculation and selection of appropriate averaging length scales will be considered in relation to the incorporation of fiber-scale void formation within the simulation.



(a) (b)

Figure 1.1 Examples of typical preform architectures: (a) discrete tow, unidirectional fabric, (b) plain weave.



(a) (b)

(c)

Figure 1.2 Basic length scales within the preform: (a) fiber scale, (b) tow scale, (c) part scale.

1.1 Process Description

Fiber impregnation processes can be divided into two main groups based on preform type: random fabric and aligned, continuous fiber reinforcements. The term random fabric refers to the orientation of the fibers in the preform and can be composed of either randomly oriented long or short fibers. Random reinforcements are typically used in applications areas, such as automotive, where costs and processing times need to be minimized. Random fabrics are usually made of glass fibers in these applications and, therefore, material costs are kept relatively low. Parts made with random reinforcements have relatively low fiber volume fractions ($V_f \leq 0.4$) and, therefore, can be impregnated at relatively fast rates. These parts have typically been used for secondary structures, but recent developments with foam cores surrounded by an impregnated random mat have made low cost, primary structures possible.

In the literature, Structural Reaction Injection Molding (SRIM) is the most common process name associated with impregnation of random fabrics. The key feature of SRIM-type processes is that the resin is mixed with the catalyst "on the fly" (i.e., in a special mixing head just outside the mold) during the injection phase. In order to achieve proper mixing, the resin is injected at relatively high rates. This has led to the development of relatively low ($\leq$ 100 cp) viscosity resins suitable for the process. Typical filling speeds in successful applications are reported on the order of 10^0-10^1 cm/s.

The other main group of impregnation processes occur in highly ordered reinforcements made of aligned, continuous fibers. These reinforcements include the unidirectional, woven, and braided types. Unidirectional fabrics can be further divided into fabrics with and without discrete fiber bundles, or tows. These reinforcements are typical of advanced composites used in high-performance sporting goods and aerospace applications. The cost of these materials, in part because of the increased order in their structure and in part because of the fiber materials used, is higher than that of the random fabrics. The fiber-volume fractions of these materials are also typically higher than the random fabrics. More importantly, however, the fiber-volume fraction can vary appreciably within the preform because of the non-uniform nature of the preform miscrostructure (i.e., fabrics made of woven tows will have alternating regions with no fibers, fibers in a single tow, and fibers in two criss-crossing tows). Because of the high fiber-volume fractions involved, the impregnation rates in these materials is slower (filling speeds typically less than 10^0 cm/s) than for the random fabrics. The processes involved with impregnation of these types of reinforcements include traditional resin transfer molding and a number of related processes including Resin Film Infusion Processing (RFIP) which is a cross between RTM and autoclave curing of a prepreg. Resins used

include epoxies, uretheanes, and polyesters. In these processes, the resin and catalyst are mixed and then injected (from a single vat) into the preform. As new resins are introduced, the trend is to decrease viscosity (~100 cp). Surface tension values are typically within the range of 20-60 x 10^{-3} N/m.

1.2 State of the Art

Discussion of the state of the art in RTM needs to consider both the technology as practiced in industry and the advances in process research and development in both industry and academia. In addition, selected results from related research in other areas, such as petroleum and soil engineering, which treat the general problem of flow through porous media are discussed because of their importance to RTM process modeling.

1.2.1 Industrial Practice

Despite the initial claims of early proponents, resin transfer molding has not proven to be the long sought, cost-effective solution to advanced composites manufacturing. The current state of the art is that RTM parts are typically produced in low to medium volumes, the main obstacles to high-volume production using resin transfer molding are the costs associated with preform, tooling, and process development. In high volume applications, such as the automotive industry, the chief obstacle is cost-effective preforming at high rates. Current automotive applications of RTM get around this problem by using lower-cost, easier-to-form preform materials such as random glass mats. In addition, most automotive RTM parts are body panels and other secondary (i.e., low-load) structures.

Development of the process for primary structural components has been limited to applications where customers are traditionally willing to pay a premium for performance. These areas include sporting goods, such as a $200 single-piece RTM hockey stick [1] and the military. The military, in particular, has led the push for large structural parts with proposed 6,000-kg, 60-mm thick tank panels [2] and aerospace components such as the proposed 24-foot long, graphite-epoxy keel beam on the RAH-66 Comanche scout helicopter [3]. In both of these proposed applications, tooling and process development costs are still considerable and the subject of on-going research.

Tooling and process design issues typically involve control of flow front progression and fiber wet-out during impregnation and heat transfer during the subsequent cure cycle of the resin. Problems that arise from poor tool design include excessive porosity at the fiber and tow scales and larger dry spot formation at the part scale. Large dry spots typically form because of unplanned, preferential resin flow through low-fiber regions in the mold

(often caused by a poor fit between the solid tooling and the compressible, fiber preform). that can entrap large pockets of air within the mold. They can be eliminated by strategic placement of resin injection ports and exhaust vents within the tool.

Fiber-scale voids are attacked by aiding the wet-out of the fibers through resin selection, schemes to lower resin viscosity by raising temperature, or by vibrating the mold to shake out whatever voids might be formed during the process.

All of these strategies are currently employed on a trial-and-error basis. As parts become larger and more complex, however, the use of this approach becomes prohibitively expensive.

At present, RTM parts are typically used for either structural or surface applications but not both. When used for surface applications, significant amounts of secondary work, such as sanding and filling, is required to prepare the part surfaces for subsequent operations. Advances in sheet molding compound (SMC) technology have produced nearly ready to use parts straight from the mold without any secondary operations, but these parts are usually not used for structural members. A major drawback with SMC is the inability to control the exact location of the fibers during the process. This results in unpredictable and inconsistent (in the sense of spatial variation) mechanical properties for the finished part. Because the fiber reinforcement, or preform, in RTM is placed and then held in place in the tooling before the impregnation stage of the process, RTM can potentially produce parts with predictable mechanical properties.

Resin transfer molding hasn't been widely used for these parts, however, because typical processes produce parts that contain significant amounts of small pores on surfaces of the parts. For parts that are not visible to the customer, this surface porosity is not critical as long as bulk properties are acceptable (although in structural parts subjected to cyclic loading, this surface porosity can act as a stress raiser and, therefore, a fatigue strength reducer). For parts that are to be visible, such as body panels, significant amounts of secondary operations (e.g., sanding, filling) are required on each part before it can be painted. If these finishing operations are neglected, then paint which fills the surface pores during painting is explosively ejected during the paint baking cycle, creating pops, or pinholes, in the surface of the part. A recent study by a major manufacturer indicated that the secondary operations on a sample part were the bottleneck in the manufacturing process and accounted for approximately 10% of the part's total manufacturing cost.

Other recent innovations include:

- Patented SCRIMP process [4], one-sided tool translates to lower tooling costs but also limits parts to applications with only one customer visible surface

- Workers in Germany [5] have done some model experiments where they claim to have reduced surface faults by filling the mold with gaseous acetone and then increasing cavity pressure during cure.

- Aero Detroit, a molder for the Chrysler Viper, has claimed that near class-A surfaces were produced by carefully controlling mold face temperatures (unequal) during the process. This technique was only used with sprayed nickel-shell tools. Whether or not the technique is easily applied to other resins and tool materials is not clear. On parts made with epoxy tools, Aero Detroit uses a gel coat. Because of the added labor, mess, and possibility of damage to the gel coat during handling, this is not an attractive alternative [6,7].

Most industrial applications involving simulation of RTM processing use finite element analysis, typically assuming that the part to be modeled can be decomposed into a series of thin, shell segments. The assumed thin shell structure allows averaging of properties and results through the thickness of the part without significant loss of accuracy at the larger length scales. As with all averaging operations, however, information regarding the shorter length scales is lost. Issues such as micro-scale void formation, for example, can not be dealt with directly when using such an averaging scheme. In addition, without fiber-scale information, values of the preform permeability must be assumed or approximated using empirical relations.

In unsaturated flows (i.e., mold filling) the advancing flow front is treated by updating the saturation of control volumes in an explicit, finite difference time formulation. Combining the FEM solution for the pressure field during each time step with the finite difference time formulation is referred to by some authors as the FEM/CV approach. An interesting variation on the FEM/CV method is the modification of an injection molding simulation package by FIAT. However, as with all of the approaches discussed so far, the selection of effective permeability values is problematic.

1.2.2 RTM Research

The current state of the art in RTM research was summarized in a recent NIST workshop on manufacturing polymer composites by liquid molding [8]. The workshop participants identified five major areas in RTM needing further work:

1) development of accurate and cost-effective model validation and test methods (preferably executable on a workstation as opposed to a large and expensive computer),

2) high-speed processing (both in the area of cost-effective preform fabrication and reducing cycle times without sacrificing product quality)

3) development of design tools based on knowledge of preform architecture,

4) understanding reinforcement/resin interactions during flow and their effects on processing windows, and

5) heating and rheokinetic effects during flow.

The bulk of the research on preform impregnation can be divided into two main groups adopting theoretical and/or computational approaches based on solution of Darcy's equation for flow through porous media [9-11] and experimental approaches concerned with measurement of the preform permeability [12-14] and correlation of processing conditions with final part quality [15,16]. Current part-scale computational efforts suffer from a reliance on empirically determined permeability values and analyses based on scaling up results from the fiber scale can be computationally intensive. A large body of work has been written on saturated flows through porous media. Much of the work has concerned calculating permeability values as a function of fiber volume fraction and fiber diameter. More recent efforts have examined the effects of variability at the micro-scale and how that affects the saturated flow or permeability at the larger length scales.

The current state of the art in experimental investigations does not provide the information on random variation of local permeability that is needed to refine existing computational models. In general, the relationships between the various process parameters and final product quality are poorly understood, in part, because it has always been necessary to study these relationships indirectly.

Current studies relating processing conditions to the quality of fiber wetting [15,16] commonly rely on micrographic studies of cured samples. Consequently, the dynamics of the impregnation process are not recorded. Previous studies of preform infiltration using flow visualization techniques [16,17] were limited to observations of the flow within the narrow gap between the preform and observation window since the fibers in the preform obstructed observation of flow inside the preform. In addition, studies using scale models of idealized fiber arrays to correlate permeability values [18] and void formation [18,19] with preform structure have been only marginally useful because they did not realistically take into account the variability that occurs in actual preforms. Studies using sensor arrays [20,21] provided information only at the array nodes and, therefore, also provided an incomplete picture of the infiltration process.

1.2.3 Related Research in Other Areas

The flow of resin through the fiber preform during RTM can be considered a special case of the general problem of flow through porous media. A large body of work has been written on saturated flows through porous media. The reviews by Scheidegger [22] and Bear [23] are extensive and often cited in the literature. Of particular interest to modeling of advanced composites manufacturing are the advances in homogenization theory that were developed in the context of incorporating laboratory-scale results on soil samples to simulations of entire flow fields at geological scales and the use of percolation theory to describe flow within porous media as a function of both flow conditions and the topology of the interstitial pore space. Incorporation of parallel processing and Monte Carlo techniques with homogenization theory [24,25] has enabled prediction of upper and lower bounds on transverse permeabilities of random fiber arrays but is still computationally intensive.

Using the analytical results of Fisher and Essam [26], Larson, Scriven and Davis [27] developed a percolation-based theory of residual non-wetting phase saturation in soil sample water floods (i.e., wetting). Describing the topology of the porous medium as a Bethe lattice of a given coordination number (i.e., average number of neighbors for each lattice site, or "pore"), they were able to make predictions of the void content of the porous material that closely matched experimental results. The key finding of interest here is that given a pore space topology, it is possible to predict flow conditions in which it is not possible to entirely flush voids out of the porous medium. As such, this gives a framework for setting lower limits on impregnation rates in composites manufacture. Unfortunately, due to variability in fiber packing, characterizing the pore space within typical fiber preforms is not trivial and, therefore, these results can only be used as a general guide.

1.3 Approach

The research was carried out using two complementary approaches: an experimental investigation of the flow within the fiber preform and computer simulation of the process. The experimental portion of this thesis consisted of two main parts: development of a quantitative flow visualization technique appropriate for the study of preform impregnation in advanced composites manufacturing and application of this technique to study fiber-scale flow and void entrapment within model fiber tows. Modeling work was then undertaken to incorporate and then generalize the experimental data. This work focused on developing a workstation-based computer model of the flow at the fiber and tow scales of a model

preform. Of key interest here was the development of a design tool that can be used to evaluate the effects of processing and material variables on part quality. The metric used to quantify quality was the predicted void content of the part.

1.3.1 Experimental

The experimental portion of this thesis employed two old and powerful tools, flow visualization and scale modeling, to study the impregnation of the fiber preform in advanced composites manufacturing. The first tool, flow visualization, enabled direct observation of the complicated flows within the preform as they were occurring. The results of preliminary visualization experiments of flows within commercial reinforcing materials then suggested the use of the second tool, scale modeling, to study in detail the parameters that led to the observed phenomena.

The idea of matching the refractive index of the impregnating fluid with that of the solid matrix in a model porous medium can be traced to the Christiansen filter [23,28], a variation of the basic sand box model. The basic technique involves packing a transparent box with glass beads (to act as the porous medium) and then filling it with a fluid whose refractive index matches the glass beads. This makes the entire model transparent. A dye is then injected into the model. Because the model is transparent, the motion of the dye through the pore space formed by the glass beads can be observed. Further refinements of the method have included making photometric measurements of the light transmitted by the model in order to trace the motion of the dye and the use of X-rays or γ-rays instead of light. These measurements are usually made on two-dimensional flow domains.

The main drawback of using the Christiansen filter to study preform impregnation is that the flow within the model is not truly unsaturated but, rather, an immiscible displacement of one liquid by another. Consequently, among other things, the initial condition of dry fibers can not be met using this approach. In addition, the method requires matching physical properties of two fluids, instead of one, with those of the model fibers and with each other. Although not insurmountable, this added requirement can make the Christiansen filter cumbersome.

In 1952, Müller applied the observed increase of transmittance as a sample was wetted to make photometric measurements of the impregnation of paper filters and in the process proposed an empirical relationship between the amount of sample saturation and observed transmittance [29]. His experiments were successful because, even though the refractive indices of the filter paper and impregnating fluid were not perfectly matched, the samples that he observed were thin and, as will be discussed in Section 2.2, the resulting number of potential scattering interfaces was relatively small.

Use of the scattering of transmitted light due to the refractive index mismatch between resin and wetting defects, such as voids, within composite panels has been used as a means of quality inspection [30] in finished parts. Measurements have also been made of the transmittance of nominally transparent composites for windshields to measure the effects of slight mismatches in the refractive indices of the resin and reinforcing fibers [31]. The method has been used to visualize the mobilization of a voids in a bed of glass beads [32]. More recently, experiments on the onset of void formation in random fiber mats [33,34] were conducted with a model fluid selected to match the refractive index of the fabric fibers and then, after the impregnation process was halted, measurements were made of the resulting voids. Another approach to increasing visibility within the impregnated fiber bed has been to employ colored dyes injected into fiber beds impregnated with "clear" fluids [35].

All of these recent efforts looked at the end effects caused by defect formation, not the actual processes that caused them. Consequently, one of the goals of this thesis was to extend the technique of refractive index matching based flow visualization to observe the key mechanisms of defect formation while they were occurring. In order to do this, however, it soon became evident that the observation of the fiber-scale phenomena would be greatly facilitated by employing scale models of the fiber tow.

Scale models have been employed by others in the study of fiber wetting. Of particular interest here were the efforts by Elmendorp and During [36] and Parnas *et al.* [37]. Elmendorp and During employed a two-dimensional array of metal fibers impregnated by corn syrup to study the effects of changing contact angle on void formation in flow transverse to the major axis of the fibers. Parnas *et al.* also used a two-dimensional model to study the formation of voids in transverse flow, but they grouped the fibers into closely packed "tows" within the model to simulate the structure of an actual fabric. The goal of this study was to verify a void entrapment model based on the permeability contrast between the fiber tows and the relatively large gaps that separated the tows. Both studies produced useful results for the case of flow transverse to the fibers, but little has been done on flow along the major axis of the fibers or on the more realistic case of combined axial and transverse flows.

The fiber-scale experiments described below begin to address both the axial and combined flow cases because the models that were constructed allowed for significant flow in both the axial and radial directions within the tow.

1.3.2 Computer Model

The computer simulation of the fiber and tow scale flow developed as part of this thesis belongs to the general family of network models that have been discussed in [38-40] for saturated flow and in [41] for unsaturated flow. As in [41], the simulation developed in this thesis incorporates the control volume analysis used by others [42-45] in order to model the motion of the advancing flow front.

The quality of the simulation results is, in a large degree, dependent on the quality of the description of the tow geometry used as input for the rest of the analysis. This has been, from a practical viewpoint, the shortcoming of much of the previously published work in this area. As noted by Sangani and Yao [46], who studied flow in random fiber arrays, the flow transverse to the fibers is particularly sensitive to the exact configuration, or arrangement, of the fibers in the modeled geometry. In a review of the existing data, Åström, Pipes, and Advani [47] found that while various theoretical results based on ideal packing geometries seemed to correlate well, the same could not be said of the various theories when applied to actual (i.e., non-ideal) fibers. Of particular interest, was their finding that even those models, such as Sangani and Yao's, that were based on many realizations of random fiber arrays did not correlate well with actual fibers. Clearly, these models were missing an important aspect of the behavior of actual fiber bundles. Åström *et al.* postulated that this aspect was related to the variability in fiber arrangement, alignment, and fiber volume fraction within the tow.

The approach adopted here was based on work done on the deformation of lubricated fiber bundles [48-49]. Their work resulted from the observation that the consolidation behavior of aligned, lubricated fibers could be described by modeling the fibers as slightly curved beams that, under increasing load, come into contact with one another at increasing frequency and, therefore, act as a stiffening spring. This fiber waviness can, in an average sense, be related to the average fiber volume fraction and the packing geometry of the tow as a fiber contact frequency that results in a periodic structure to the interstitial pore space of the tow. The simulation uses this conceptual framework as the basis for the definition of the model geometry.

Although, as reported by Kim, McCarthy, and Fanucci [50], the compression behavior of dry fibers is different in many respects from that of lubricated fibers (e.g., displaying hysteris during repeated loading-unloading cycles and requiring a greater load to achieve a given fiber volume fraction than lubricated fibers), the general behavior of both lubricated and dry fibers is similar. For the present analysis of a dry preform that is presumably loaded only once, during the clamping stage of the injection process, these differences have been ignored. As indicated by compression experiments performed on the model fibers

used in the scale models, this proved to be a reasonable assumption since it was possible to fit the observed behavior of the dry fibers to the theory developed for lubricated fibers.

Models of void formation in RTM have focused primarily on the mechanical entrapment of air within the preform by the advancing flow front [51-53] and on the effects of processing parameters, such as level of vacuum assistance, on the final void content once the voids are formed [54]. Parnas and Phelan [51] considered the case of transverse flow in a fabric composed of distinct fiber tows. The models in [52-54] considered axial flow (i.e., along the fibers).

In the analysis by Parnas and Phelan, the permeability contrast between the inter-tow spaces and the inter-fiber spaces within the tows leads to the presence of two distinct time scales for macroscopic mold filling and full wet-out of the tows. With the time to fully impregnate the tows being much greater than the time to fill the mold, tows become surrounded by fluid as the global flow front passes individual tows, thereby, entrapping the air within the tows. Subsequent experiments using scale models [37] confirmed the basic mechanism proposed in [51].

Chan and Morgan also considered flow in fabrics composed of distinct fiber tows. In their analysis, they assumed that the flow in the inter-fiber gaps, or channels, would be much quicker than in the axial flows within the tows. Consequently, filling of the tows would be characterized primarily by radial flow into the tows from adjacent channels. They argued that void formation within the tows occurred mainly at the region close to the local flow front and that it would be minimized if the flow front within the tow, characterized by the region of full wet-out, did not lag too far behind the flow front within the external channels. The experiments by Lundström *et al.* confirmed that voids formed within the tows during global axial flow were indeed formed by entrapment close to the flow front and not by the release of volatile gases within the resin. They did not, however, examine the causes of void entrapment in their experiments.

Parnas *et al.* [37] argued that the observed void formation in the case of flow parallel to the fibers was due to heterogeneity of the tow structure caused by the presence of defects in the fibers and/or the tow. The result of these defects was to create regions of localized high, or low, permeability. They then went on to model the effects that these permeability variations would have on the flow within the fiber bundle using the standard analysis based on Darcy's law. Their analysis, however, did not provide a framework for predicting the proposed permeability variations. Their conclusion was that some kind of on-line monitoring of the infiltration process would be necessary to characterize the defect structure within a given preform fabric and, therefore, the expected void content within the finished part.

Both sets of analyses and the confirming experiments in [37] were based on idealized, two-dimensional geometries and neglected the effects of surface tension. The consequence of these assumptions is probably most significant in the axial flow case since the effect of capillary flows would be to significantly increase the amount of time to fully wet-out the fiber bundle as compared to the predictions in [52,53].

1.4 References

1. Composites-Busch & Co., 1996, World-Wide Web advertisement for its line of RTM hockey sticks, http:// pax.eunet.ch/Busch/Balance.E.html.

2. Sawyer, D. 1994, "Advanced Composite Armored Vehicle Platform", *Vehicle Survival Newsletter,* Fall, http://surviac.flight.wpafb.af.mil/curr_awar/veh_surv_newsletter/fall_94/fall_94.html

3. Fickie, K., 1994, "Manufacturing Simulation", *AHPCRC Bulletin*,**4**(3), http://www.arc.umn.edu/html/publications/bulletins/v4n3/v4n3.html

4. *Plastics Technology*, 1995, "SCRIMP Process Makes 15,000-lb Part in One Shot," *Plastics Technology*, March:11.

5. Michaeli, W., and Dyckhoff, J., 1993, "Improvement in the Surface Quality of Structural Components Produced by the RTM-Process," in *Proceedings of the International Conference on Advanced Materials 1993*, TMS, pp. 749-754.

6. Gabriele, M., 1995, "Chrysler's Viper Innovation," Plastics Technology, March:38-42.

7. Monks, R., 1993, "RTM: Hot Tools Drive The Push for Speed," Plastics Technology, March:40-44.

8. Parnas, R., Salem, A., Kendall, K., and Bruschke, M., 1994, "NIST Workshop on Manufacturing Polymer Composites by Liquid Molding", NISTIR 5373.

9. Bruschke, M., and Advani, S., 1990, "A Finite Element/Control Volume Approach to Mold Filling in Anisotropic Porous Media," *Polymer Composites* , **11**(6):398-405.

10. Coulter, J., and Güçeri, S., 1988, "Resin Impregnation During the Manufacturing of Composite Materials Subject to Prescribed Injection Rate," *Journal of Reinforced Plastics and Composites* , **7**:200-219.

11. Parnas, R., and Phelan, F., 1991, "The Effect of Heterogeneous Porous Media on Mold Filling in Resin Transfer Molding," *SAMPE Quarterly* , **22**(2):53-60.

12. Trevino, L., *et al.*, 1991, "Analysis of Injection Molding Molds With Preplaced Fiber Mats I: Permeability and Compressibility Measurements," *Polymer Composites* , **12**(1):20-29.

13. Adams, K., Miller, B., and Rebenfeld, L., 1986, "Forced In-Plane Flow of an Epoxy Resin in Fibrous Networks", *Polymer Engineering and Science*, **26**(20):1434-1441.

14. Salem, A., and Parnas, R., 1991, "The Unidirectional and Radial In-Flow of Fluids Through Woven Composite Reinforcements," in *Proceedings of the ASC 6th Technical Conference on Composite Materials* , October.

15. Hayward, J., and Harris, B., 1989, "Processing Factors Affecting the Quality of Resin Transfer Moulded Composites," *Plastics and Rubber Processing and Applications* , **11**:191-198.

16. Hayward, J., and Harris, B., 1990, "The effect of Vacuum Assistance in Resin Transfer Molding", *Composites Manufacturing*, **1**(3):161-166.

17. Sadiq, T., Parnas, R., Advani, S., 1992, "Experimental Investigation of Flow in Resin Transfer Molding," in *Proceedings of the 24th International SAMPE Technical Conference*, **24**:660-674.

18. Karbhari, V., *et al.*, 1991, "Effect of Preform Architecture and Injection Strategies on the Robustness of RTM Parts", in *Advanced Composite Materials: New Developments and Applications Conference Proceedings*, Detroit, MI, Sept. 30 - Oct. 3, pp. 91-103.

19. Patel, N., *et al.*, 1991, "Resin-Fiber Wetting and Bonding in Resin Transfer Molding and Structural Reaction Injection Molding," in *Proceedings of the SPE 49th Annual Technical Conference ANTEC '91*, pp. 1985-1990.

20. Walsh, S., 1990, "Artificial Intelligence: Its Application to Composite Processing," in *Proceedings of the 35th International SAMPE Symposium*, **35**:1280-1291.

21. Kranbuehl, D., Williamson, A., Loos, A., 1991, "Sensor-Model In-Situ Control of the RTM Composite Process, in *Proceedings of the 36th International SAMPE Symposium*, **36**:536-545.

22. Sheidegger, A., 1974, *The Physics of Flow Through Porous Media*, 3rd ed., University of Toronto Press, Toronto.

23. Bear, J., 1972, *Dynamics of Fluids in porous Media*, Ch. 11, Dover, New York.

24. Cruz, M., 1993, *A Parallel Monte-Carlo Partial-Differential-Equation Procedure for the Analysis of Muticomponent Random Media*, PhD. dissertation, Massachusetts Institute of Technology.

25. Ghaddar, C., 1995, *Parallel Analytico-Computational Methods for Multicomponent Media: Application to Thermal Composites and Porous-Media Flows*, PhD. dissertation, Massachusetts Institute of Technology.

26. Fisher, M., and Essam, J., 1961, "Some Cluster Size and Percolation Problems", *Journal of Mathematical Physics*, **2**(4):609-619.

27. Larson, R., Scriven, L., and Davis, H., 1977, "Percolation Theory of Residual Phases on Porous Media", *Nature*, **268**(4):409-413.

28. Heller, J., 1959, "Christiansen-Type Transparent Porous Media for Flow Studies", *Review of Scientific Instrumentation*, **30**:1056-57.

29. Müller, R., 1952, "Congress Lecture: Research in Analytical Instrumentation", *Analyst*, 77:557-563.

30. Glossop, N., Tsaw, R., Measures, R., and Tennyson, R., 1989, "Image-Enhanced Backlighting: A New Method of NDE for Translucent Composites", *Journal of Nondestructive Evaluation*, **8**(3):181-193.

31. Olson, J., Day, D., and Stoffer, J., 1992 "Fabrication and Mechanical Properties of an Optically Transparent Glass Fiber/Polymer Matrix Composite", *Journal of Composite Materials*, **26**(8):1181-1192.

32. Ng, K., Davis, H., and Scriven, L., 1978, "Visualization of Blob Mechanics in Flow Through Porous Media", *Chemical Engineering Science*, **33**:1009-1017.

33. Mahale, A., Prud'Homme, R., and Rebenfeld, L., 1992, "Quantitative Measurement of Voids Formed During Liquid Impregnation of Nonwoven Multifilament Glass Networks Using an Optical Visualization Technique," *Polymer Engineering and Science*, **32**(5):319-326.

34. Mahale, A., Prud'Homme, R., and Rebenfeld, L., 1993, "Characterization of Voids Formed During Liquid Impregnation of Non-Woven Multifilament Glass Networks as Related to Composite Processing", *Composites Manufacturing*, **4**(4):199-207.

35. Patel, N., and Lee, L., 1995, "Effects of Fiber Mat Architecture on Void Formation and Removal in Liquid Composite Molding", *Polymer Composites*, **16**(5):386-399.

36. Elmendorp, J., and During, F., 1990, "Dynamic Wetting Aspects of Melt Impregnation of Fiber Mats," in *Proceedings of the SPE 48th Annual Technical Conference ANTEC '90*, pp. 1361-1364.

37. Parnas, R., 1994, *et al.*, "The Interaction Between Micro- and Macroscopic Flow in RTM Preforms", *Composite Structures*, **27**:93-107.

38. Dullien,F., 1975, "New Network Permeability Model of Porous Media", AIChE Journal, **21**(2):299-307.

39. Payatakes, A., 1982, "Dynamics of Oil Ganglia During Immiscible Displacement in Water-Wet Porous Media", *Annual Review of Fluid Mechanics*, **14**:365-393.

40. Ruth, D., and R. Suman, R., 1992, "The Role of Microspic Cross Flow in Idealized Porous Media", *Transport in Porous Media*, **7**:103-125.

41. M. Blunt,, M., and King, P., 1991, "Relative Permeabilites from Two- and Three-Dimensional Pore-Scale Network Modelling", *Transport in Porous Media*, **6**:407-433.

42. Fracchia, C., Castro, J., and Tucker, III, C., 1989, "A Finite Element/Control Volume Simulation of Resin Transfer Mold Filling", in *Proceedings of The American Society for Composites Fourth Technical Conference*, Blacksburg, VA, Oct. 3-5, 1989, pp. 157-166.

43. Bruschke, M., and Advani, S., 1991, "RTM: Filling Simulation of Complex Three Dimensional Shell-Like Structures", *SAMPE Quarterly*, **23**(1):2-11.

44. Molina, G., Boero, G., and Smeriglio, P., 1994, "Resin Transfer Molding for Car Body Parts: An Integrated Approach Using CAE Methodology", *Journal of Reinforced Plastics and Composites*, **13**:681-697.

45. Dessenberger, R., and Tucker, III, C., 1995, "Thermal Dispersion in Resin Transfer Molding", *Polymer Composites*, **16**(6):495-506.

46. Sangani, A., and Yao, C., 1988, "Transport Processes in Random Arrays of Cylinders. II. Viscous Flow", *Phys. Fluids*, **31**(9):2435-2444.

47. Åström, T., Pipes, R., and Advani, S., 1992, "On Flow Through Aligned Fiber Beds and Its Application to Composites Processing, *Journal of Composite Materials*, **26**(9):1351-1373.

48. Cai, Z., and Gutowski, T., 1992, "The 3-D Deformation Behavior of a Lubricated Fiber Bundle," *Journal of Composite Materials*, **26**(8):1207-1237.

49. Gutowski, T., and Dillon, G., 1992, "The Elastic Deformation of Lubricated Carbon Fiber Bundles: Comparison of Theory and Experiments," *Journal of Composite Materials*, **26**(16):2330-2347.

50. Kim, Y., McCarthy, S., and Fanucci, J., 1990, "Compressibility and Relaxation of Fiber Reinforcements During Composite Processing", in *Proceedings of the SPE 48th Annual Technical Conference ANTEC '90*, pp. 1252-1256.

51. Parnas, R., and Phelan, F., 1991, "The Effect of Heterogeneous Porous Media on Mold Filling in Resin Transfer Molding," *SAMPE Quarterly* , **22**(2):53-60.

52. Chan, A., and Morgan, R., 1992, "Modeling Preform Impregnation and Void Formation in Resin Transfer Molding of Unidirectional Composites", *SAMPE Quarterly*, **23**:48-52.

53. Chan, A., and Morgan, R., 1993, "A Model on Formation of Rod-Like Voids in Fiber Tows", *SAMPE Quarterly*, July 49-53.

54. Lunström, T., Gebart, B., and Lundemo, C., 1993, "Void Fromation in RTM", *Journal of Reinforced Plastics and Composites*, **12**:1339-1349.

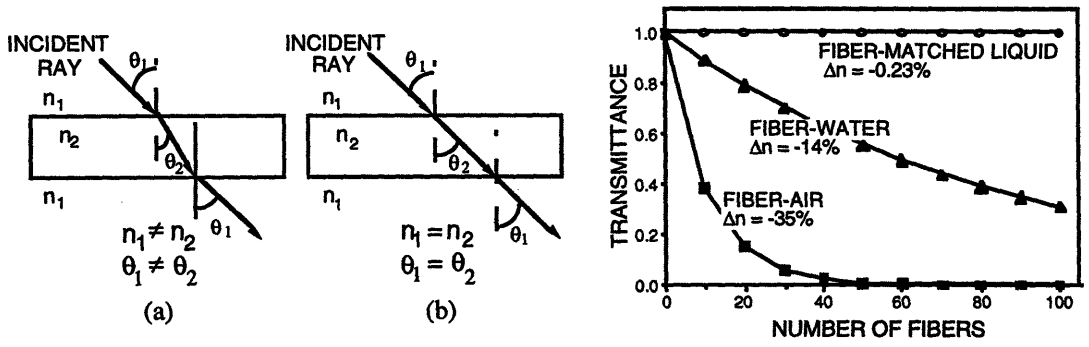
PART I:

MATCHED REFRACTIVE INDEX FLOW VISUALIZATION

2 EXPERIMENTAL PROCEDURE: Matched Refractive Index Flow Visualization

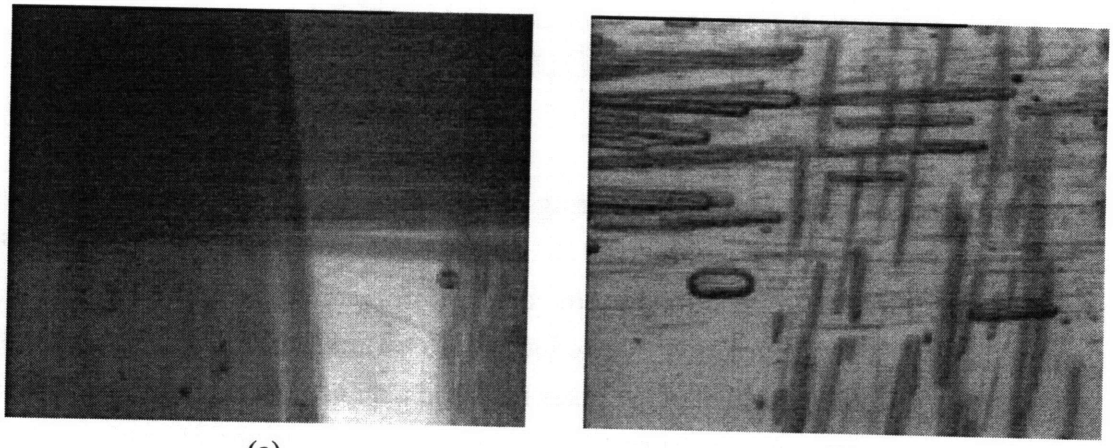
An approximate analysis of the scattering of transmitted light by dry and wet fiber preforms is discussed in Sections 2.1.1 and 2.1.2, respectively. Photometric measurements of the light transmitted by the dry preform can be used to estimate local variations in fiber volume fraction. Introduction of a fluid within the preform with refractive index approximating that of the fibers results in a dramatic reduction of the optical scatter caused by the fibers within the preform. This enables observation of wetting defects deep within the preform. Matching of the refractive indices of the fibers and impregnating fluid extends the visualization technique of making observations through transparent molds. Wetted portions of the preform then appear transparent. Dry portions remain translucent. Figures 2.1 and 2.2 illustrate the effect of refractive index mismatch on preform transmittance. The level of fluid-fiber refractive index match can be enhanced by using a monochromatic illumination source, thereby increasing image contrast between wet and dry regions of the preform and reducing noise within the wetted portion of the image.

Photometric measurements of the light transmitted by the partially wetted fiber preform can be used to estimate saturation levels during the impregnation process. As discussed in Section 2.1.3, photometric analysis and knowledge of the scattering characteristics of the dry fibers can be used to reconstruct some three-dimensional information on flow front and defect location from a single image. Some practical issues related to selection of model fluids and light sources are discussed in Section 2.2.



(a) (b)

Figure 2.1 Effect of refractive index mismatch on transmittance: (a) schematic of the physical mechanism, (b) transmittance of selected fiber-fluid combinations.



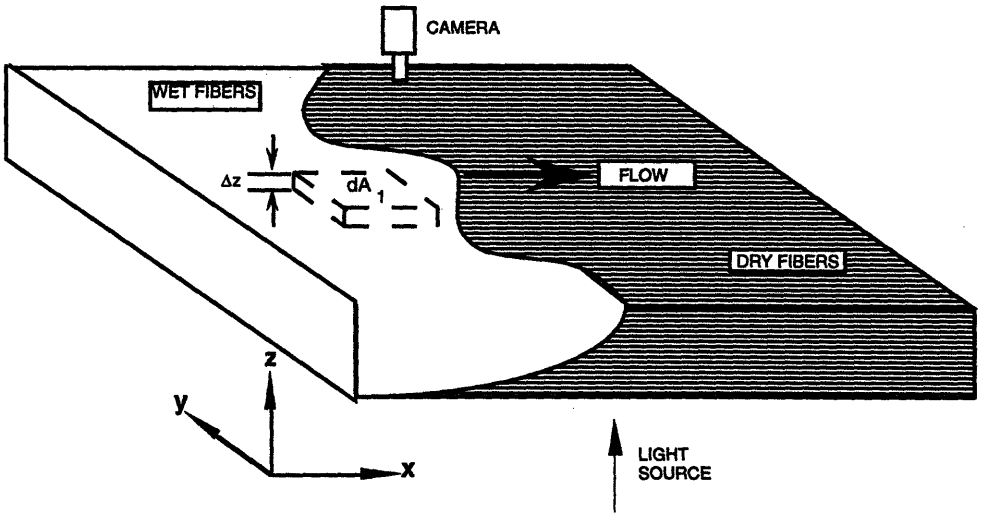
(a) (b)

Figure 2.2 Significance of refractive index mismatch in observations at the tow and fiber scales in a plain woven fabric. (a) Glass fibers saturated with water, $\Delta n = -14.4\%$, (b) Glass fibers saturated with index-matched immersion liquid, $\Delta n \sim 0$.

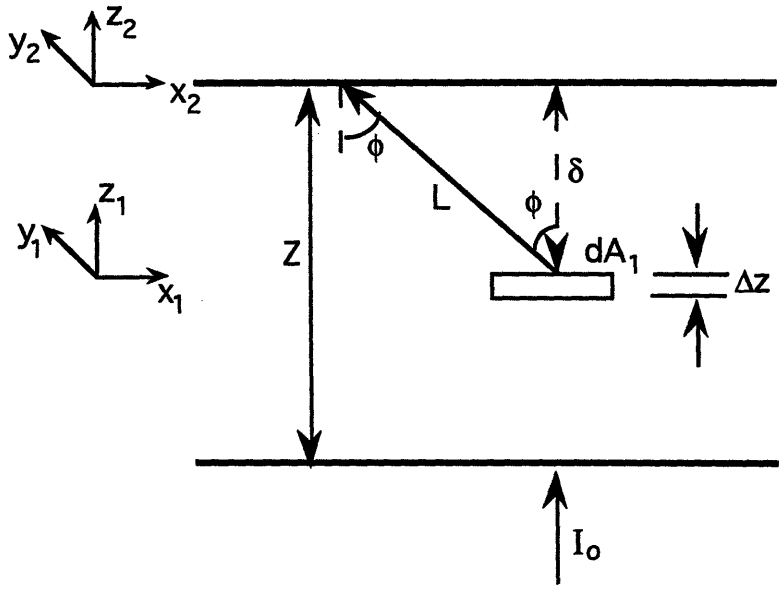
2.1 Image Analysis

The goals of the image analysis were two-fold: characterization of the dry preform and then characterization of the flow within the preform during impregnation. A schematic representation of the geometry and nomenclature for image analysis are shown in Figure 2.3. Analysis of the dry preform was performed to determine the fiber-volume fraction within the preform. Knowledge of the local fiber-volume fraction enabled prediction of expected flow within the preform and identification of potential trouble spots during impregnation. Characterization of the flow during experiments consisted of identifying the location of the fluid within the preform at any given time (i.e., preform saturation as a function of position and time, S(x,y,t)) and the determination of the depth of selected flow features, such as the flow front and voids.

The analytical approach was based on three basic experimental observations. First, as demonstrated in Figures 2.4(a) and 2.4(b), measurements of the transmittance of fiber preforms as a function of depth showed that the transmitted intensity decayed exponentially with increasing depth. Furthermore, as shown in Figure 2.5, this observed attenuation in the transmitted intensity was fairly insensitive to in-plane orientation of the plies within multi-layer composites. Finally, as summarized in Table A.1, microscopic measurements at the fiber scale confirmed that the observed attenuation of the transmitted light was due primarily to interfacial scattering.

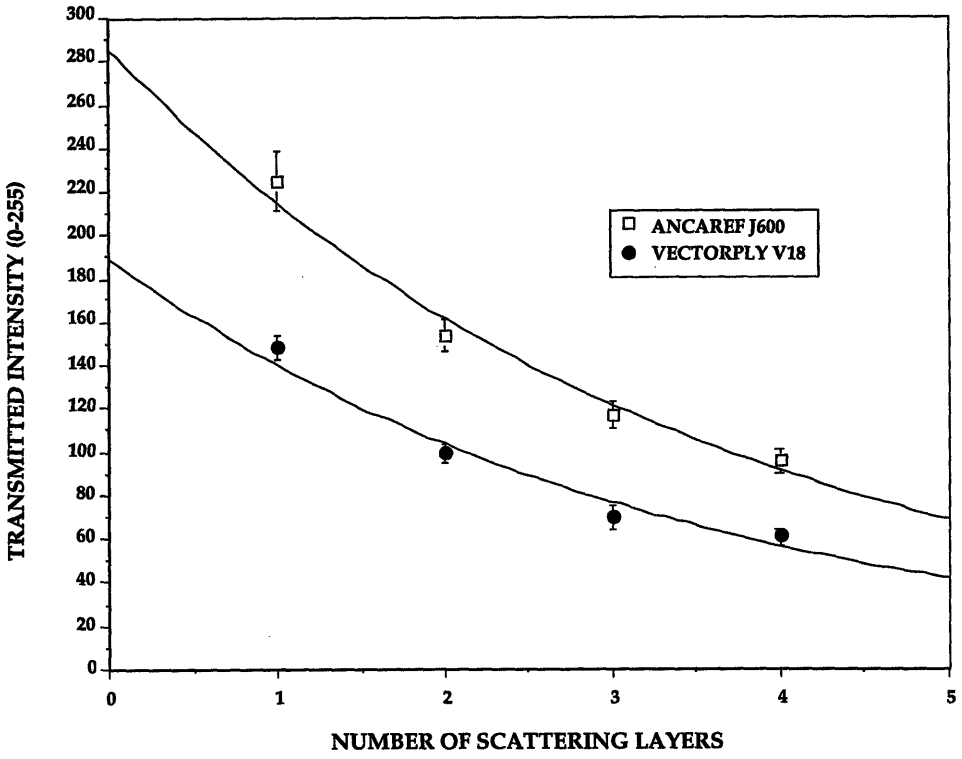


(a)

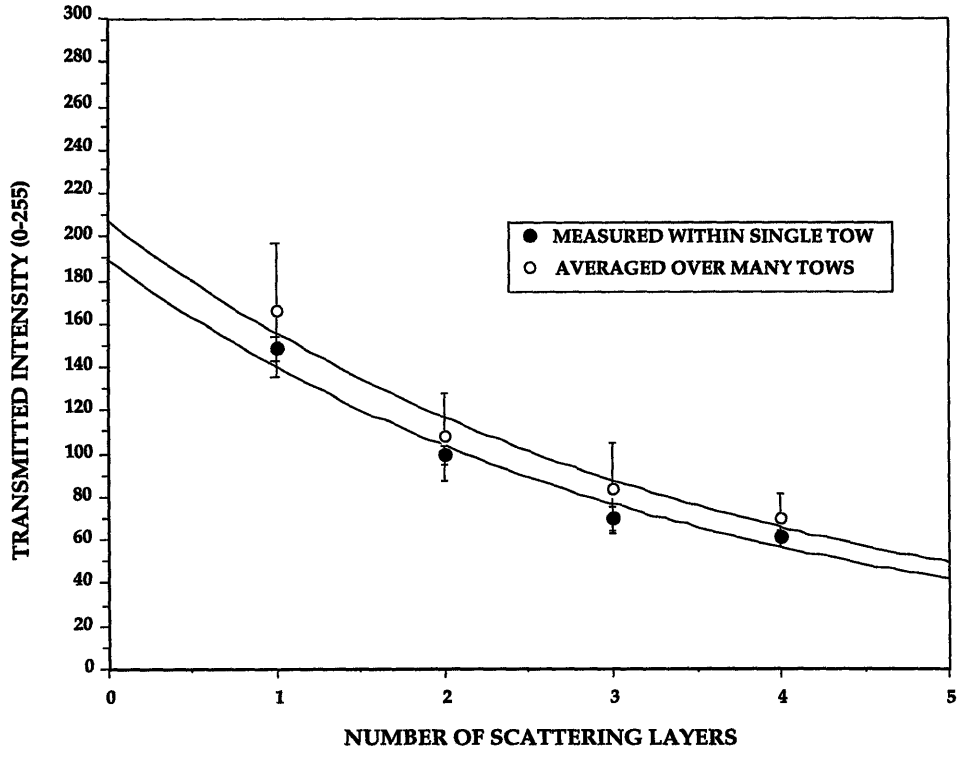


(b)

Figure 2.3 Nomenclature (a) Schematic representation of flow geometry and (b) nomenclature for image analysis.



(a) Transmission characteristics based on tow-scale measurements.



(b) Effects of changing sampling area.

Figure 2.4 Transmission characteristics of two unidirectional fabrics.

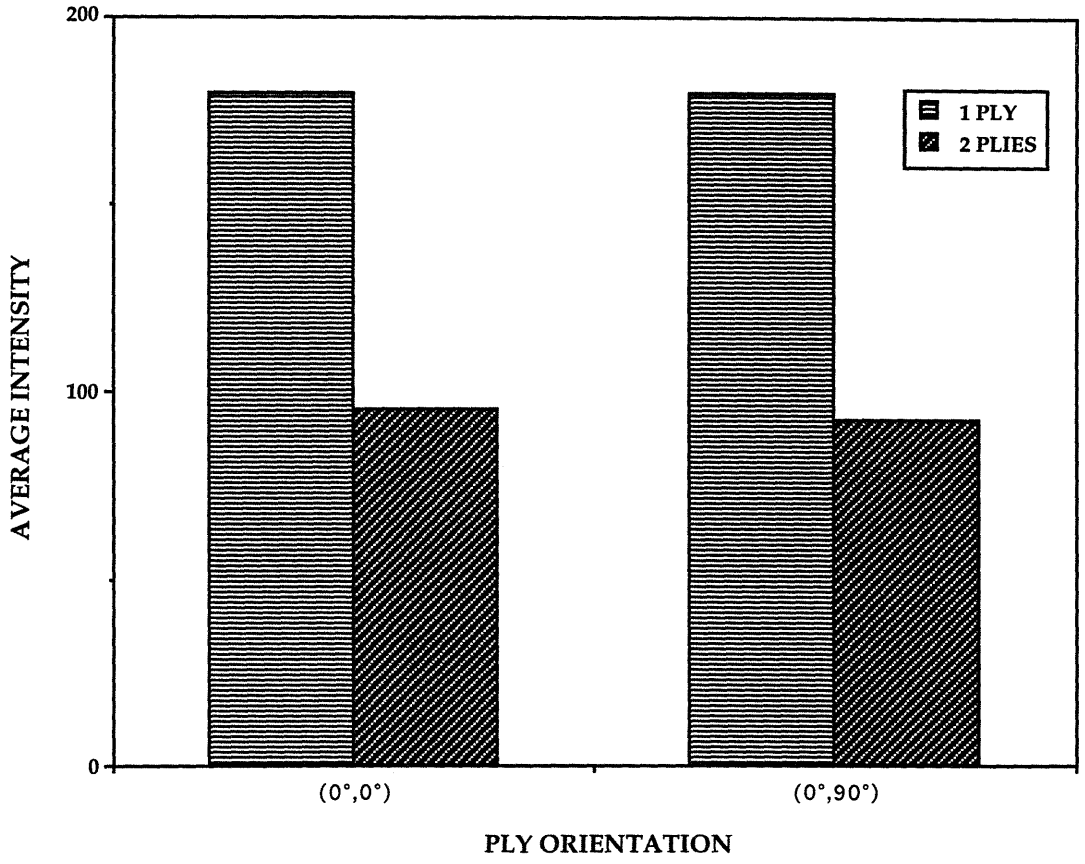


Figure 2.5 Effect of in-plane ply orientation on light transmission.

2.1.1 Characterization of Dry Preform

Measurements of the transmitted light intensities for various sample thicknesses are shown in Figure 2.4. The specimens were commercial reinforcement fabrics. Physical properties of the specimens are summarized in Appendix A. The ratio of the transmitted, $I(Z)$, to incident, I_o, intensities--the transmittance, $T(Z)$--is given by the expression

$$T(Z) = \frac{I(Z)}{I_o} = e^{-\beta Z} \tag{2.1}$$

where Z is the sample thickness. The extinction coefficient, β, is defined as an average value characteristic of the medium corresponding to the inverse of a characteristic extinction length [1] at which

$$\frac{I(1/\beta)}{I_o} = e^{-1} \tag{2.2}$$

In order to obtain a valid average value for the extinction coefficient, measurements were sampled over discrete areas as opposed to individual points within the image. Figure 2.4(a) illustrates transmitted intensity measurements taken over areas completely contained within the tows making up the respective fabrics. The error bars in Figure 2.4 correspond to the standard deviations in the intensity measurements within the sample areas at each sample thickness. The standard deviations of measurements taken within single tows were similar for both fabrics. Figure 2.4(b) shows the effect of increasing the sampling area to incorporate multiple tows in the VECTORPLY fabric. Note that incorporation of the large inter-tow gaps in the sampling area increased the standard deviation of the measured intensity values, but had very little effect on the mean values.

The extinction coefficient defined in Eq. 2 corresponds to an average coefficient that incorporates the effects of both the preform's glass fibers and the inter-fiber spaces within the mat. Solving Eq. 1 for the average mat extinction coefficient, $\beta(x,y)$, and noting that

$$Z = Z_f(x,y) + Z_a(x,y) \tag{2.3}$$

yields

$$\beta = \frac{\beta_f Z_f(x,y) + \beta_a Z_a(x,y)}{Z} \tag{2.4}$$

where $Z_f(x,y)$ and $Z_a(x,y)$ are equal to the total distance in the fibers and the inter-fiber gaps, respectively, through which the light propagates, β_f is the extinction coefficient of an individual fiber and β_a is the extinction coefficient of an inter-fiber air gap. Because the transmittance of the inter-fiber air gaps is 1 (i.e., the extinction coefficient of each inter-fiber air gap, β_a, is equal to zero), the effect of these gaps is to decrease the overall average extinction coefficient, since

$$\beta = \frac{\beta_f Z_f(x,y)}{Z} \tag{2.5}$$

A first-order approximation for β_f can be made by assuming that the attenuation of light by an individual fiber is due primarily to scattering at its fiber-air interfaces. Assuming that the light incident on a given fiber is approximately normal to the fiber-air interface (this is equivalent to assuming that the fiber is a slab and that the incident light is normal to the slab surface), the ratio of transmitted to incident flux at each fiber-air interface, T_{fa}, is given by the relation [1]

$$T_{fa} = \frac{4n_f n_a}{(n_f + n_a)^2} \approx \frac{4n_f}{(n_f + 1)^2} \tag{2.6}$$

where n_f and n_a are the refractive indices of the fiber and air, respectively, and n_a is approximately equal to one. Since the light must cross a total of two fiber-air interfaces (i.e., one each upon entering and exiting the fiber) the total transmittance of the fiber, T_f, is given by

$$T_f = T_{fa}^2 \tag{2.7}$$

The extinction coefficient, β_f, can be determined by substituting Eqs 2.6 and 2.7 into Eq. 1, yielding

$$\beta_f = -\frac{2}{d_f} \ln\left[\frac{4n_f}{(n_f + 1)^2}\right] \tag{2.8}$$

where it is assumed that d_f, the fiber diameter, corresponds to the optical path length, Z, in Eq. 1.

The validity of Eqs 2.6 and 2.7 for the fibers used in this study was checked by measuring the transmittance of individual fibers under a microscope and substituting into Eqs 2.7 and 2.6 and solving for the fiber refractive index, n_f. This value was then compared with the value determined using the immersion technique. The results are summarized in Table A.1. The discrepancy in fiber refractive index values using the two approaches was -0.23%. Consequently, it was assumed that the observed attenuation of the transmitted light occurred primarily at the fiber-air interfaces.

Applying Eq. 2.7 to an assembly of many fibers (i.e., a tow or a fabric sample) leads to an estimate of the total number of fiber-air interfaces (and, therefore, a first-order estimate of the fiber-volume fraction) within the dry preform. Consider a dry preform of thickness, Z, made up of N_f fibers with diameter, d_f. The intensity of the light transmitted at position (x,y) on the top surface would be attenuated at each of the fiber-air interfaces throughout the thickness of the preform. Assuming that the fabric sample is replaced by a stack of N_f slabs of thickness d_f, the total transmittance, $T(x,y)$, of the preform would be given, to a first-order, by the expression

$$T(x,y) = \frac{I(x,y)}{I_o} = (T_{fa})^{N_{fa}} \tag{2.9}$$

where N_{fa} is the total number of fiber-air interfaces throughout the specimen's thickness at (x,y) and is equal to $2N_f$. Solving Eq. 2.9 for the number of fiber-air interfaces yields

$$N_{fa} = \frac{\ln T(x,y)}{\ln T_{fa}} \tag{2.10}$$

In the case that the resolution of the transmission measurements is coarser than a single fiber diameter, one can use the number of fiber-air interfaces calculated from 2.10 to approximate the thickness-averaged fiber-volume fraction, $V_f(x,y)$. Writing $V_f(x,y)$ as the ratio of the actual distance traveled within the fibers, $Z_f(x,y)$, to the thickness of the specimen, Z, yields

$$V_f(x,y) \approx \frac{d_f \cdot \frac{1}{2} N_{fa}(x,y)}{Z} \tag{2.11}$$

Substituting Eq. 2.10 into Eq. 2.11 and noting that

$$\frac{V_f(x,y)}{V_f^o} = \frac{N_{fa}(x,y)}{N_{fa}(x_o,y_o)} \tag{2.12}$$

where V_f^o is the thickness-averaged fiber-volume fraction at reference point (x_o,y_o) on the top surface of the specimen, and that

$$\frac{I(x,y)}{I_o} = \frac{I(x,y)}{I(x_o,y_o)} \cdot \frac{I(x_o,y_o)}{I_o} \tag{2.13}$$

yields

$$\frac{V_f(x,y)}{V_f^o} = 1 + \frac{\ln\left[\frac{I(x,y)}{I(x_o,y_o)}\right]}{\ln\left[\frac{I(x_o,y_o)}{I_o}\right]} \tag{2.14}$$

Examination of Eq. 2.14 indicates that the thickness-averaged fiber-volume fraction at any point on the preform relative to an arbitrary reference point (x_o,y_o) can be determined experimentally by measuring the absolute transmittance only at (x_o,y_o) and then by measuring the relative intensity, $I(x,y)/I(x_o,y_o)$, elsewhere within the image. Note that, experimentally, determining the relative intensities within an image is much easier than determining absolute transmittances.

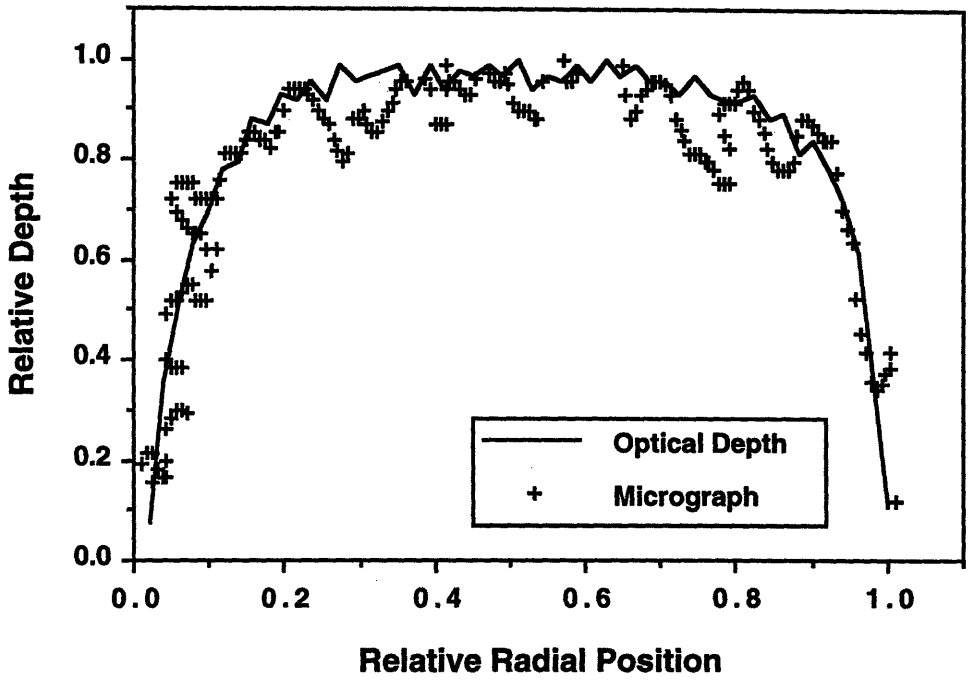


Figure 2.6 Comparison of measured optical and micrograph-based cross-section profiles.

Figure 2.6 shows the relative cross-section depth of a typical tow of the VECTORPLY fabric determined using Eq. 2.14. Equivalent data obtained by directly measuring a typical tow cross-section are plotted on the same figure for comparison. As can be seen, the results obtained using the two approaches are in qualitative agreement. The photometric method has the advantages of being nondestructive and rapid. Consequently, large portions of fabric can be analyzed with relatively little effort before an experiment and potential problem areas during the impregnation process can be identified ahead of time.

2.1.2 Saturation Measurement

Optical measurement of the thickness-averaged preform saturation as a function of time and position, $S(x,y,t)$, requires determination of the number of scattering interfaces present at time t. Consider the general case of infiltration where the wetting liquid, displaced inter-fiber air and preform fibers have refractive indices n_l, n_a and n_f, respectively. Assume that the preform can be approximated by the slab model used in the analysis of dry fibers discussed in Section 2.1.1. In this case, the number of scattering interfaces includes not only N_{fa} fiber-air and N_{la} liquid-air interfaces, but also N_{fl} fiber-liquid interfaces. Extending the result in Eq. 2.9 to incorporate the three types of interfaces present in the general infiltration case, yields

$$T(x,y,t) = \left(T_{fa}\right)^{N_{fa}} \cdot \left(T_{la}\right)^{N_{la}} \cdot \left(T_{fl}\right)^{N_{fl}} \tag{2.15}$$

for the transmittance at (x,y) where T_{fa} is defined in Eq. 2.6 and T_{la}, the transmittance at each liquid-air interface, is given by

$$T_{la} = \frac{4n_l n_a}{\left(n_l + n_a\right)^2} \approx \frac{4n_l}{\left(n_l + 1\right)^2} \tag{2.16a}$$

and T_{fl}, the transmittance at each fiber-liquid interface, is given by

$$T_{fl} = \frac{4n_f n_l}{\left(n_f + n_l\right)^2} \tag{2.16b}$$

Note that N_{fa}, N_{la} and N_{fl} are all functions of the form $N(x,y,t)$ and that, in general,

$$N_{fa} + N_{la} + N_{fl} \neq \left(N_{fa}\right)^d \tag{2.17}$$

where N_{fa}^d is the total number of fiber-air interfaces within the dry preform at (x,y), determined by applying Eq. 2.10. The general validity of Eq. 2.17 is demonstrated by noting the increase in optical interfaces caused by the introduction of air bubbles into the inter-fiber spaces. Since it is usually not possible to separate the effects of the fiber-liquid scattering from the fiber-air and liquid-air scattering, optical measurement of the preform saturation in the general case can not be accomplished without assuming some kind of empirical relationship between transmitted intensity and saturation.

If, on the other hand, $n_f \approx n_l$ then $T_{fl} \rightarrow 1$ and $T_{la} \rightarrow T_{fa}$, so that Eq. 2.15 becomes

$$T(x,y,t) \approx \left(T_{fa}\right)^{\left(N_{fa}+N_{la}\right)} \tag{2.18}$$

Solving for the number of scattering interfaces, $N_{fa} + N_{la}$, and dividing by the original number of fiber-air interfaces in the dry preform, N_{fa}^d, yields the fraction of original scattering interfaces that remain unwetted,

$$\frac{N_{fa} + N_{la}}{\left(N_{fa}\right)^d} = \frac{1}{\left(N_{fa}\right)^d} \cdot \frac{\ln\left[T(x,y,t)\right]}{\ln\left(T_{fa}\right)} \tag{2.19}$$

Consequently, the saturation, $S(x,y,t)$ is

$$S(x,y,t) = 1 - \frac{1}{\left(N_{fa}\right)^{d}} \cdot \frac{\ln\left[T(x,y,t)\right]}{\ln\left(T_{fa}\right)} \tag{2.20}$$

Substituting Eqs 2.9 and 2.10 into Eq. 2.20 yields

$$S(x,y,t) = 1 - \frac{\ln\left[\frac{I(x,y,t)}{I_o}\right]}{\ln\left[\frac{I_d(x,y)}{I_o}\right]} \tag{2.21}$$

where $I_d(x,y)$ is the transmitted intensity in the dry fabric and I_o is the incident intensity. Assuming a perfect refractive index match ($\Delta n = 0$) a fully saturated specimen will have $S(x,y,t) = 1$ and, therefore, $I(x,y,t) = I_o$.

2.1.3 Depth Determination of Selected Features

Depth information within images of the back-illuminated preform can be obtained by accounting for two complimentary effects. One effect is defocusing that occurs because an object moves either behind or in front of the "object" plane defined by the geometric optics lens formula [1],

$$\frac{1}{f} = \frac{1}{l_o} + \frac{1}{l_i} \tag{2.22}$$

where f is the optical system's focal length and l_o and l_i are the object and image plane distances from the focal plane, respectively. At relatively low magnifications (i.e., large depth of focus) and for images of relatively thin parts this is not a major factor. The second effect is the blurring of an imaged object caused by the presence of dry fibers between the object and the observation plane. This effect is important since the amount of blurring caused by the dry fibers can be correlated with the object's depth below the surface of the fiber reinforcement (i.e., the number of scattering fibers). As shown in Figure 2.7, the effects of scattering far outweigh the effects of depth of focus at low magnification.

In order to avoid the problem of trying to separate the effects of defocusing and scattering, the total effect caused by both phenomena was measured. This was done, experimentally, by focusing on the top surface of the fabric sample for all experiments. Consequently, the observed degradation of edge definition within collected images was due to the total effect caused by both phenomena.

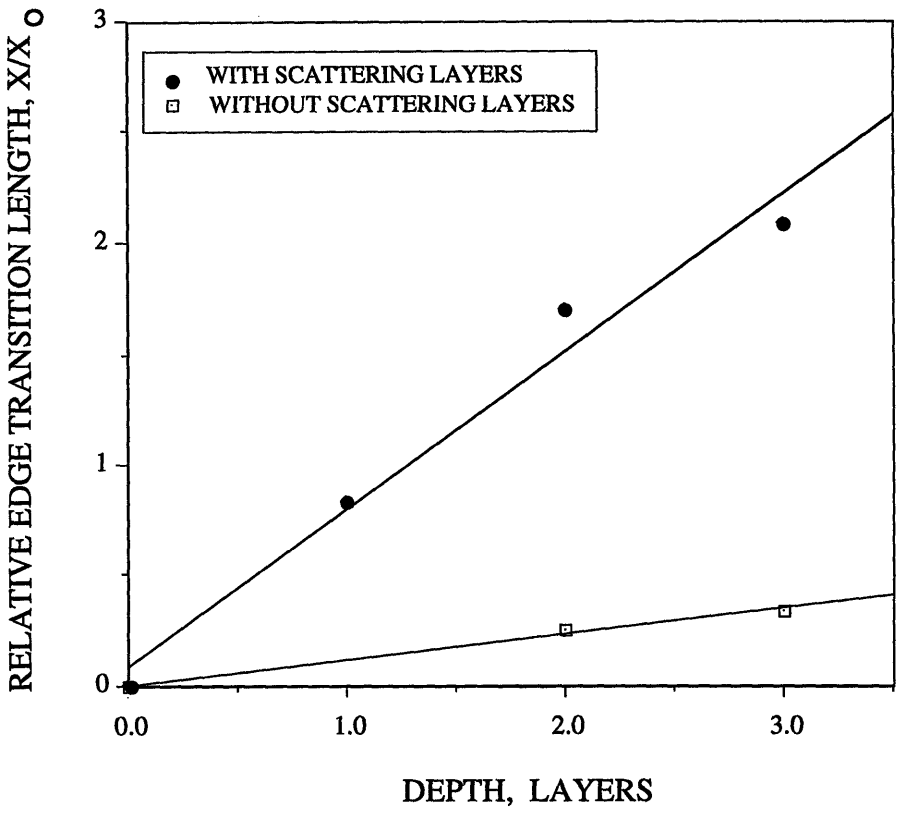


Figure 2.7 Relative contributions of scattering by dry fibers and depth of focus of optics.

As shown in Figure 2.8, the sharpness of boundaries between the light and dark regions within the image decreased with the introduction of additional scattering plies between the object, in this case the edge of a hole placed within a sandwich of multiple layers of dry fiber mats, and the observer. Plotting the image intensity as a function of position along a line crossing the observed edge enabled us to quantify this degradation of image sharpness. The edge transition length defined in Figure 2.8 corresponds to the number of pixels necessary for the transition from light to dark within the image. The terms "light" and "dark" were quantified by using the local, average pixel intensities in the neighborhood of the edge. The number of pixels necessary for the edge transition within the image was determined by counting the number of pixels necessary for the image intensity to fall from the local, average "light" value to the local, average "dark" value. A graphical representation of the process is outlined in Figure 2.8. Note that it is necessary to define some tolerance range about the local averages in order to define the beginning and the end of the imaged edge transition. We used the local standard deviation of the intensity to define this tolerance.

As shown in Figure 2.8(a), this transition occurred very quickly for an edge located on the top surface (i.e., no obstructing fibers to scatter the light from the boundary). As shown in Figures 2.8(b) and 2.8(c), addition of scattering fibers between the object and the camera, increased the number of pixels over which this transition occurred.

The scattering phenomena responsible for the loss of edge definition within the image can be described using a simple model. Consider the specimen, a single mat of fabric, of thickness Z shown in Figure 2.3. The top surface, surface 1, of a wetted region of the specimen of thickness Δz is located at a depth δ below the top surface of the sample. The sample is illuminated from below by a source of intensity I_o and observations are made from above. The amount of light energy, E, per unit area of surface 2, radiated from surface 1 is given by

$$I_2(x_2,y_2)=\frac{dE_{1,2}}{dA_2}=\int I_1(x_1,y_1)\frac{\cos^2\phi}{L^2}e^{-\beta L}dA_1 \tag{2.23}$$

where the extinction coefficient, β, is as defined in Eq. 2.2 and is a function of the path, L, traveled by the light. For a uniform intensity source at surface 1, one can simplify evaluation of the integral in Eq. 2.23 by considering the ratio

$$\frac{I_2(x_2,y_2)}{I_1}=\int\frac{\cos^2\phi}{L^2}e^{-\beta L}dA_1 \tag{2.24}$$

which depends only on geometry.

The relations given in Eqs 2.23 and 2.24 assume that the specimen attenuates radiation exponentially [1]

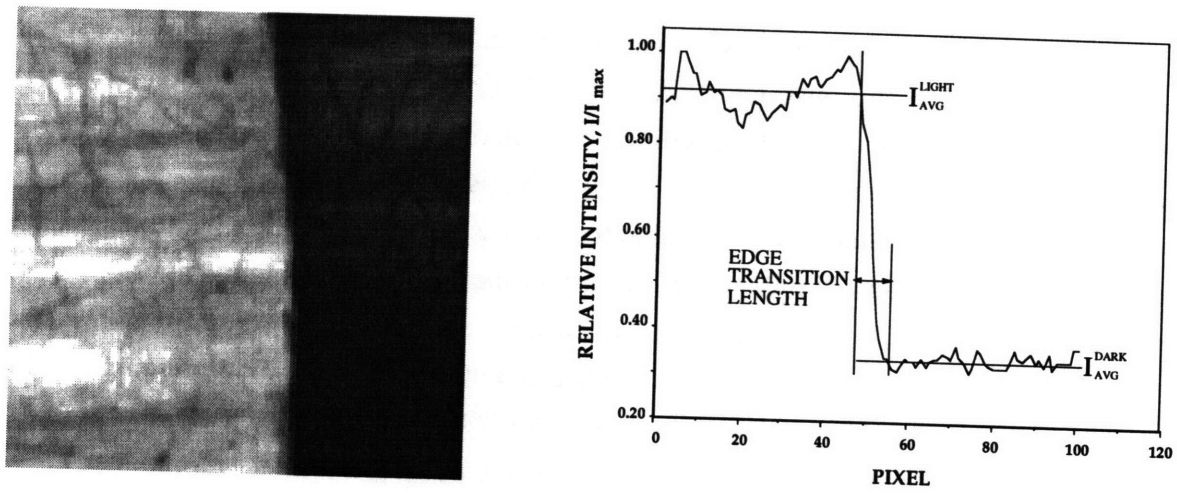
$$I(x,y,z)=I_o e^{-\beta z} \tag{2.25}$$

and that the observed wet region acts as a diffuse (i.e., Lambertian) source. Such a source radiates

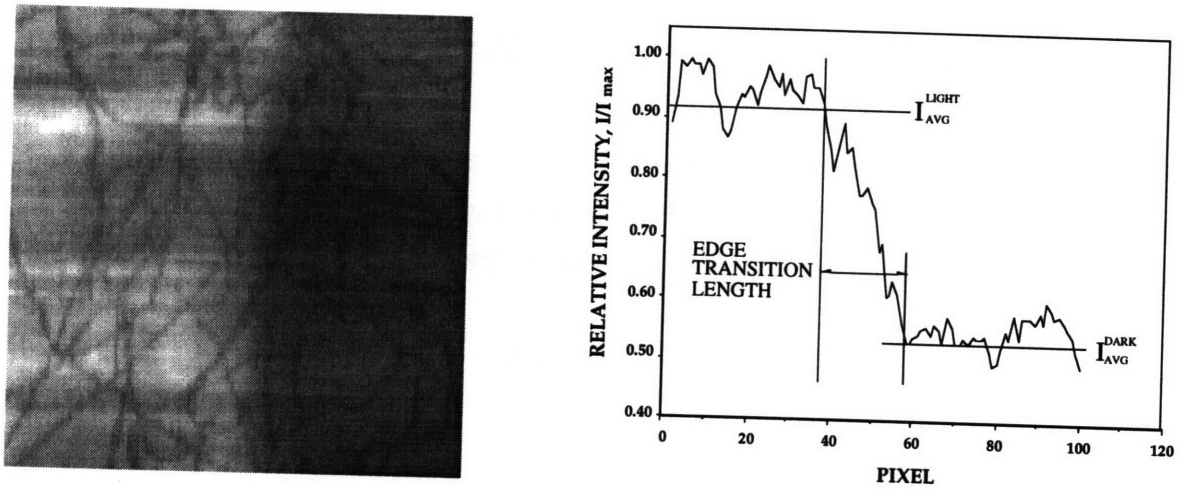
$$I(\phi)=I_1\cos\phi \tag{2.26}$$

energy per unit area in the ϕ-direction [2].

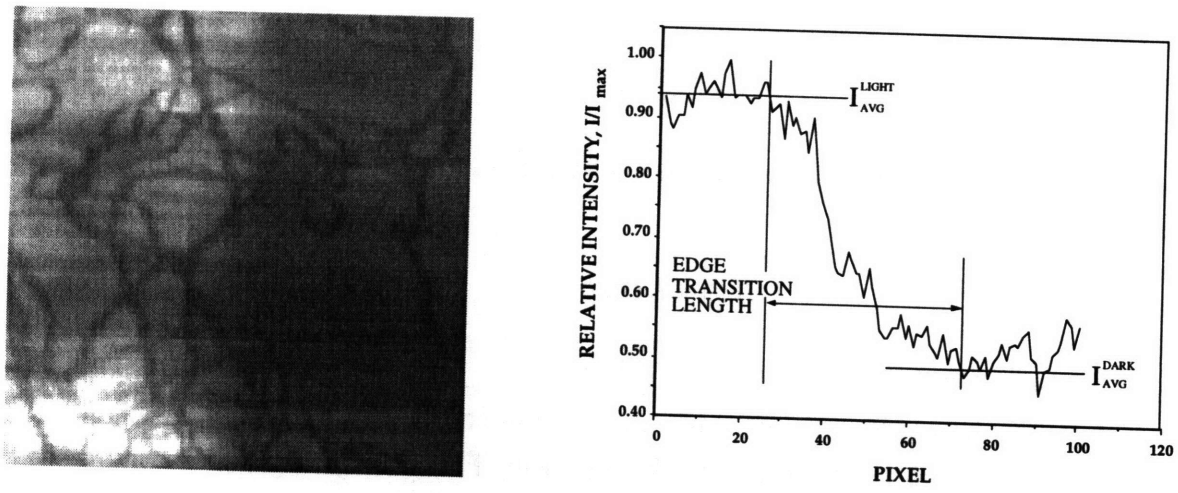
A first-order approximation of the measured intensity distribution in the vicinity of a wetting boundary at a depth δ can be obtained using Eq. 2.23 for a light source at the same depth. It can be shown that the calculated intensity distribution expressed relative to the surface intensity at a dry portion of the preform is given by



(a) Edge located at surface.



(b) Edge located one layer below surface.



(c) Edge located two layers below surface.

Figure 2.8 Degradation of edge definition as a function of depth within preform.

$$\frac{I_2(x_2,y_2)}{I_{dry}} = \frac{I_2(x_2,y_2)}{I_1} e^{\beta\delta} \tag{2.27}$$

In Eq. 2.27 the measured relative brightening of wetted regions of the image are related to the calculated relative intensity given in Eq. 2.23. Evaluation of Eqs 2.23 and 2.27 shows that, due to the scattering of light by the fabric, the observed transition between wet and dry regions below the preform surface will not occur abruptly but will, instead, occur over some finite distance determined by the product $\beta\delta$, the depth relative to the characteristic extinction length of the fabric below the reinforcement surface.

Figure 2.9 shows the comparison between the calculated and measured relationship between the observed length of edge transition and the depth of the observed edge for a large source modeled with Eq. 2.23 as a semi-infinite plane. Definition of the calculated transition length was set as the distance from the boundary at which the image intensity was equal to e^{-2} of its maximum value. The measured transition length was defined as in Figure 2.3. The assumption of a semi-infinite source in Eq. 2.23 represents observation of

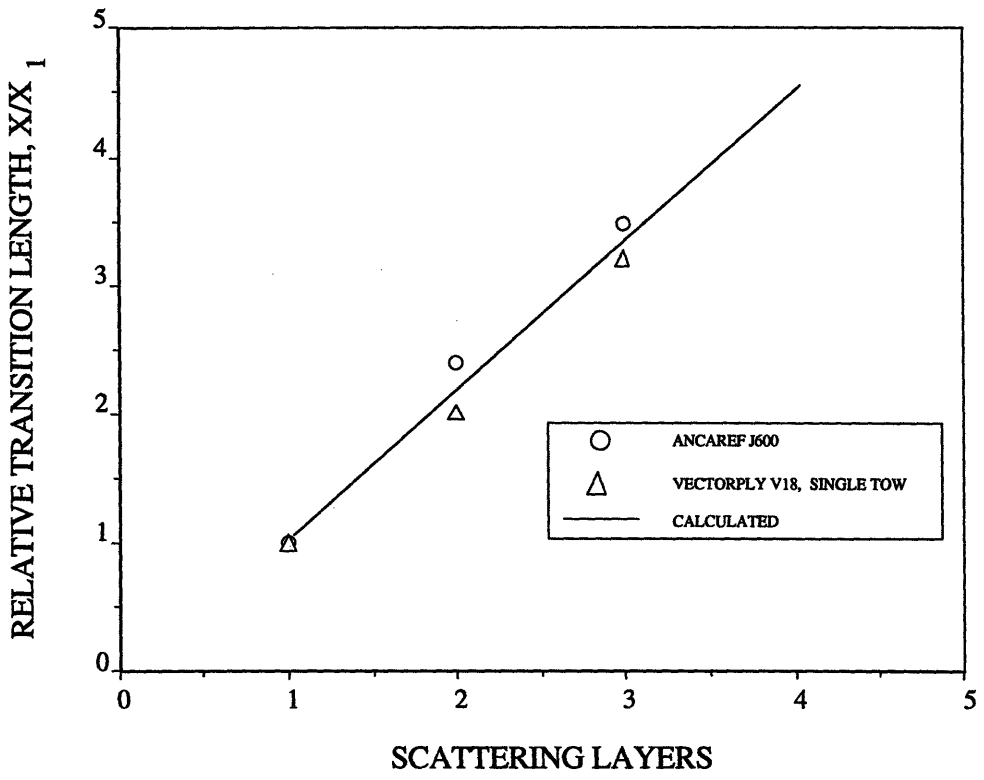


Figure 2.9 Length of observed edge transition vs. number of scattering layers.

a large subsurface flow front. Note that this assumption is reasonable since the integral in Eq. 2.23 converges rapidly to zero for large values of $\beta L(x)$.

Practical application of the technique requires determination of the extinction coefficient β and calibration of the imaging system for measurement of the edge transition lengths at the desired magnification. Note that the resolution of the imaging system is the limiting factor in deciding the minimum image magnification. In addition, Eq. 2.23 was derived without considering the depth of focus of the imaging optics and needs to be corrected for use at high magnifications.

2.2 On Selection of Test Fluids and Illumination Source

Selection of optically compatible modeling fluids was based on three sets of parameters: the wavelength dependence of the fiber and liquid refractive indices (or dispersion), the wavelength dependence of the illumination source, and the wavelength dependence of the sensor CCD array. Initial selection of the modeling fluids was based on the value of the fiber index of refraction at a selected test wavelength (determined by directly testing the match between the fibers and modeling fluids using a standard immersion technique).

The index of refraction of a material is not a constant but a function of a number of variables, chief of which are temperature and the wavelength of incident light. All of the experiments performed in this thesis were done at room temeprature, therefore, the variation of refractive index with temperature was not considered. On the other hand, the wavelength dependence of the refractive index, or dispersion, was an issue that needed to be addressed because the available monochromatic source (a He-Ne laser operating at its fundamental wavelength of 632.8 nm) emitted at relatively low-power and produced speckle within the acquired images. For convenience, a conventional polychromatic source such as an incadescent lamp is desirable.

For materials that are transparent in the visible range of the electromagnetic spectrum, the refractive index $n(\lambda)$ is a monotonically decreasing function of (increasing) wavelength, λ. The wavelength dependence of the refractive index can be approximated using a first-order Taylor series expansion around the test wavelength λ_o

$$n(\lambda) \approx n(\lambda_o) + \frac{dn}{d\lambda}(\lambda - \lambda_o) \tag{2.28}$$

As discussed in [3], for typical solids one can approximate $dn/d\lambda$ with

$$\left.\frac{dn}{d\lambda}\right|_{solid} \approx \frac{\Delta n_{solid}}{\Delta\lambda} = -C \times 10^{-4}/nm \tag{2.29}$$

where $0.5 \leq C \leq 1$. For typical liquids, one can write

$$\frac{\Delta n_{liquid}}{\Delta \lambda} \approx 2\left(\frac{\Delta n_{solid}}{\Delta \lambda}\right) \tag{2.30}$$

Consequently, selection of a liquid that matches the fiber refractive index at the test wavelength will result in a fairly good match if the liquid also has low dispersion (i.e., $\Delta n/\Delta\lambda \rightarrow 0$) since the fibers will display even less dispersion than the liquid. An estimate of the wavelength dependence of the liquid, $n(\lambda)$, can be made using a power series approximation, the Cauchy equation [1],

$$n(\lambda) = \sum_{i=0}^{\infty} \frac{a_i}{\lambda^{2i}} \tag{2.31}$$

where the a_i are empirically determined constants that are available for a broad range of liquids (including the commercially prepared liquids used in this thesis). Typically, only the first three terms of the series are used. Table B.1 lists the Cauchy equation coefficients for the commercially prepared liquids (Cargille #5095 and #1057) and C_6H_5Br. Figure 2.10(a) shows the resulting dispersion behavior over the visible spectrum ($400 \text{ nm} \leq \lambda \leq 770 \text{ nm}$) of the three liquids and an estimate of the glass fiber dispersion based on Eqs 2.28 and 2.29 with $C = 1$. Figure 2.10(b) is a plot of the magnitude of the resulting refractive index mismatch given by

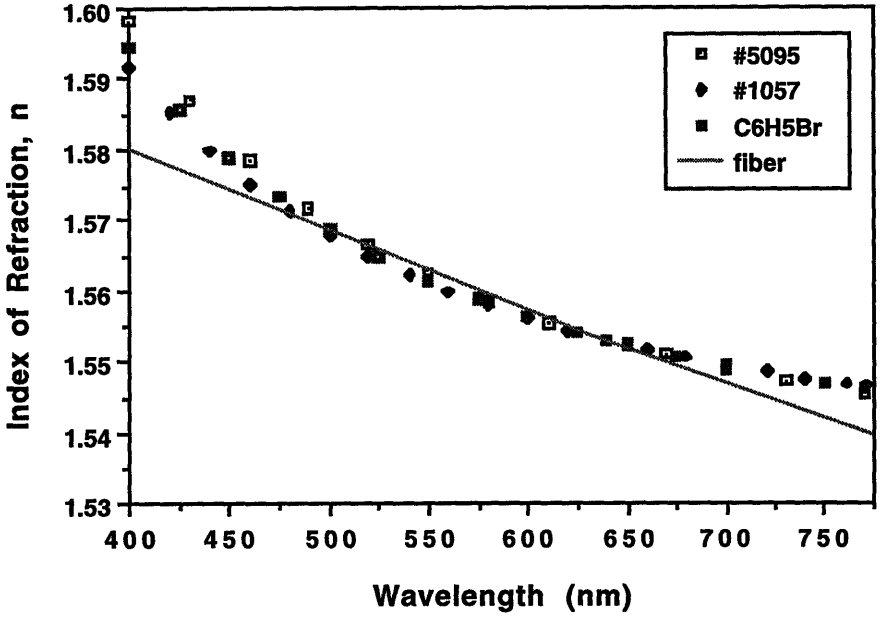
$$\Delta n(\lambda) = \left| n_f(\lambda) - n_l(\lambda) \right| \tag{2.32}$$

where $n_f(\lambda)$ and $n_l(\lambda)$ are the wavelength dependent refractive indices of the fiber and liquid, respectively. As shown in Figure 2.10(b), the approximate refractive index mismatch is on the order of 10^{-3} or less over a large portion of the visible spectrum.

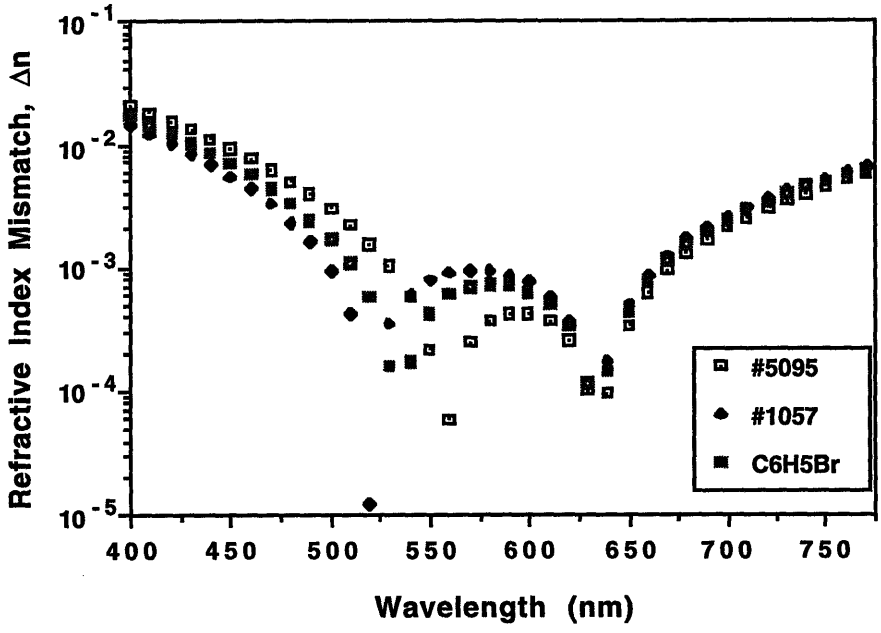
As shown in Figure 2.10, the wavelength distribution of the illumination source will affect the quality of the observed match between fibers and test fluid. For the fibers and fluids used, it can be seen that a purely monochromatic source is not necessary if the visible output of the source is predominantly in the wavelengths $\lambda \geq 500$ nm. The wavelength distribution of an incandescent source, such as a flood lamp, can be approximated by modeling it as a blackbody following the Planck radiation law [4]

$$I_b(\lambda) = \frac{2\pi hc^2}{\lambda^5}\left[\frac{1}{e^{hc/\lambda kT} - 1}\right] \tag{2.33}$$

where $I_b(\lambda)$ is the emitted intensity per unit wavelength interval, k is Boltzmann's



(a)



(b)

Figure 2.10 Dispersion of test materials: (a)wavelength dependence of test fluids and typical glass fiber (b) resulting mismatch in refractive indices, $\Delta n = \left| n_{fiber} - n_{liquid} \right|$.

constant, c is the speed of light in vacuum, $h = (6.6256 \pm 0.0005) \times 10^{-34}$ J - s is Planck's constant and T is the blackbody temperature in degrees Kelvin. For a typical incandescent source [5], $T \approx 2500$K. Integrating Eq. 2.33 over the interval, $[\lambda_o, \lambda]$, yields the total energy flux, $I(\lambda_o, \lambda)$, emitted over $[\lambda_o, \lambda]$, or

$$I(\lambda_o, \lambda) = \int_{\lambda_o}^{\lambda} I_b(\xi) d\xi \tag{2.34}$$

Integrating Eq. 2.34 from $(390 \text{ nm} \leq \lambda \leq 780 \text{ nm})$ yields the total energy flux emitted by the illumination source over the visible spectrum. Figure 2.11, a plot of $I(\lambda_o, \lambda)$ normalized by the total visible energy flux, shows the relative wavelength distribution of the energy radiated by a typical incandescent source at 2500K. As shown by the plot, less than 10% of the total visible energy radiated by the source is contributed by wavelengths $\lambda < 500$ nm. Consequently, an incandescent source can be used with the selected test fluids as long as the sensor is limited to the visible portion of the spectrum.

The refractive indices of the fluids can be manipulated by introducing appropriate amounts of miscible additives of varying refractive indices. The effective index of refraction of the mixture, n_{mix}, can be found using using an empirical relation [3] based on the Gladstone-Dale specific refraction of each component, r_i, given by

$$r_i = \frac{n_i - 1}{\rho_i} \tag{2.35}$$

where n_i and ρ_i are the index of refraction and density, respectively, of the ith of the mixture. The resulting value of n_{mix} can then be found from the expression for the specific refraction of the mixture, r_{mix}, based on a weighted average of the component r_i

$$\begin{aligned} n_{mix} &= 1 + \rho_{mix} r_{mix} \\ &= 1 + \rho_{mix} \sum_i w_i r_i \end{aligned} \tag{2.36}$$

where w_i is the weight fraction of the ith component and ρ_{mix} is the final density of the mixture. Using the definition of density (i.e., mass per unit volume), Eq. 2.36 can be simplified to

$$n_{mix} = \sum_i v_i n_i \tag{2.37}$$

where v_i is the volume fraction of the ith component of the mixture. Since all three fabrics had similar fiber refractive indices, the same fluids were used with all three fabrics.

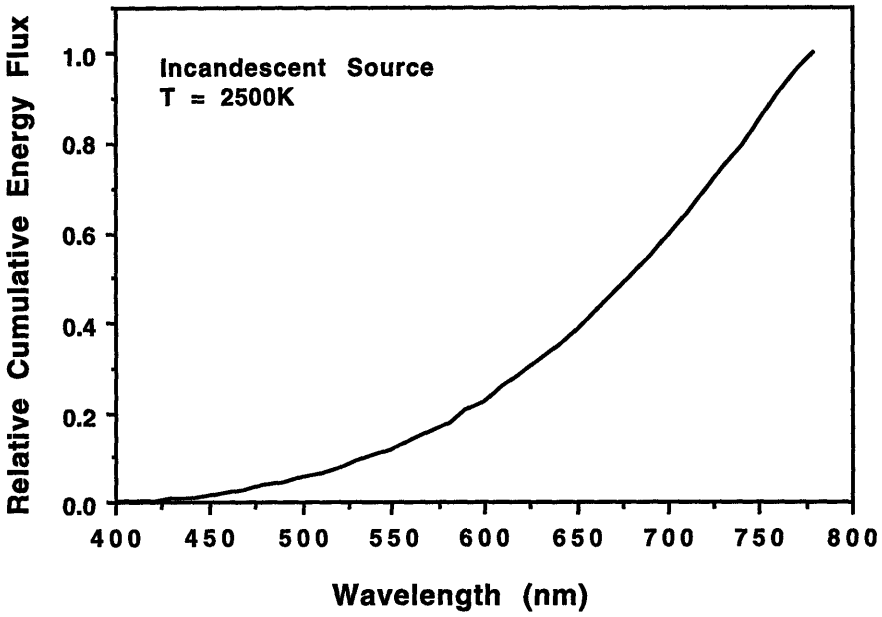


Figure 2.11 Relative distribution of emitted energy flux of incandescent source.

2.3 References

1. Hecht, E., 1988, *Optics*, 2nd ed., Addison-Wesley, Reading, MA.

2. Born, M., and Wolf, E., 1975, *Principles of Optics*, 5th ed., Pergamon Press, Oxford.

3. Chamot, E., and Mason, C., 1958, *Handbook of Chemical Microscopy*, v1, 3rd ed., John Wiley and Sons, New York.

4. Look, D., and Sauer, H., 1982, Thermodynamics,Brooks/Cole Engineering Division, Monterey, CA, p. 435.

5. Oriel Corporation, 1985, *Light Sources, Monochromators, and Detection Systems*, v2.

3 MATERIALS, EQUIPMENT, AND PROCEDURES

This chapter describes the procedures and apparatus used for mold filling studies using the flow visualization method described in Chapter 2. The goal of this work was to examine fiber and tow-scale flows within typical reinforcement materials. As a result of initial flow visualization experiments on both undeformed and deformed fabrics, the investigation moved on to a detailed study of void entrapment at the fiber scale. This part of the research was carried out using scale models of typical fiber tows.

Figure 3.1 schematically illustrates the apparatus used in this study. The illumination source was positioned so as to backlight the specimen. In the setup, the laser light is expanded and filtered and then directed onto a diffuser. The purpose of the diffuser is to produce uniform illumination of the preform. A sheet of vellum was used as the diffuser. Intensity variation of the illumination source imaged through the diffuser was measured to be approximately ±2% of the mean intensity.

Depending on the desired resolution (i.e., image magnification), a number of images of the dry, backlit preform were captured using a frame grabber (Perceptics PixelGrabber™) and stored on a Macintosh IIfx computer. Analysis of the pre-impregnation images, to be discussed in the next section, yielded a 2-D map of the relative variations of the thickness-integrated fiber volume fraction. Visual inspection of the dry, backlit preform gave

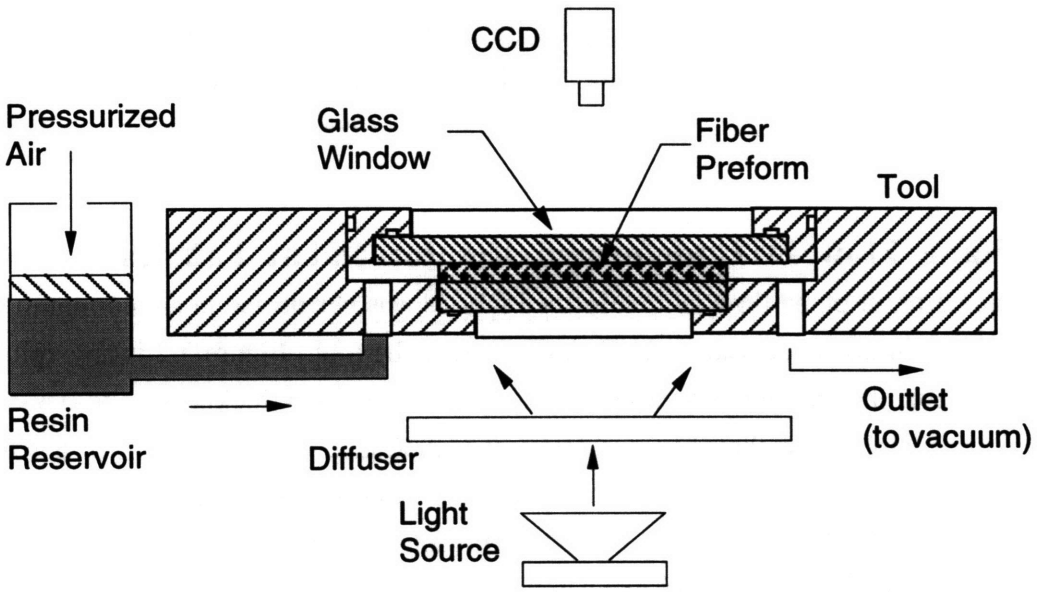


Figure 3.1 Schematic of basic setup.

qualitative information on the location of high fiber content, and therefore, low permeability regions within the preform. Infiltration progress was recorded during the experiments on video tape at 30 frames a second for later retrieval and analysis.

3.1 Materials

Four commercial, glass fiber fabrics were used, two unidirectional fabrics (ANCAREF J600 and Bean Fiber Glass VECTORPLY V18), a plain woven fabric (Fiber Glass Evercoat Co., Sea-Glass Cloth) and an eight-harness satin weave (BGF Industries Style 3783 8HS E-glass). Table A.2 summarizes physical properties of the fabrics. Experiments on scale models of typical tows were performed using polymer fibers (Berkley Trilene, a commercial fishing line), 4.5×10^{-4} m and 7×10^{-4} m in diameter and corresponding to geometric scaling by a factor of approximately 45 and 70, repectively. The refractive index of these fibers was similar to that of the glass fibers in the commercial fabrics.

In addition to the optical compatibility of the test fluids discussed in Chapter 2, other variables that were considered included density, viscosity and surface tension. The goal was to select fluids that would approximate typical resins used in RTM and SRIM applications. Table B.2 summarizes physical properties of the fluids used in the experiments.

Although the refractive index match using C_6H_5Br was very good, it was not used for many experiments because of the health risks associated with prolonged exposure (C_6H_5Br can be carcinogenic). In addition to its health risks, C_6H_5Br was deemed impractical because of the difficulty in identifying other miscible liquids that could be combined with C_6H_5Br to produce changes in other properties, such as viscosity.

3.2 Plane Flow in Undeformed Reinforcement Fabrics

Visualization of plane flow in undeformed fabrics was performed in an aluminum tool with illumination and observation windows built into the tool. Figure 3.2 is an exploded view of the tool. The mold has a variable uniform, molding gap thickness up to 0.635 cm (0.25 in) repeatable to within $\pm 1.27 \times 10^{-3}$ cm ($\pm$0.0005 in), inlet plenum to ensure a uniform inlet flowfront, 0.95 cm (0.375 in) thick glass observation windows, 10 cm x 7.5 cm (4 in x 3 in). Sealing of the tool for use with vacuum was achieved using O-rings as indicated in Figure 3.2. Prolonged exposure of the O-ring material to the molding fluids was found to produce swelling and cracking of the O-rings. Consequently, selection of O-

ring material was constrained by compatibility with the molding fluids. In addition, it was necessary to periodically replace the O-rings in order to maintain proper sealing.

In order to minimize racetracking along the sides of the mold, preform layers were cut with a roller cutter and carefully placed in the mold. Mold gap thickness was measured with a custom-built, deep-throat calliper and adjusted until the desired gap thickness was uniform to within $\pm 1.27 \times 10^{-3}$ cm (± 0.0005 in). Racetracking on the top surface was minimized by ensuring intimate contact between the upper and lower mold surfaces and the preform. This was achieved by maintaining the mold gap thickness at or below the mat thickness defined as the mat thickness when the fabric fibers begin to carry load during a compression test.

The "molding" fluid was placed in a hydraulic cylinder which acted as the feed reservoir. Depending on whether or not a vacuum was applied during the impregnation experiment, the fluid was degassed (i.e., volatile gases dissolved in the fluid were removed) by applying a vacuum to the fluid reservoir for an amount of time experimentally determined for each fluid. The cylinder piston was driven using a constant pressure gas, which forced the fluid into the mold. Flow rate was determined by measuring the piston displacement in the feed cylinder (reservoir) and a pressure transducer was mounted at the cylinder outlet to measure the change in back pressure as the fluid impregnated the specimen. The experimental pressure range did not exceed 300 kPa (45 psi).

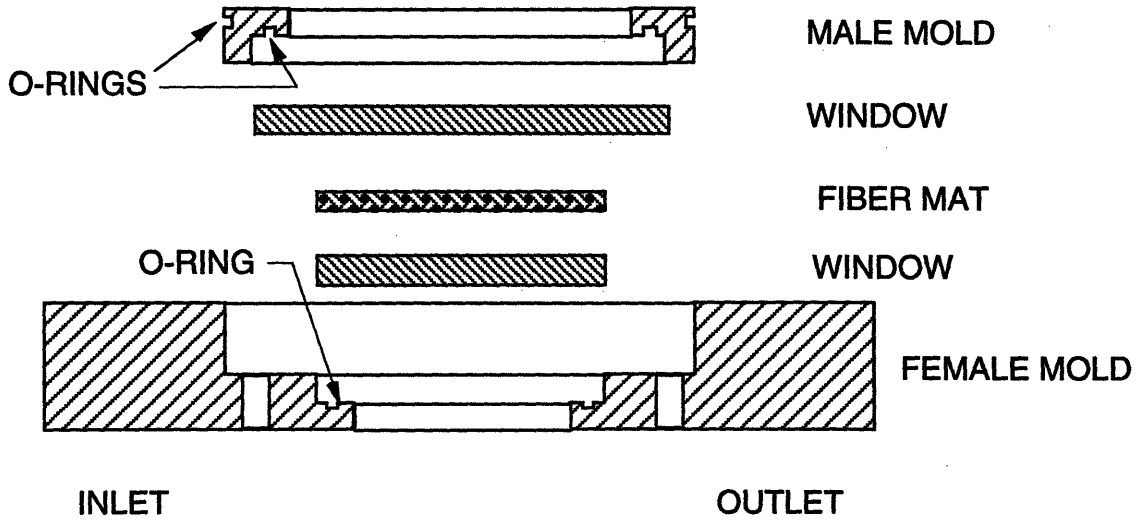


Figure 3.2 Exploded view of plane flow mold.

3.3 Radial Flow in Deformed Reinforcement Fabrics

Observation of radial flow in deformed fabrics were performed using the apparatus developed in [1]. The basic set-up was modified to enable transmission of light through the thickness of the sample. This was done by replacing the original, aluminum lower plate with one of machined PMMA. As discussed in [1], selection of tooling materials is complicated by issues of compatibility of the model fluid and the tooling material. The PMMA was found to be acceptably resistant to the model fluids used in this thesis.

These experiments were performed using the eight-harness satin weave (BGF Industries Style 3783 8HS E-glass). The fabric sample was deformed using the trellising rack described in [1]. The resulting observable area was a 12.7 cm diameter circle with a 0.625 cm diameter center gate. Fluid injection was done using the same set-up as in the plane flow experiments. No experiments using vacuum were performed using this set-up since the sample is exposed to the atmosphere. Observations of flow in the deformed fabric were all performed on single layer specimens.

3.4 Uniaxial Flow in Model Tow

In order to facilitate observations of interfiber flow, experiments investigating fiber-scale void formation were carried out using scale models of a typical tow, constructed out of large polymer fibers 45 and 70 times larger in diameter than typical glass fibers used in the commercial fabrics. Figure 3.3 illustrates the basic set-up. The fiber-volume fraction, V_f, used in the experiments was within the range $0.59 \leq V_f \leq 0.73$. The specimen length was approximately 3×10^{-1}m. The "mold" was pyrex tubing cut to the proper length. Scanning of the specimen was done using the set-up developed in [2].

Specimens were prepared by cutting lengths of polymer fiber to slightly longer than the length of the test tube. The fibers were then grouped into manageable bundles and placed within the test tube, taking care to mainain fiber alignment and to avoid any twisting of fibers. Once the desired number of fibers was placed within the tube, one end was selected as the mold "inlet" and the fibers at that end were trimmed to present a uniform "entrance" into the model tow. After trimming the tow entrance, the fiber bundle was carefully pulled up from the exit end so that the tow entrance was located within the tube. The fibers at the tube exit were then trimmed so that the necessary fittings for the outlet hose could be attached.

Unlike the experiments described above, the model tow experiments were performed at a constant flow rate instead of constant inlet pressure. This was to ensure uniformity of local flow conditions, expressed using the dimensionless group $Ca = \mu U/\gamma$, throughout

the entire sample. The significance of Ca will be discussed in Chapter 4. As shown in Figure 3.3, the constant flow rate condition was produced by replacing the pressure reservoir in Figure 3.1 with a connection to a tensile testing machine that acted as a positive displacement pump. The resulting average standard deviation in flow front speed, and consequently Ca, along the length of a typical specimen was under 15% of the average value.

Selection of the tube inner diameter, D, and fiber diameter, d_f, was constrained by the need to reduce the effects of the tube walls on flow in the interior of the bundle. This was done by ensuring that the number of fibers was above an experimentally determined minimum. Initial experiments indicated that a ratio of $D/d_f \geq 11$ provided results that were consistent with flow observed in commercial glass fibers packed in a similar test tube and with predictions based on flow in infinite arrays of fibers (see Eq. 4.18). The experimental results reported in this thesis were produced using the small $\left(d_f = 4.5\times10^{-4}\text{m}\right)$ fibers in test tubes with $D = 5\times10^{-3}\text{m}$. This resulted in models with between 70 and 90 fibers and, consequently, between three and four layers of fibers between the center of the model and the tube wall for the range of fiber volume fractions considered.

In order to eliminate the effects of specimen-to-specimen variations in fiber packing, each specimen was used for multiple runs over the complete range of capillary numbers, Ca, considered. In order to produce repeatable initial conditions for each experiment, samples were first impregnated, then drained, and exposed to vacuum for a set amount of time before each recorded run. Initial experiments showed that applying the vacuum for 1 minute after drainage of the bulk fluid produced repeatable results.

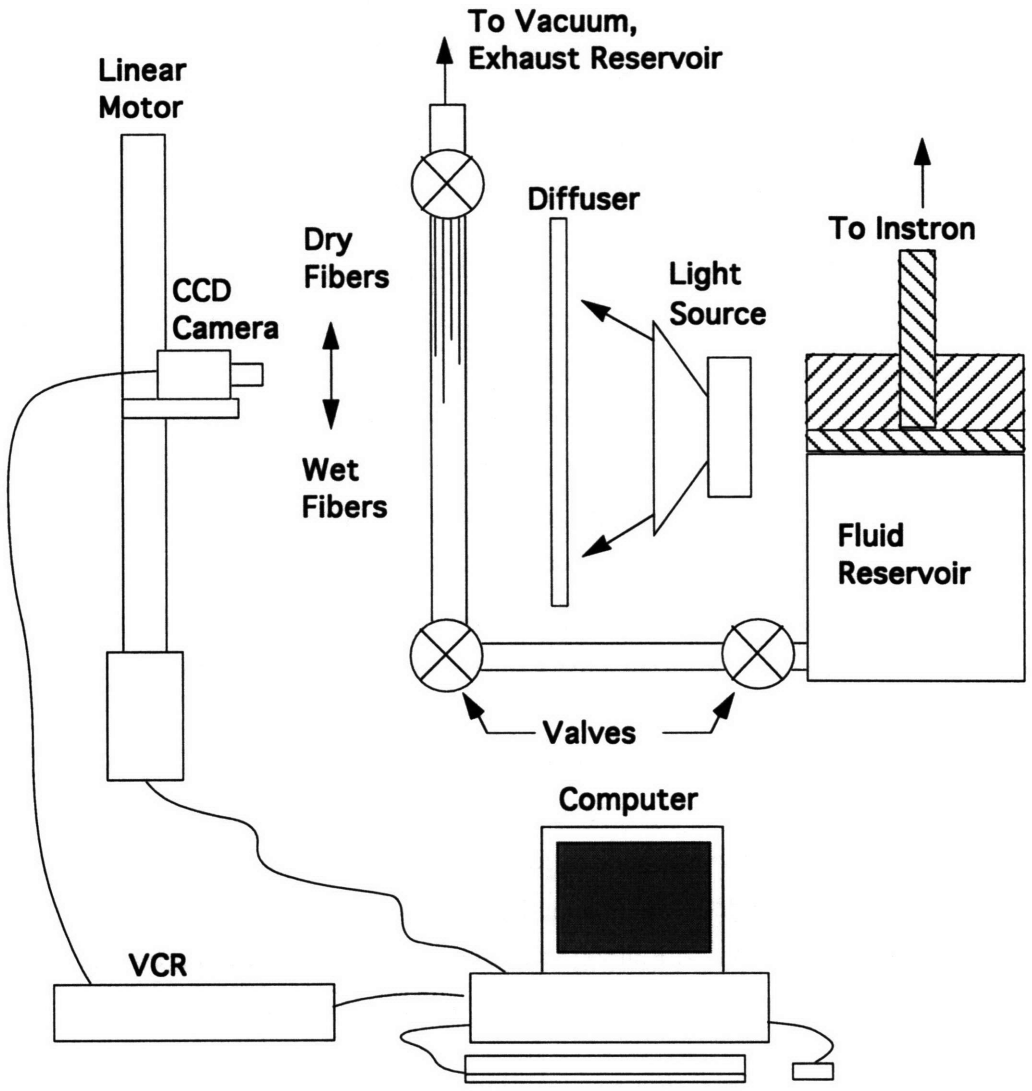


Figure 3.3 Schematic setup for fiber-scale experiments.

3.5 References

1. Ueda, S., 1993, *Visualization of the Flow in Complex Shapes Made by Resin Transfer Molding*, S.M. thesis, Massachusetts Institute of Technology.

2. Poonen, A., 1992, *Design and Construction of a Flow Front Tracking Mechanism*, S.B. thesis, Massachusetts Institute of Technology.

PART II:

FLOW VISUALIZATION STUDY OF MOLD FILLING IN RTM

4 Analysis of Flow Regimes in Fiber Impregnation Processes

Based on a survey of literature on industrial practice and research and the results of the experiments and simulations performed in the course of this thesis, a classification scheme for the basic flow regimes present in fiber impregnation processes is proposed. The basic mechanisms characterizing each regime and implications for process design are discussed.

4.1 Wetting and Void Entrapment Mechanisms

Review of the literature indicates that there are two distinct mechanisms of fiber-scale, void entrapment possible. The first is due to variability in the structure of the porous medium. Local variations in the pore structure cause fingering and potential entrapment of air pockets (this mechanism is analogous to the tow-scale entrapment mechnism modeled in [1]). The second mechanism is due to instability of the meniscus leading to meniscus collapse and air entrapment. This raises the questions of when these two types of void entrapment occur, whether or not these mechanisms are independent, competing or reinforcing, and whether or not these (one or both) mechanisms are significant in composites processing.

In order to simplify the discussion, the effects of gravity will be neglected (this corresponds to a nominally horizontal mold). Consider the advance of the meniscus moving at a speed U in a circular capillary of radius R as sketched in Figure 4.1.

The liquid partially wets the wall of the capillary, the meniscus meeting the wall at a contact angle θ measured through the liquid. Assuming inertia-free, quasi-steady flow, an axial (or z-direction) linear momentum balance yields

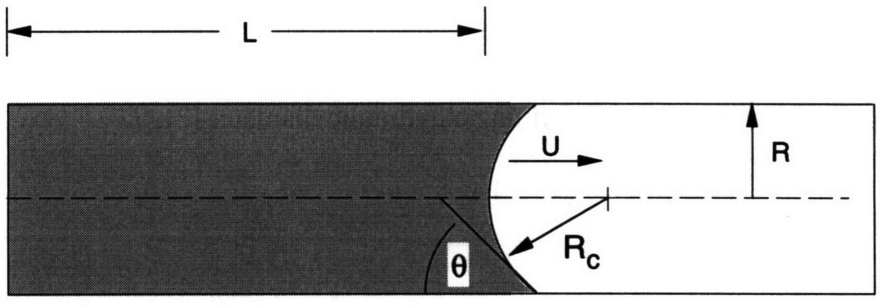


Figure 4.1 Schematic of advancing meniscus.

$$0 = -\frac{1}{\mu}\frac{dp}{dz} + \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{\partial u}{\partial r}\right) \tag{4.1}$$

Writing an order of magnitude approximation of Eq. 4.1,

$$\begin{aligned} 0 &\sim \frac{p_o}{L} + \frac{p_c}{L} + \frac{\mu U}{R^2} \\ &\sim \frac{p_o}{L} + \frac{2\gamma\cos\theta}{LR} + \frac{\mu U}{R^2} \\ 0 &\sim \frac{p_o}{L} + \frac{\mu U}{R^2}\left(\frac{2\gamma\cos\theta}{\mu U}\frac{R}{L} + 1\right) \end{aligned} \tag{4.2}$$

where the symbol ~ signifies order of magnitude equivalence (i.e., an estimate to within a factor of three or so) and p_o/L is the applied pressure gradient over the characteristic length L, U is a characteristic speed describing the advance of the meniscus, R is a characteristic radial dimension (in this case the capillary radius) and p_c is the interfacial pressure difference given by

$$p_c = \frac{2\gamma}{R_c} = \frac{2\gamma}{R/\cos\theta} = \frac{2\gamma\cos\theta}{R} \tag{4.3}$$

As written, Eq. 4.2 is a statement of the balance of forces acting on a unit volume of the advancing fluid. Consequently, in order for the surface tension force per unit volume to be much less than the viscous force per unit volume of liquid

$$\frac{2\gamma\cos\theta}{\mu U}\frac{R}{L} << 1 \tag{4.4}$$

or,

$$\frac{\mu U}{\gamma\cos\theta} >> \frac{2R}{L} = \frac{D}{L} \tag{4.5}$$

where $D = 2R$ (in this case the diameter of the capillary). Equation 4.5 can be adapted for flow along the fibers in a tow by defining a hydraulic diameter [2], D_h,

$$D_h = \frac{4A}{S} \tag{4.6}$$

where A and S are the area and wetted perimeter of the cross section of interest. For flow along the fibers, the hydraulic diameter can be approximated by

$$D_h = d_f\left(\frac{1-V_f}{V_f}\right) \tag{4.7}$$

Substituting Eq. 4.7 into Eq. 4.5 yields the criterion for surface tension forces to be much less than the viscous forces in flow along the fibers in an aligned fiber bundle

$$\frac{\mu U}{\gamma\cos\theta} >> \frac{d_f}{L}\left(\frac{1-V_f}{V_f}\right) \tag{4.8}$$

Analogously, for flow transverse to the bundle axis, a limiting condition on the balance of surface tension to viscous forces can be approximated by modeling the gap between adjacent fibers as parallel plates separated by a distance δ_o. Consequently, the characteristic "radial" dimension would be the fiber separation δ_o which is related to the fiber diameter, fiber volume fraction V_f, and packing arrangement within the tow by

$$\delta_o = d_f\left[\left(\frac{V_a}{V_f}\right)^{1/2} - 1\right] \tag{4.9}$$

where V_a is the maximum possible fiber-volume fraction for a given packing structure. It is equal to 0.785 for square arrays and 0.907 for hexagonal arrays. The characteristic length defining the dominant pressure gradient would be the radius of the tow, R_{tow}. The resulting criterion for surface tension forces to be much less than the viscous forces in flow transverse to the fibers is

$$\frac{\mu U}{\gamma\cos\theta} >> \frac{\delta_o}{R_{tow}} = \frac{d_f}{R_{tow}}\left[\left(\frac{V_a}{V_f}\right)^{1/2} - 1\right] \tag{4.10}$$

Analogously, for flow transverse to a unidirectional fabric, one might substitute the fabric thickness (i.e., transverse dimension which now would also correspond to the direction of the dominant pressure gradient) for R_{tow} in Eq. 4.10.

The dimensionless group $\mu U/\gamma$ is commonly used as a measure of the ratio of viscous force to surface tension force per unit volume of fluid and is refered to as the capillary number, Ca. Other authors have prefered to use a modified capillary number, $Ca^* = \mu U/\gamma\cos\theta$. Use of the modified capillary number Ca^* is complicated by the fact that the contact angle θ is, in fact, dependent on the value of the capillary number Ca.

Although most authors have used the static, or equilibrium, contact angle θ_s when evaluating Ca^*, it will prove important in interpreting the reported data to understand, at least qualitatively, the dependence of the contact angle on capillary number. Due to hysteris [3], the exact relationship between the apparent contact angle and Ca will be different for the advancing and receding meniscii (i.e., wetting and drying cases, respectively). In general, however, the receding contact angle will also be dependent on Ca. For the purpose of illustrating the basic ideas, consideration of the advancing contact angle will be sufficient.

As demonstrated in experiments on the advancing meniscus in the capillary geometry of Figure 4.1 [4] and discussed in [5], the contact angle can be correlated to the group $Ca = \mu U/\gamma$ using the relation

$$Ca = F(\theta) - F(\theta_s) \tag{4.11}$$

where

$$F(\theta) = const \times \theta^m \tag{4.12}$$

with $m = 3 \pm 0.5$. For complete wetting as defined by the static, or equilibrium, contact angle $(\theta_s = 0)$ $F(\theta_s) = 0$. For both fully and partially wetting fluids, the contact angle was seen to increase from the static value θ_s to a maximum of π as $Ca \to 1$. Solving for the constant in Eq. 4.12, one can write formally

$$\frac{\theta}{\pi} \sim Ca^{1/m} \approx Ca^{1/3} \tag{4.13a}$$

for full wetting and

$$\frac{\theta}{\pi} \sim \left[Ca + \left(\frac{\theta_s}{\pi} \right)^m \right]^{1/m} \approx \left[Ca + \left(\frac{\theta_s}{\pi} \right)^3 \right]^{1/3} \tag{4.13b}$$

for partial wetting $(\theta_s \neq 0)$.

Equation 4.4 suggests that the wetting of a surface by a liquid can be divided into two distinct regimes categorized by the value of the dimensionless group $Ca = \mu U/\gamma$. The first, when $Ca \leq 1$, is characterized by significant surface tension forces. The second, when $Ca >> 1$, is dominated by viscous forces. Furthermore, Eq. 4.13 suggests that surface tension dominated flow can be categorized into two additional regimes. The first, when $Ca << 1$, is characterized by advance of the meniscus wetting the solid at a contact

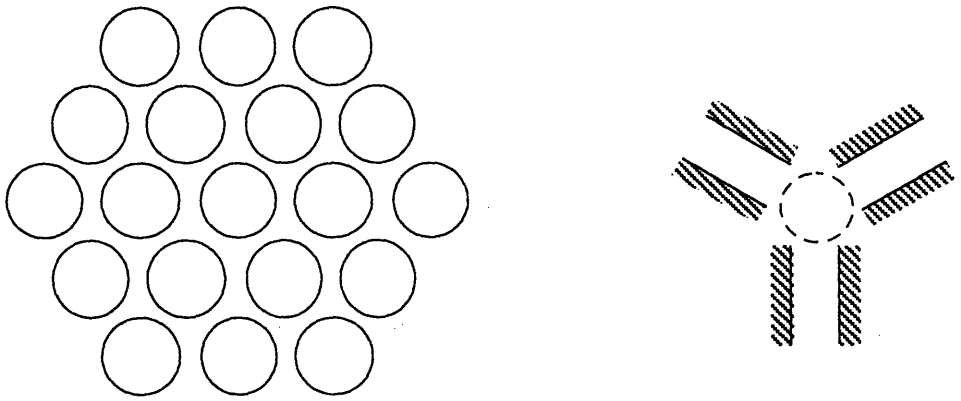
angle varying approximately as $Ca^{1/3}$. The second, when $Ca \rightarrow 1$ is characterized by a contact angle approaching π . The implications of flow in these two regimes will be examined in greater detail below.

4.1.1 Low Capillary Number Flow: $Ca << 1$

Due to non-uniformity of the pore space formed by the fibers within the bundle, the advancing flow front will be uneven and characterized by fingers extending beyond the boundary between the fully and partially saturated portions of the bundle. Note that the type of non-uniformity responsible for this observed behavior is the in-plane variation of the pore space (i.e., at any arbitrary value of z) which arises quite naturally when packing circular cylinders together. It is this variability, sketched in Figure 4.2(a), that gives rise to the "pores" and "gaps" analyzed in Chapter 4.

Understanding the fingering at the flow front is important in molding processes such as RTM and SRIM because void entrapment can only occur if the flow front is uneven. Results of previous flow visualization experiments [6-11] on this aspect of mold filling, however, have been typically of limited use because of the obstructed view into the preform, the results being limited to the gap between the preform and the observation window. Analytical efforts [12,13] have concentrated on the static equilibrium problem related to capillary rise in vertical fiber arrays and stable configurations of stationary fluid columns in horizontal fiber arrays [14].

In order to understand the nature of the fingering observed in RTM type processes, consider the conditions necessary for the existence of a stable finger within the ideal array of fibers sketched in Figure 4.2. The simplified model shown in Figure 4.2(b) will be the



(a) (b)

Figure 4.2 Ideal fiber array: (a) in-plane non-uniformity due to fiber shape, (b) unit cell of simplified model for analysis.

basis for the following analysis. As a start to the analysis, the effects of gravity will be neglected (this would correspond to injection into a horizontal tool). Non-uniformity of the preform will result in the formation of fingers at the advancing flow front, such as the one shown schematically in Figure 4.3.

In order for a finger of length z_f to be stable, the speed of the finger tip, U_f, must equal the speed of the bulk front, U_b, or $U_f = U_b = U$, which can be estimated as

$$U = U_f \approx -\frac{\delta^2}{12\mu}\frac{dp}{dz} \tag{4.14}$$

where μ is the fluid viscosity, δ is the minimum fiber separation given by Eq. 4.9, and dp/dz is the axial pressure gradient within the finger. It can be approximated as

$$\begin{aligned}\frac{dp}{dz} &\approx \frac{P_f - P_b}{z_f} \\ &\approx \frac{\gamma}{z_f}\left(-\frac{1}{R_f}+\frac{2}{R_b}\right) \\ &\approx \frac{\gamma}{z_f}\left(\frac{2}{R_b}-\frac{1}{R_f}\right)\end{aligned} \tag{4.15}$$

where R_f and R_b are the radii of curvature for the advancing meniscii for the finger and the bulk front, repsectively. Under the assumption of the simplified geometry of Figure 4.2(b), the finger meniscus will have single curvature (this comes from the parallel plate assumption for the gap) and the bulk (i.e., pore) meniscus is assumed to be spherical (i.e., having double curvature). Substituting Eq. 4.15 into Eq. 4.14 yields

$$\begin{aligned}U &\approx -\frac{\delta^2}{12\mu}\cdot\frac{\gamma}{z_f}\left(\frac{2}{R_b}-\frac{1}{R_f}\right) \\ &= \frac{\gamma}{12\mu}\left(\frac{\delta}{z_f}\right)\left(\frac{\delta}{R_f}-\frac{2\delta}{R_b}\right)\end{aligned} \tag{4.16}$$

Notice that the linear dimensions can all be scaled quite naturally by the minimum fiber separation, δ. Assuming that the radii of curvature for the advancing meniscii are approximately half the fiber separation (consistent with the parallel plate assumption) for the finger front and half the axial hydraulic diameter for the bulk front, one can write

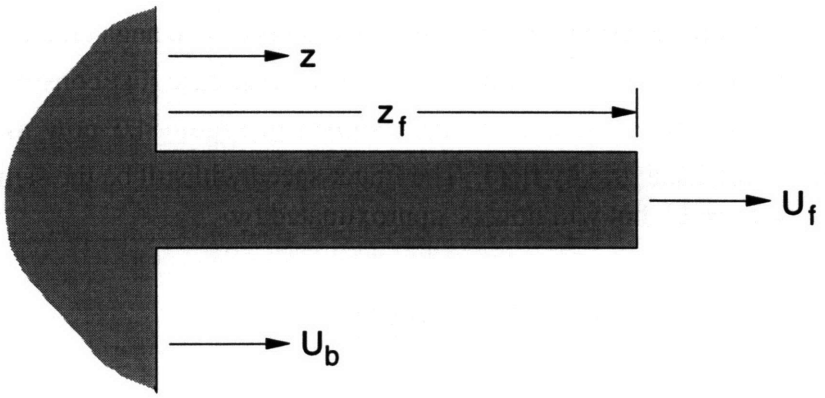


Figure 4.3 Nomenclature for analysis of ideal, stable finger.

$$R_f \approx \frac{\delta}{2} = \frac{d_f}{2}\left[\left(\frac{V_a}{V_f}\right)^{1/2} - 1\right]$$
$$R_b \approx \frac{D_h}{2} = \frac{d_f}{2}\left(\frac{1-V_f}{V_f}\right) \tag{4.17}$$

Substituting Eqs 4.17 into Eq. 4.16 and rearranging terms yields the relationship of the stable finger length z_f/δ to the bundle structure (V_f, V_a) and the local flow conditions $(Ca = \mu U/\gamma)$,

$$\frac{z_f}{\delta} = \frac{1}{6}\left(\frac{1}{Ca}\right)\left\{1 - \frac{2V_f\left[\left(V_a/V_f\right)^{1/2} - 1\right]}{1-V_f}\right\} \tag{4.18}$$

where $Ca = \mu U/\gamma$ is the capillary number. Examining the asymptotic behavior predicted by Eq. 4.18, we see that

$$\lim_{Ca\to 0} z_f = \infty \tag{4.19a}$$

$$\lim_{Ca\to \infty} z_f = 0 \tag{4.19b}$$

and

$$\lim_{\delta\to 0} z_f = \lim_{V_f \to V_a} z_f = \infty \tag{4.20}$$

The effect of gravity needs to be considered when examining flow in a vertical tool. Of particular interest will be determining when gravity effects can be neglected in the design of the process. The analysis will be similar to the horizontal case just considered. For the sake of simplicity, consider the stable finger shown in Figure 4.3 now assumed to be flowing up against the local gravity field. The finger speed will still be the same as the bulk front speed, $U_f = U_b = U$, but will now be approximated by

$$U \approx \frac{\delta^2}{12\mu}\left[-\frac{d}{dz}(p+\rho gz)\right] \tag{4.21}$$

where ρ is the density of the impregnating fluid and all other variables are as defined in the horizontal case. Similarly, the combined pressure and gravity potential gradient within the finger will be given by

$$\begin{aligned} -\frac{d}{dz}(p+\rho gz) &\approx \frac{P_b - P_f - \rho g z_f}{z_f} \\ &\approx \frac{\gamma}{z_f}\left(\frac{1}{R_f} - \frac{2}{R_b}\right) - \rho g \end{aligned} \tag{4.22}$$

Substituting Eq. 4.22 into Eq. 4.21, approximating R_f and R_b using Eq. 4.17, and rearranging terms yields the condition for a stable vertical finger. It is given by

$$\frac{z_f}{\delta} = \frac{1}{6}\left(\frac{1}{Ca + Bo/12}\right)\left\{1 - \frac{2V_f\left[\left(V_a/V_f\right)^{1/2} - 1\right]}{1 - V_f}\right\} \tag{4.23}$$

where, as usual, $Ca = \mu U/\gamma$ is the capillary number and $Bo = \rho g \delta^2/\gamma$ is the Bond number [15,16] based on the minimum fiber separation, δ. As used here, the Bond number is a measure of the relative importance of gravity compared to surface tension within the moving finger.

Comparing Eqs 4.18 and 4.23, one sees that the only difference between the horizontal and vertical flow cases is the contribution of the Bond number in the denominator of Eq. 4.23 which acts to reduce the length of the finger. As shown by Eq. 4.9, the minimum fiber separation, δ, scales with the fiber diameter, d_f. Consequently, the Bond number should scale as $Bo \sim d_f^2$ and the effect of gravity on the length of the stable finger should decrease dramatically as the fiber diameter decreases. Substituting Eq. 4.9 into the definition of the Bond number, one finds that

$$Bo = \frac{\rho g d_f^2}{\gamma}\left[\left(V_a/V_f\right)^{1/2} - 1\right]^2 \tag{4.24}$$

which shows not only the dependence of Bo on d_f but also on the packing geometry (expressed in terms of the available fiber volume fraction, V_a) and the fiber volume fraction, V_f.

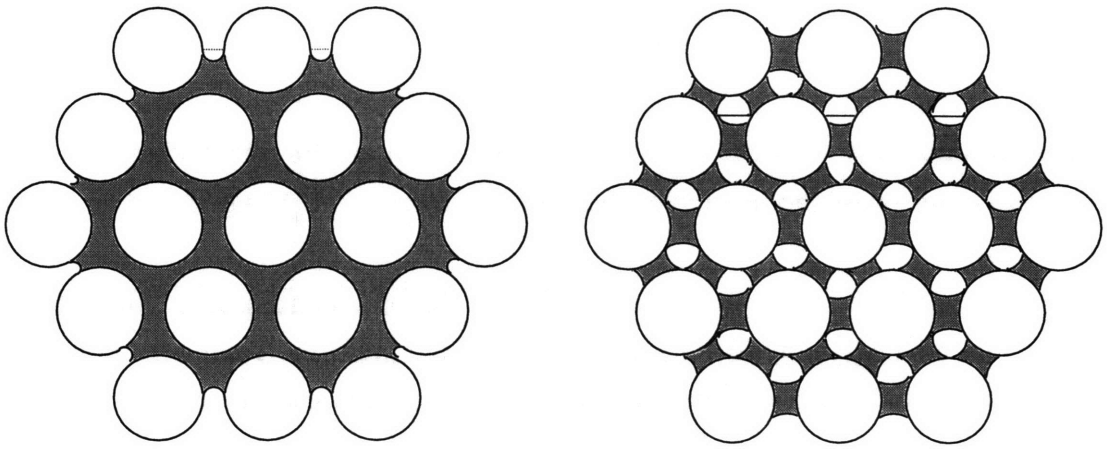
Using Eq. 4.24, it is now possible to estimate Bo and, consequently, the effect of changing tool orientation on the fingering within a given preform. Taking the ratio of the horizontal to vertical finger lengths given in Eqs 4.18 and 4.23, respectively yields

$$\left.\frac{z_f^h}{z_f^v}\right|_{V_a,V_f,Ca} \sim 1 + \frac{1}{12}\left(\frac{Bo}{Ca}\right) \tag{4.25}$$

where the ratio is evaluated for a fixed V_a, V_f, and Ca. Note that the coefficient $(1/12)$ of the ratio Bo/Ca is a function of the geometry of the inter-fiber gap, modeled here as two parallel plates.

The model of a stable finger propagating through the fiber bundle suggests a possible void entrapment mechanism. As discussed in [14], the advancing fluid fronts in both the fingers within the inter-fiber gaps and the bulk front (characterized by the meniscii formed within the large pores) create a complex, fluid surface. For the surface to be in equilibrium, it will tend to adopt the lowest energy configuration possible. In the study of the equilibrium fluid configurations adopted by fluids wetting arrays of vertical fibers [13], it was noted that the effect of gravity was to stabilize the growth of the fluid fingers. Removing the stabilizing force of gravity (by rotating the fibers to a horizontal position) it was seen that the fluid within the bundle would adopt one of two configurations, sketched in Figure 4.4, depending on the geometry of the pore space formed by the fibers.

Analogously, the dynamic case during impregnation of the fiber bundle might be characterized by the advancing fluid surface seeking the lowest possible energy configuration. In the case of uniformly packed, identical fibers this lowest energy case would be characterized by stable fingers described by Eq. 4.18 (or Eq. 4.23). In a real composite, however, fiber waviness, variations in fiber diameter and fiber surface chemistry (e.g., dirt and other contaminants) will cause variability in the local geometry within the fiber bundle. If these variations are significant, they can potentially force the advancing fluid surface into unstable configurations. The resulting non-equilibrium adjustments in the meniscus position can then lead to entrapment of air within the preform.



(a) (b)

Figure 4.4 Possible fluid configurations within fiber bundle.

4.1.2 High Capillary Number Flow: $Ca \geq 1$

As discussed above, for both fully and partially wetting fluids, the contact angle θ is a function of the capillary number Ca with $\theta \rightarrow \pi$ as $Ca \rightarrow 1$. The significance of this dependence of the contact angle on the capillary number is far greater than the associated change in the value of $\cos\theta$ in Eq. 4.4. As discussed in [17-21] there is a maximum speed of wetting below which the advancing meniscus is stable. Above this maximum speed of wetting, $\theta \approx \pi$, the meniscus becomes unstable, and adopts a sawtooth profile [20,21] as illustrated in Figure 4.5. Air is then entrapped within the fluid as the meniscus collapses locally at the trailing vertices of the advancing fluid front. This result has been observed in wetting of various tapes plunging into initially quiescent tanks of resin [20,21], single fibers [17,18], and bundles of fibers [19]. The studies in [17-19] were designed to model filament winding. As such, they were similar to those in [20,21] in that the fibers were also pulled through initially quiescent tanks of resin. Of particular interest in all of these studies was the identification of possible process rate limiting mechanisms and, therefore, these studies examined a range of wetting speeds on the order of 10^{-3} - 10^{3} cm/s. In all of theses studies, significant air entrapment was found to occur at the higher wetting speeds.

An important difference between the filament winding and mold filling experiments is that the void content at high impregnation rates is much lower in the mold filling experiments than in the filament winding experiments. As will be discussed in Section 4.2, this difference is due to mobilization, or flushing of the voids in the mold filling processes at high impregnation rates. The key differentiating factor is that there is literally not enough

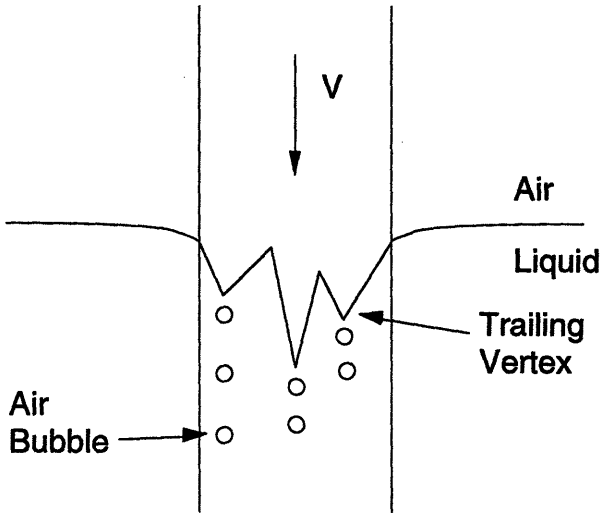


Figure 4.5 Void entrapment due to meniscus instability.

time for void mobilization in the high speed filament winding processes (i.e., the contact time between fibers and resin is too short). In the mold filling experiments, on the other hand, the fiber-resin contact time is relatively much longer, especially near the gates (i.e., resin inlets). If there were to be a void mobilization problem it would most likely occur near the spots farthest from the gate (i.e., the spots to fill last) and only if the injection cycle is not allowed to proceed long enough to flush out the entrapped voids. For example, in [17,18] wetting experiments were performed by pulling glass yarns through a 2-inch thick layer of epoxy resin at rates up to 500 in/min which resulted in minimum contact times of approximately 0.25s. Comparison of this time to characteristic processing times in RTM that can range from 10-10^3s indicates the disparity of time scales between filament winding and mold filling types of processes.

4.2 Void Mobilization

Final void content within a part is dependent not only on the entrapment mechanisms that lead to void formation, but also on what happens to the voids once formed. Adapting the arguments in [22] to flow within an aligned fiber bed, consider the status of a void located between the two fibers shown in Figure 4.6. As indicated in the figure, the fibers are neither perfectly aligned nor straight. This fiber-scale variability creates a varying pore geometry that might be described by characteristic waist and throat (or constriction) diameters $2r_w$ and $2r_t$, respectively. Assuming slow, creeping flow along the fibers, one can approximate the fluid speed within the pore as

$$U = \left(\frac{Q}{A}\right) = \frac{k_e}{\mu}\left|\frac{dp}{dz}\right| \tag{4.26}$$

where k_e is the effective permeability of the pore. Because of the viscous action of the surrounding fluid, the void will flow in the direction of the dominant pressure gradient and will assume the tapered shape shown schematically in Figure 4.6. The radii of curvature of the two ends of the void are related to the dominant pressure gradient, $dp/dz \approx \Delta p/\ell$, in the pore since

$$\begin{aligned} \Delta p &= p_2 - p_1 \\ &\approx \left(p_{void} - \frac{2\gamma}{r_2}\right) - \left(p_{void} - \frac{2\gamma}{r_1}\right) \\ &\approx 2\gamma\left(\frac{1}{r_1} - \frac{1}{r_2}\right) \end{aligned} \tag{4.27}$$

where contact angle hysterisis has been neglected for the sake of clarity. Consequently, in order to move the void through the constriction of radius r_t, the local pressure gradient must be greater than a critical mobilization gradient, or

$$\left|\frac{dp}{dz}\right| \approx \frac{|\Delta p|}{\ell} > |\nabla p_m| \sim \frac{2\gamma}{\ell}\left(\frac{1}{r_2} - \frac{1}{r_1}\right)_{max} \tag{4.28}$$

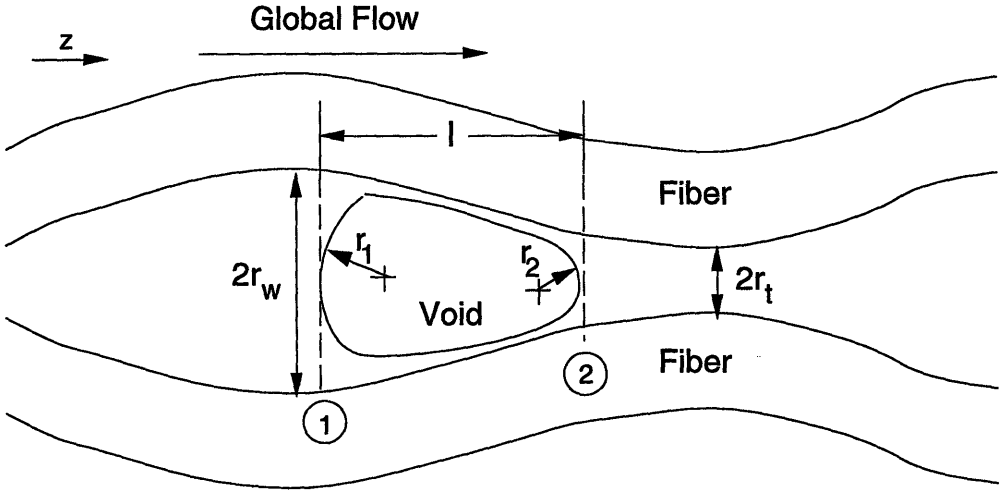


Figure 4.6 Schematic of assumed void mobilization geometry.

Noting that, in order to pass the void through the constriction, $r_2 < r_t$ and approximating the radius of the void surface at station 1 just prior to mobilization as $r_1 \leq r_w$, one can write

$$|\nabla p_m| \sim \frac{2\gamma}{\ell}\left(\frac{1}{r_t} - \frac{1}{r_w}\right) \tag{4.29}$$

Substituting Eqs 4.28 and 4.29 into Eq. 4.27, yields the condition on the local fluid velocity to mobilize the void

$$U > U_m \approx \frac{2k_e\gamma}{\mu\ell}\left(\frac{1}{r_t} - \frac{1}{r_w}\right) \tag{4.30}$$

or, rearranging terms and writing Eq. 4.30 in terms of the capillary number $Ca = \mu U/\gamma$,

$$Ca > Ca_m \approx \frac{2k_e}{\ell}\left(\frac{1}{r_t} - \frac{1}{r_w}\right) \tag{4.31}$$

The effective permeability, k_e, of the pore can be approximated by further simplifying the assumed pore geometry as a series of alternating cylindrical capillaries as shown in Figure 4.7. Assuming Poiseuille flow in the cylindrical capillary, one can immediately write

$$U = \left(\frac{Q}{A}\right) = \frac{r_w^2}{8\mu}\frac{dp}{dz} \approx \frac{r_w^2}{8\mu}\frac{\Delta p}{\ell} \tag{4.32}$$

and, therefore,

$$k_e \approx \frac{r_w^2}{8} \tag{4.33}$$

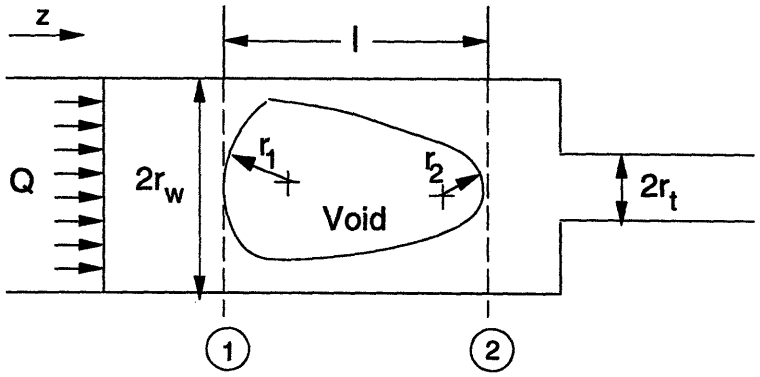


Figure 4.7 Simplified model of void mobilization geometry.

Substituting Eq. 4.33 into Eq. 4.31 and rearranging terms yields

$$Ca > Ca_m \approx \frac{1}{8}\frac{1}{(\ell/2r_w)}\left(\frac{r_w}{r_t}-1\right) \tag{4.34}$$

Estimating r_w and r_t in an aligned fiber bed with the appropriate hydraulic diameter (noting that $2r_w$ is equal to the waist diameter and $D_h = 4\times$ hydraulic radius) as given by either Eq. 4.7 or 4.9, one obtains

$$Ca > Ca_m \approx \frac{1}{8}\frac{1}{(\ell/D_h^w)}\zeta \tag{4.35}$$

where

$$\zeta = \frac{D_h^w}{D_h^t}-1 \tag{4.36}$$

can be thought of as a parameter characterizing the variability of the pore space. Equation 4.35 shows that mobilizing a void of length ℓ is dependent on the relationship between the characteristic axial and transverse dimensions (ℓ and the hydraulic diameter D_h^w) and some measure of the variability (the shape factor ζ) within the preform.

For flow along the fiber axis, ζ can be approximated using Eq. 4.7 to obtain

$$\zeta_z = \frac{1-V_f^w}{V_f^w}\cdot\frac{V_f^t}{1-V_f^t}-1 \tag{4.37}$$

If one defines the average fiber volume fraction as $\overline{V}_f = \frac{1}{2}\left(V_f^w + V_f^t\right)$ and the deviation from the mean of the fiber volume fraction as $\delta V_f = \frac{1}{2}\left(V_f^t - V_f^w\right)$, one can rewrite Eq. 4.37 in terms of the average fiber volume fraction and the relative variation in fiber volume fraction $\delta V_f/\overline{V}_f$, or

$$\zeta_z = \frac{1-\overline{V}_f\left(1-\dfrac{\delta V_f}{\overline{V}_f}\right)}{1-\overline{V}_f\left(1+\dfrac{\delta V_f}{\overline{V}_f}\right)}\cdot\frac{1+\dfrac{\delta V_f}{\overline{V}_f}}{1-\dfrac{\delta V_f}{\overline{V}_f}}-1 \tag{4.38}$$

Although Eq. 4.38 appears cumbersome, it has the advantage of being expressed in measurable terms. The average fiber volume fraction can be found by measuring the

density of a representative sample and the variation in fiber volume fraction can be estimated by a variety of means.

In the case of flow transverse to the fiber axis (as in radial flow in a fiber bundle), one can estimate ζ by approximating the waist dimension with the hydraulic diameter for axial flow in aligned fibers (Eq. 4.7) and the throat dimension with the hydraulic diameter for two parallel plates separated by a distance equal to the inter-fiber gap (Eq. 4.9). This approach yields

$$\zeta_r = \frac{\left(\frac{1-V_f}{V_f}\right)}{2\left[\left(\frac{V_a}{V_f}\right)^{1/2} - 1\right]} - 1 \tag{4.39}$$

since the hydraulic diameter for flow between two parallel plates is equal to twice the plate separation [2].

Having defined the shape factors ζ for both axial and transverse flow, we can return to Eq. 4.35 and rewrite it in a more useable form. Note that in both the axial and transverse flow cases, D_h^w can be approximated using the axial hydraulic diameter for the bundle given by Eq. 4.7. Substituting into Eq. 4.35 yields

$$Ca_m\left(\ell/d_f\right) \approx \frac{1}{\ell/d_f}\left[\frac{V_f\zeta}{8\left(1-V_f\right)}\right] \tag{4.40}$$

where d_f is the nominal fiber diameter. Equation 4.40 indicates that mobilization of a trapped void in the preform depends on the length of the void (normalized by the fiber diameter in Eq. 4.40), the fiber volume fraction, and on the variability of the pore structure (expressed as ζ in Eq. 4.40). For a given pore structure, it should be easier to flush out large (i.e., long) voids than small (i.e., short) voids and, in agreement with intuition, for a perfectly uniform pore structure $(\zeta = 0)$ the requirement for void mobilization is any arbitrarily small, non-zero flow.

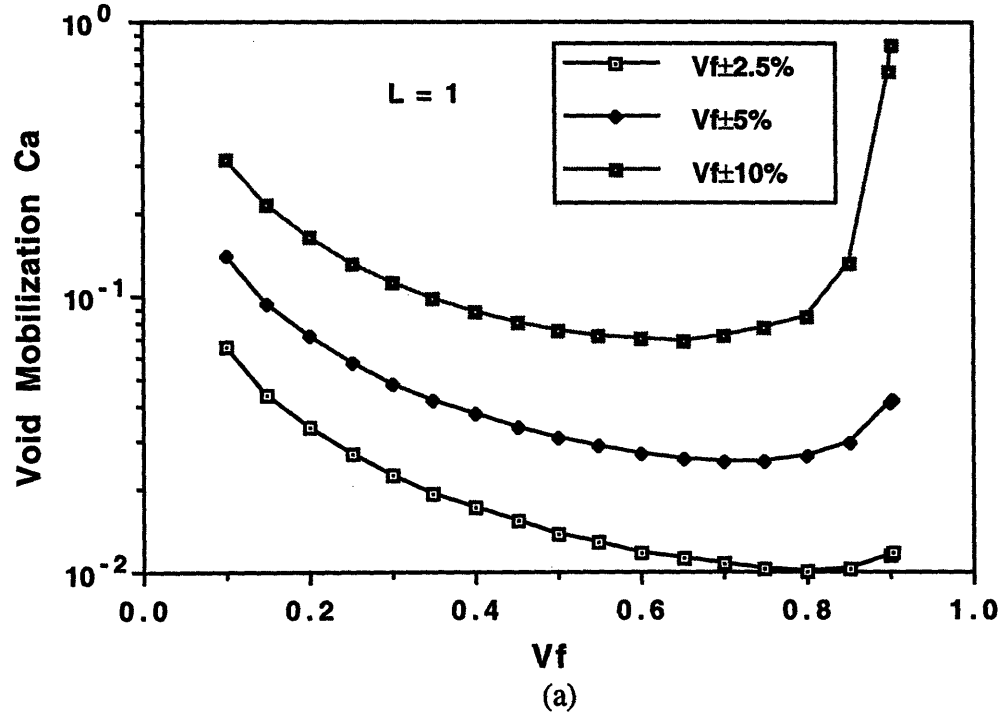
From a practical standpoint, Eq. 4.40 suggests a strategy for producing impregnated composites with low void content. It has long been known that the resistance to flow within the preform will decrease if the fiber volume fraction is decreased (i.e., the permeability will be increased. See, for example, the discussion in [3] on the Kozeny-Carmen equation.). This enables either a reduction in cycle time because of faster impregnation rates or reduced force (i.e., pressure) requirements enabling use of less

expensive machinery for a given cycle time. As discussed in Section 4.1.2, air entrapment due to meniscus instability can impose an upper limit on wetting rates. As discussed in this section, these (and voids formed during "slow" wetting in the sense of Section 4.1.1) can be flushed out of the preform if the local capillary number can be increased above the mobilization value indicated in Eq. 4.40. A process wherein the impregnation step is performed in a loosely packed preform, as in filling the preform before closing the tooling and fully compressing it, would therefore benefit from both the higher permeability and the easier void flushing characteristics of a low-V_f perform.

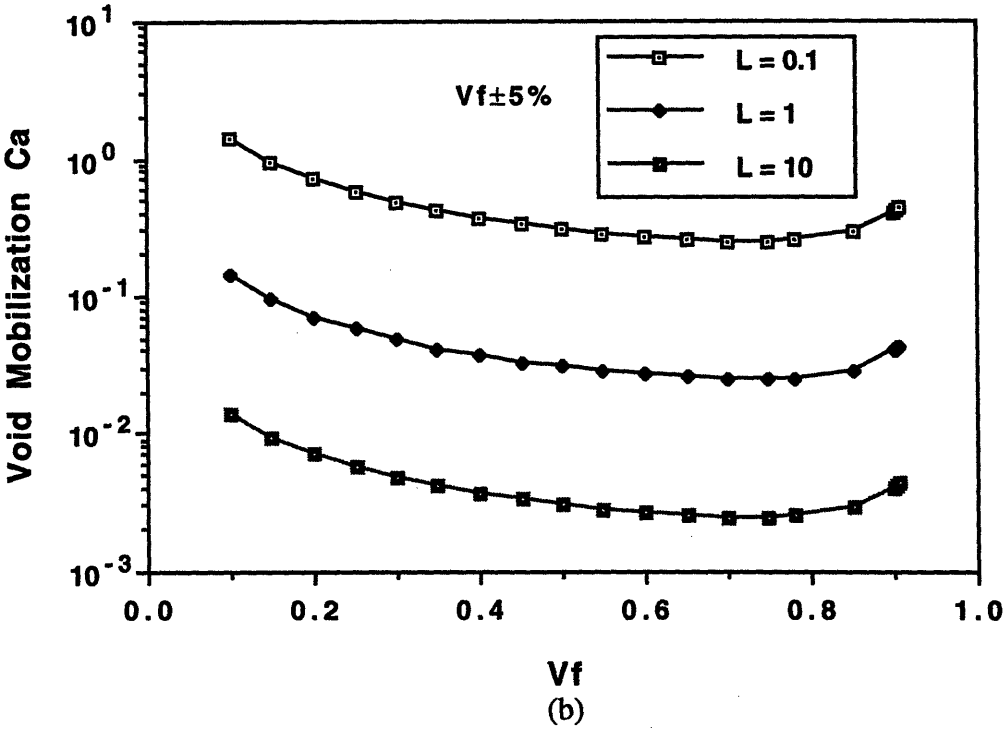
This, in fact, is the basis for a new RTM-like process being introduced in automotive applications [23-25]. The process is a combination of RTM and compression molding. The preform is typically a random fabric. The resin is injected at elevated temperatures, further decreasing injection time by the accompanying viscosity reduction, and the injection step is performed as the tooling is being closed. The tool is closed only for the curing step of the process. Injection times of 3-4 s and cycle times on the order of minutes are claimed with fast-curing resins.

Equation 4.38 indicates that for a given amount of relative axial variability, $\delta V_f/\overline{V}_f$, within a fabric ζ_z will approach some finite limiting value (when adjacent fibers come into contact, or $V_f/V_a = 1$) and, therefore, so will Ca_m. On the other hand, Eq. 4.39 shows that ζ_r increases without bound as the fiber volume fraction approaches the maximum available fiber fraction for the tow or (in the case of transverse flow) fabric. Consequently, it will become increasing difficult to achieve Ca_m for flows transverse to the fibers as the fiber volume fraction increases. Figures 4.8 and 4.9 summarize the behavior of the void mobilization criterion, Ca_m, for both axial and trnasverse flows, respectively.

As indicated in Eq. 4.38, the axial shape factor ζ_z and, therefore, the axial Ca_m is not directly dependent on the packing structure of the fibers (described via the available fiber volume fraction, V_a). Consequently, only one set of curves such as Figure 4.8(a) are required to describe the axial mobilization criterion for a void of given length (expressed in terms of the waist hydraulic diameter as in Eq. 4.35 and in terms of the fiber diameter in Eq. 4.40). The only consequence of packing structure relative to the curves of Figure 4.8(a) is how far to the right one reads the graph. Of far greater importance in determining the axial void mobilization is the amount of axial variability present within the fiber bed.
In contrast to the packing structure independence of axial mobilization, the transverse case is much more sensitive to the value of V_a (0.785 and 0.907 for square and hexagonal packing, respectively). This dependence is illustrated in Figure 4.9(a) and can be seen explicitly in Eq. 4.31. The significance of the packing structure stems from the dependence of the fiber separation at a given V_f to the ratio V_f/V_a. Comparing Figures 4.8(b) and



(a)



(b)

Figure 4.8 Axial void mobilization criterion: (a) dependence on relative axial variability in fiber volume fraction, (b) dependence on void length (normalized to waist hydraulic diameter).

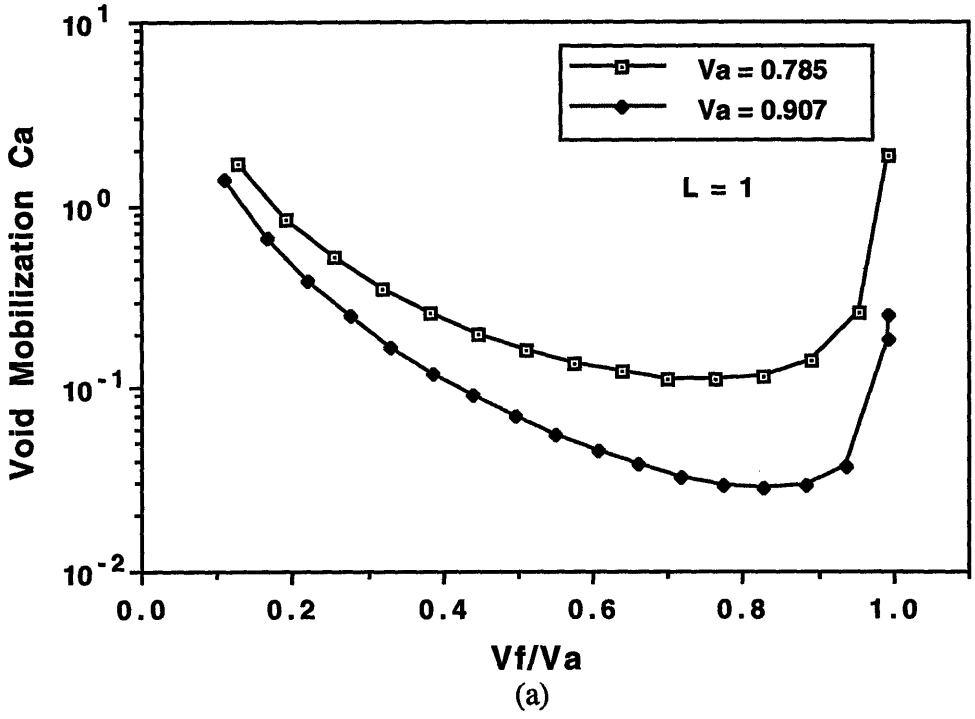


Figure 4.9 Transverse void mobilization criterion: (a) dependence on packing structure, (b) dependence on void length (normalized to waist hydraulic diameter).

4.9(b), one can see that the transverse void mobilization is much more sensitive to void length than the axial case. Note that, in addition to the sensitivity to void length illustrated in Figure 4.9(b), transverse void mobilization will often be more difficult because of the elliptical shape of most voids within the fiber bundle (i.e., most of these voids are longer in the axial rather than the transverse direction). Consequently, axial mobilization is typically easier to achieve than transverse mobilization.

In a real composite, the variability of the preform pore space (described, for example, by some random probability distribution for ζ) leads to formation of a number of voids of various sizes and, therefore, various values for the mobilization criterion, Ca_m. An ad hoc approach to dealing with this uncertainty might involve making conservative estimates of the critical capillary number for void mobilization, Ca_m, based on void length, average preform fiber volume fraction, and expected preform variability.

Another approach to handling the random nature of the preform and the behavior of a large population of voids is the use of percolation theory and network models [15,26-28] to describe soil flows. A detailed description of percolation theory is beyond the scope of this thesis, however, it has been used with some success in modeling relative permeabilities and capillary pressure-saturation relationships in fluid-fluid displacement flows. Of particular interest to the analysis of void formation in advanced composites processing, however, are predictions of residual non-wetting phase saturations (i.e., final void content) made using percolation theory. As reported in [26,27], experiments and calculations show that values of $Ca > 10^{-2}$ will result in void contents approaching zero in a wide variety of soils with various microstructure.

Results from the study of void content in random fabrics [29,30] are seen to agree with the residual, non-wetting phase saturation behavior predicted using percolation theory. The percolation theory results on random networks were shown to succesfully correlate displacement processes in soil samples. Percolation theory worked with the soil samples because soil is a "random" medium. For the same reasons, the random fabrics in [29,30] are also "random". It should come as no surprise, therefore, that the results in [29,30] match the percolation theory predictions. It should be stressed that experiments in [29,30] did not include observation of void formation/entrapment. Instead, the authors measured and reported final void content as a function of distance from fluid injection point and therefore, indirectly, fluid velocity. Although the connection was not made, the results in [29,30] confirm the applicability of percolation theory to prediction of void mobilization in composites processing in random fabrics.

4.3 References

1. Parnas, R., and Phelan, F., 1991, "The Effect of Heterogeneous Porous Media on Mold Filling in Resin Transfer Molding," *SAMPE Quarterly*, **22**(2):53-60.

2. White, F., 1986, *Fluid Mechanics*, 2nd ed., McGraw-Hill Book Co., New York.

3. Bear, J., 1972, *Dynamics of Fluids in porous Media*, Ch. 9, Dover, New York.

4. Hoffman, R., 1975, "A Study of the Advancing Interface: I. Interface Shape in Liquid-Gas Systems", *Journal of Colloid and Interface Science*, **50**(2):228-241.

5. de Gennes, P., 1985, "Wetting: Statics and Dynamics", *Rev. Mod. Phys.*, 57(3), Part I:827-863.

6. Molnar, J. Trevino, L., and Lee, L., 1989, "Mold Filling in Structural RIM and Resin Transfer Molding" *Proceedings of the 44th Annual Conference, Composites Institute, The Society of the Plastics Industry.*

7. Fong, L., and Lee, L., 1994, "Preforming Analysis of Thermoformable FiberMats--Preforming Effects on Mold Filling", *Journal of Reinforced Plastics and Composites*, **13**:637-663.

8. Patel, N., Lee, L., Young, W., and Liou, M., 1991, "Resin-Fiber Wetting and Bonding in Resin Transfer Molding and Structural Reaction Injection Molding," in *Proceedings of the SPE 49th Annual Technical Conference ANTEC '91*, pp. 1985-1990.

9. Patel, N., Rohatgi, V., and Lee, L., 1993, "Influence of Processing and Material Variables on Resin-Fiber Interface in Liquid Composite Molding", Polymer Composites, 14(2):161-172.

10. Patel, N., Han, K., and Lee, L., 1993, "Micro-Mechanics Study in Resin Transfer Molding", *Proceedings of NSF Design and Manufacturing Grantees Conference*, pp. 257-263.

11. Patel, N., and Lee, L., 1995, "Effects of Fiber Mat Architecture on Void Formation and Removal in Liquid Composite Molding", *Polymer Composites*, **16**(5):386-399.

12. Princen, H., 1969, "Capillary Phenomena in Assemblies of Parallel Cylinders. I. Capillary Rise Between Two Cylinders", *Journal of Colloid and Interface Science*, **30**(1):69-75.

13. Princen, H., 1969, "Capillary Phenomena in assemblies of Parallel Cylinders. II. Capillary Rise in Systems with More Than Two Cylinders", *Journal of Colloid and Interface Science*, **30**(3):359-371.

14. Princen, H., 1970, "Capillary Phenomena in assemblies of Parallel Cylinders. III. Liquid Columns Between Horizontal Parallel Cylinders", *Journal of Colloid and Interface Science*, **34**(2):171-184.

15. Wilkinson, D., 1986, "Percolation Effects in Immiscible Displacement", *Physical Review A*, **34**(2):1380-1391.

16. Dussan V., E., 1979, "On the Spreading of Liquids on Solid Surfaces: Static and Dynamic Contact Lines", *Annual Review of Fluid Mechanics*, **11**:371-400.

17. Bascom, W., 1965, "Some Surface Chemical Aspects of Glass-Resin Composites. Part I: Wetting Behavior of Epoxy Resins on Glass Filaments", presented at the 20th Annual Meeting of the Reinforced Plastics Division of Society of the Plastics Industry, Chicago, Ill.

18. Bascom, W., and Roman, J., 1968, "Microvoids in Glass-Resin Composites", *Industrial and Engineering Chemistry Product Research and Development*, **7**(3):172-178.

19. Inverarity, G., 1969, "Dynamic Wetting of Glass Fibre and Polymer Fibre", *Br. Polym. Journal*, **1**:245-251.

20. Burley, R., and Kennedy, B., 1976, "An Experimental Study of Air Entrainment at a Solid/Liquid/Gas Interface", *Chemical Engineering Science*, **31**:901-911.

21. Blake, T., and Ruschak, K., 1979, "A Maximum Speed of Wetting", *Nature*, **282**(29):489-491.

22. Ng, K., Davis, H., and Scriven, L., 1978, "Visualization of Blob Mechanics in Flow Through Porous Media", *Chemical Engineering Science*, **33**:1009-1014.

23. ---, 1993, "One-minute RTM cycles are here", *Plastics Technology*, **39**(3):11.

24. Kim, Y., Hwang, K., and Lee, J., 1995, "Effects of Initial Fiber Volume and Mold Closing Speed During Advanced RTM Processes", in *Proceedings of the 50th Annual Conference, Composites Institute, The Society of the Plastics Industry,* January 30-February 1, 1995, Session 3-D.

25. Ikegawa, N., Hamada, H., and Maekawa, Z., 1996, "Effect of Compression Process on Void Behavior in Structural Resin Transfer Molding", *Polymer Engineering and Science*, **36**(7):953-962.

26. Larson, R., Scriven, L., and Davis, H., 1977, "Percolation Theory of Residual Phases on Porous Media", *Nature*, **268**(4):409-413.

27. Larson, R., Davis, H., and Scriven, L., 1981, "Displacement of Residual Nonwetting Fluid From Porous Media", *Chemical Engineering Science*, **36**:75-85.

28. Levine, S., Reed, P., and Shutts, G., 1977, "Some Aspects of Wetting/Dewetting of a Porous Medium", *Powder Technology*, **17**:163-181.

29. Mahale, A., Prud'Homme, R., and Rebenfeld, L., 1992, "Quantitative Measurement of Voids Formed During Liquid Impregnation of Nonwoven Multifilament Glass Networks Using an Optical Visualization Technique," *Polymer Engineering and Science*, **32**(5):319-326.

30. Mahale, A., Prud'Homme, R., and Rebenfeld, L., 1993, "Characterization of Voids Formed During Liquid Impregnation of Non-Woven Multifilament Glass Networks as Related to Composite Processing", *Composites Manufacturing*, **4**(4):199-204.

5 FORMULATION OF COMPUTER MODEL

One of the goals of this thesis has been to develop an analytical tool that can approximately model quality-determining phenomena at the tow scale using estimates of the preform microstructure. Figure 5.1 summarizes the basic solution algorithm incorporated into the simulation. A listing of the source code, written in ANSI standard C, is located in Appendix C. (Note that the software consists of two main parts: a set of utility routines for manipulating the various data structures in the program labeled *ops.c and the "working" parts of the solution algorithm outlined in Figure 5.1.)

5.1 Modeling Assumptions

The underlying assumption of the model is that the inter-fiber pore space can be described as a network of n finite control volumes and that the flow between control volumes can be approximated by a series of steady-state, one-dimensional, creeping flows. The pressure within each control volume is assumed constant and is assigned to a node located at the center of each control volume, resulting in a non-uniform finite difference grid. The general geometry of the model is shown in Figure 5.2. Other assumptions of the model are summarized below:

- The process is modeled as an immiscible displacement process. The invading fluid ("resin") is incompressible and the displaced fluid ("air") is stationary.
- The resin is a Newtonian fluid (i.e., the viscosity, μ, is constant).
- The interfacial pressure difference between the invading and displaced fluids is given by the Young-Laplace equation. This corresponds to assuming meniscus equilibrium at the interface.
- Each node i corresponds to a control volume with $V_i = A_z^i \cdot \Delta z$ and the model consists of n_z layers in the axial, or z, direction.
- The entries $h_{i,j}$ of the conductance matrix **H** are assumed constant across each interface (i,j). This implies that that there is no significant fiber motion during the process and, as noted above, that the resin is a Newtonian fluid.
- Voids are formed by entrapment of air behind the global flow front.
- The conductance across the interface between node i and a neighboring void node v is "shut off" (i.e., $h_{i,v} = 0$). The saturation s_v of the void node is consequently held constant once it is incorporated into the void.

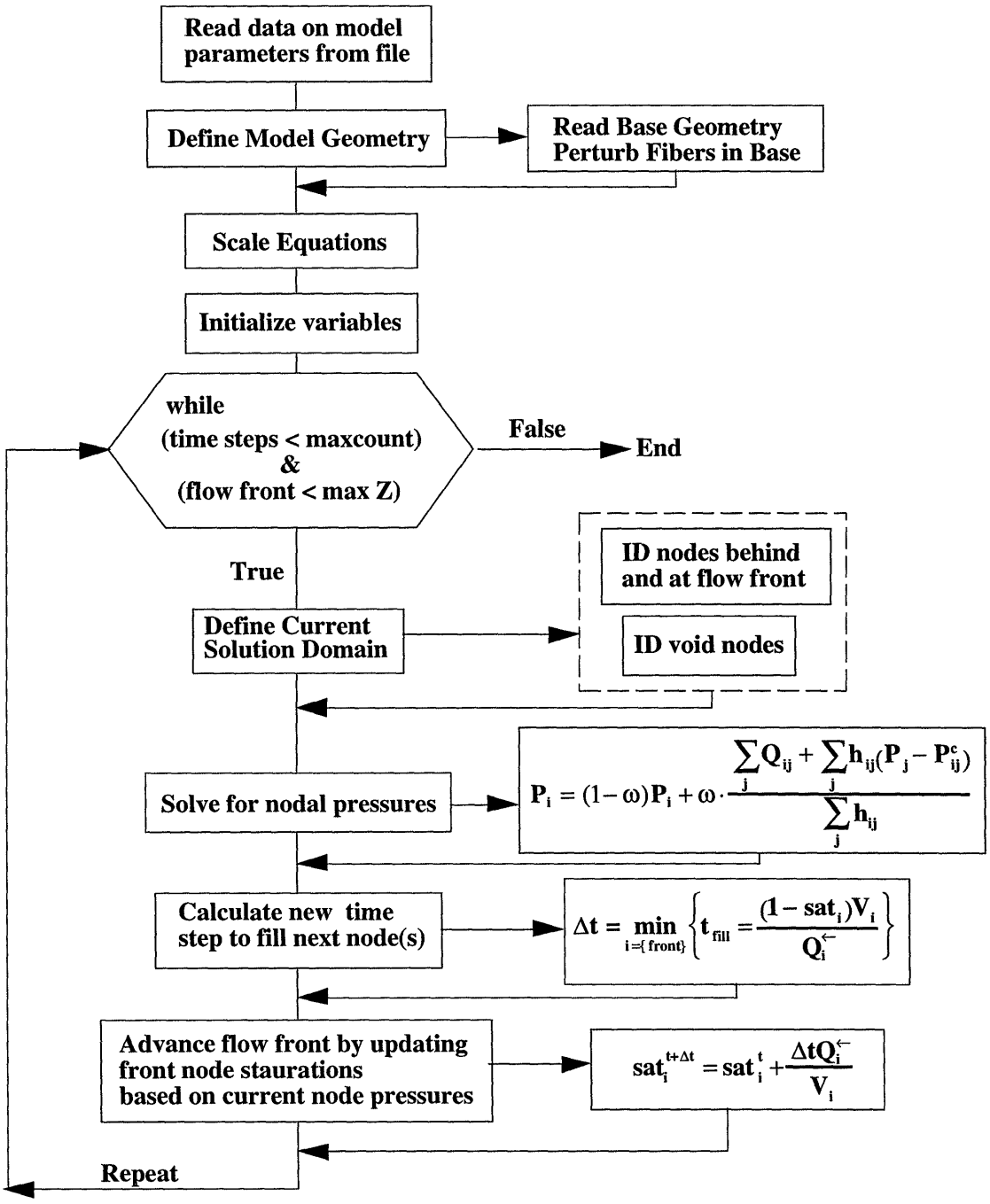


Figure 5.1 Simulation Flow Chart

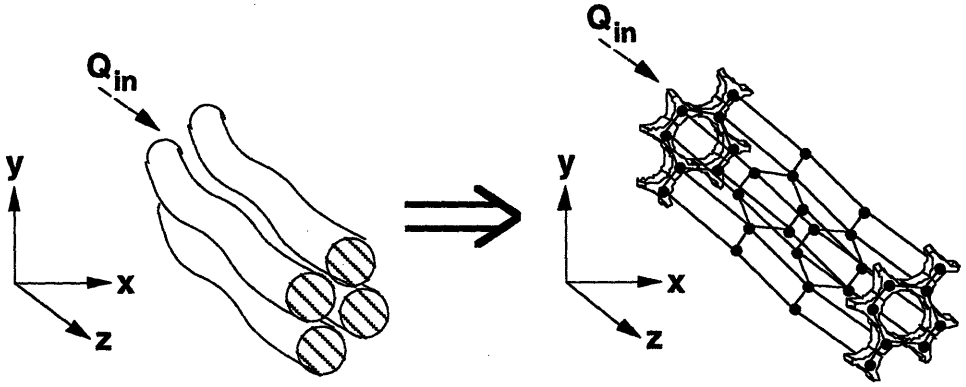


Figure 5.2 Assumed problem geometry .

Using the above assumptions, a mass balance on a typical node i, shown in Figure 5.3, yields

$$\sum_j Q_{i,j} = 0 \tag{5.1}$$

The mass flow rate between nodes i and j is given by

$$Q_{i,j} = h_{i,j}\left(P_i - P_j\right) \tag{5.2}$$

where calculation of the values of $h_{i,j}$ is discussed in Section 5.5. Combining Eqs 5.1 and 5.2 yields

$$\sum_j h_{i,j}\left(P_i - P_j\right) = 0 \tag{5.3}$$

Applying Eq. 5.3 to each of the nodes in the network results in a system of n simultaneous equations

$$\mathbf{H}P = Q \tag{5.4}$$

where $\mathbf{H}$ is an $(n \times n)$ matrix and P and Q are both $(n \times 1)$ vectors. Solution of Eq. 5.4 is greatly simplified by the fact that $\mathbf{H}$ is typically positive definite, symmetric and very sparse (although, due to node connectivity in three-dimensional problems, it still can have a very broad bandwidth). If each node has σ neighbors (i.e., adjacent nodes), $\mathbf{H}$ will contain, at most, only $n(\sigma+1)$ non-zero elements. (The equality is strict for the case of a completely

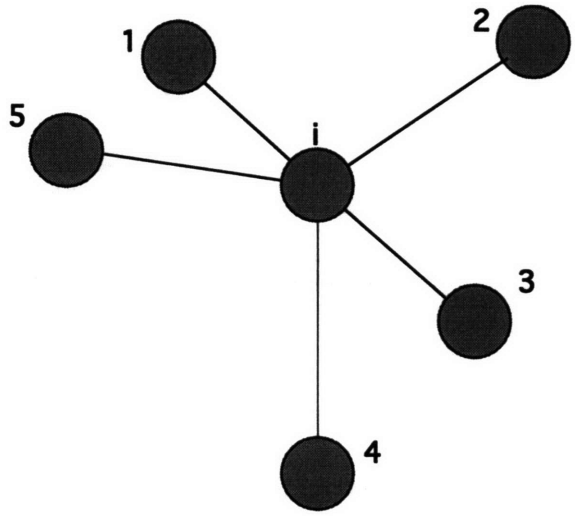


Figure 5.3 Schematic of computaional cell for node i.

saturated network.) Consequently, Eq. 5.3 is a prime candidate for solution using an iterative technique such as relaxation (Gauss-Seidel and successive over-relaxation are two well-known examples).

5.2 Boundary Conditions

The boundary conditions assumed in the simulation are a known flux, Q_{in}^{s}, at the inlet node (at $z = 0$) and an interfacial pressure difference $P_{i,j}^{c}$ at the fluid front between nodes i and j defined by the Young-Laplace equation

$$P_{i,j}^{c} = \frac{2\gamma \cos\theta_{i,j}}{R_{c}^{i,j}} \tag{5.5}$$

where γ, $R_{c}^{i,j}/\cos\theta_{i,j}$, and $\theta_{i,j}$ are the surface tension, radius of curvature, and contact angle at the interface between nodes i and j. As discussed in Section 3.?, the contact angle is dependent on the local capillary number, which will be defined here as

$$Ca_{i,j} = \frac{\mu U_{i,j}}{\gamma} = \frac{\mu}{\gamma}\left(\frac{Q_{i,j}}{A_{i,j}}\right) \tag{5.6}$$

In order to simplify the calculations, however, it was assumed that the dynamic contact angle was approximately equal to the equilibrium value. As discussed in Section 4.1, this is a reasonable approximation for $Ca_{i,j} << 1$.

5.3 Definition of Model Geometry

In order to generate a physically realistic model of the tow pore space, it is necessary to also model the packing of the fibers that make up the tow. Consequently, definition of the model geometry is done in two steps describing the fibers and then the inter-fiber pore space.

5.3.1 Fiber Placement

Fiber "placement" in the model is done by defining the $x,y,z-$coordinates of the fiber centers assuming that $x = x(z)$ and $y = y(z)$. The dependence of (x,y) on z is a user-defined function perturbing the fiber positions in a reference plane at $z = 0$. Figures 5.4 and 5.5 schematically illustrate the nomenclature and four possible sets of assumptions for the dependence of (x,y) on z.

Figure 5.5(a) represents the ideal geometry achieved with perfectly aligned fibers. Although the fiber separations are uniform and constant with z, the pore space is still non-uniform because, as shown in Figure 5.4(b), packing of circular cylinders creates two types of regions, fairly large pores formed by groups of three or more fibers and the fairly small regions within the immediate neighborhood of the gap between two adjacent fibers. These regions will be referred to as "pores" and "gaps", respectively.

Figure 5.5(b) illustrates the effect of randomly perturbing the fiber centers along the z-axis. Random networks are often used to model natural porous media formed by packing of random particles, such as soils [1-4]. For modeling the pore space formed in an aligned fiber bed, however, a truly random network can be problematic if one tries to limit the model to physically consistent realizations. Of particular concern is the potential for forcing a model fiber into a bend with such a small radius of curvature that the resulting stress on the fiber's outer surface would cause it to break (an analogous problem exists in sheet metal forming).

Figures 5.5(c) and 5.5(d) illustrate another approach to modeling the random pore space variations in the fiber bed. The basic idea is to assume that the fibers are not perfectly straight, as in Figure 5.5(a), but that they can be described as a bundle of waves each with a characteristic wavelength, L_i. If the characteristic L_i's are all equal (or nearly so) to a single value L that is representative of the entire wave bundle and all of the fibers are perfectly aligned and "in phase", then the bundle would look like Figure 5.5(c). The net result would be that the in-plane fiber separations would remain constant along z, but

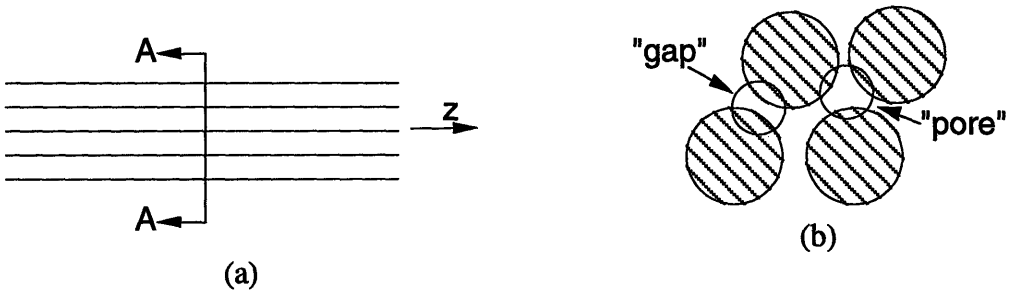


Figure 5.4 Nomenclature for fiber placement: (a) fiber bundle, (b) section A-A showing resulting variation due to packing of round fibers.

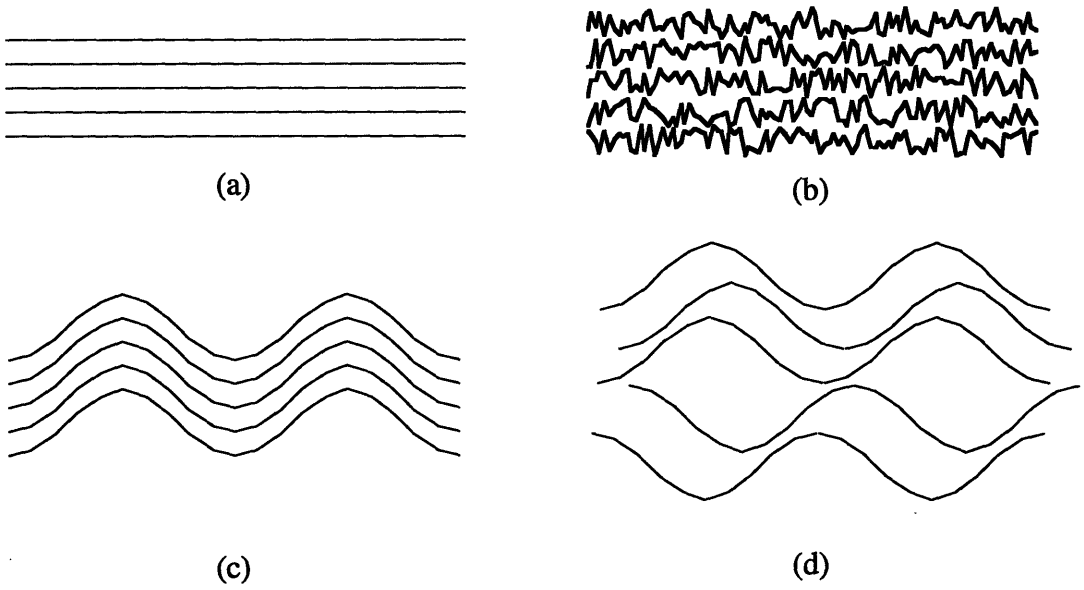


Figure 5.5 Assumptions of fiber bundle variability: (a) ideal packing, no variation, (b) random variation, (c) uniform fiber waviness, (d) uniform fiber waviness with non-uniform phase shift added to model fiber misalignment.

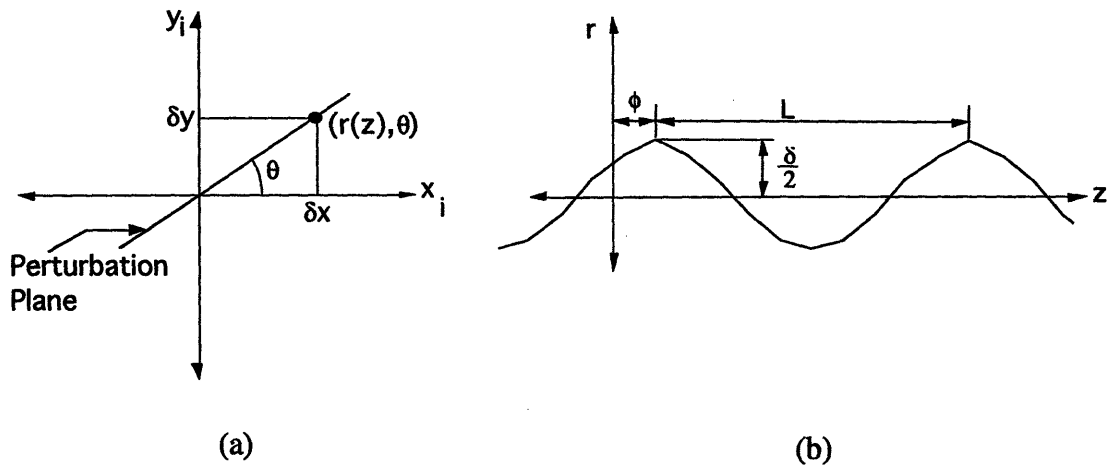


Figure 5.6 Assumed geometry for periodic perturbation of fiber center coordinates.

the flow path would become more tortuous. Although an effective bundle permeability might be calculated if one were to introduce a tortuosity factor as discussed in [5], one would not be able to examine void formation and mobilization because of the lack of variability along z. This problem can be overcome if one models the misalignment of the fibers when brought together in the bundle. In terms of the wave analogy, one can both rotate the orientation of the wave (all assumed co-planar in Figure 5.5) and add a random phase shift to each fiber. The co-planar case is illustrated in Figure 5.5(d). Note that the resulting pore space varies with the same wavelength as the fiber perturbation.

The procedure for periodic perturbation of the fiber centers is as follows. Figure 5.6 defines the nomenclature. The fiber positions within the reference plane are currently defined off-line and read from a file into a linked list. All linear dimensions are normalized by the nominal fiber diameter, d_{nom}. A set of local coordinates (x_i, y_i) is defined for each fiber i with origin at the fiber center (x_i^o, y_i^o) in the reference plane at $z = 0$. At each z every fiber is visited in the linked list and its center is shifted by the trial amounts

$$\delta x_i(z) = r_i(z)\cos\theta_i \tag{5.7a}$$

and

$$\delta y_i(z) = r_i(z)\sin\theta_i \tag{5.7b}$$

where θ_i is the orientation of the perturbation plane for fiber i. As indicated in Figure 5.6, the perturbation plane is the plane containing the fiber's "wave" profile, given by

$$r_i(z) = \varepsilon_i \frac{\bar{\delta}_o}{2}\left[1 - \cos\left(\frac{2\pi z}{L} + \phi_i\right)\right] \tag{5.8}$$

where $\bar{\delta}_o$ is the average inter-fiber separation in the reference plane, ϕ_i is the phase shift for fiber i, and

$$\varepsilon_i = \begin{cases} 0 & \text{ideal packing} \\ \varepsilon_o & \text{initial trial, no overlap} \\ rand(\,) & \text{if overlap} \end{cases} \tag{5.9}$$

As indicated in Eq. 5.9, ε_i is set to its nominal value ε_o for the initial trial perturbation. The in-plane distances from fiber i to the previously perturbed fibers at the same z are calculated. If any of the distances is negative, then there is an overlap of fibers and ε_i is set to a random value such that $0 \le \varepsilon_i < \varepsilon_o$, $r_i(z)$ is recalculated, and the previously perturbed fibers are checked again for overlap. The process iterates for fiber i until there is no overlapping and then proceeds to the next fiber in the linked list. Once all the fibers at a given z are successfully perturbed, the process is repeated at the subsequent values of z.

Selection of a value for the characteristic fiber wavelength L is based on the model proposed in [6,7] for the analysis of a bundle of aligned, lubricated fibers. Although the fibers in an RTM preform are nominally dry, the basic compression behavior of dry fiber beds [8,9] is similar enough to that reported in [6,7] that the model for lubricated fibers was applied as an approximation. Consequently, the fiber wavelength L was assumed to be proportional to the average fiber separation in the reference plane $\bar{\delta}_o$, or

$$\begin{aligned} L &= \beta \bar{\delta}_o \\ &= \beta\left[\left(\frac{\bar{V}_a}{V_f}\right)^{1/2} - 1\right] \end{aligned} \tag{5.10}$$

where, as in [6,7], β is the constant of proportionality and is assumed to be constant for a particular fiber type. The variable $\bar{V}_a$ represents an average value of the available fiber volume fraction in the reference plane. Details of the calculation of $\bar{V}_a$ will be discussed in Section 5.3.2 since a similar procedure is also performed when defining the control volumes.

In addition to perturbations of the fiber positions, the simulation incorporates the possibility for variability in fiber diameter as a function of z. Currently, this is modeled as

a random perturbation of the values (not necessarily equal) of the fiber diameters in the reference plane which are read in from the same file as the reference fiber coordinates.

5.3.2 Definition of Control Volumes

Once the fibers have been "placed" in the model, it is possible to define the resulting control volumes and their associated nodes for use in the flow simulation. Figure 5.7 illustrates the nomenclature. As in Figure 5.4(b), these control volumes will be referred to as pores and gaps. In addition, the terms control volume and element will be used interchangeably. For our purposes, definition of the control volumes can be summarized as determining the following parameters

- Global coordinates of control volume centroid. These coordinates locate the associated nodes and, therefore, define the grid for the finite difference solution of the governing equation.
- Fiber volume fraction. In addition to providing statistics on the effects of varying fiber packing assumptions as discussed in Section 5.3.1, knowledge of the local fiber volume fraction enables calculation of the volume available for the fluid and, therefore, calculation of the approriate control volume interface areas and hydraulic radii. In addition, knowledge of the local (i.e., cell) fiber volume fraction enables characterization of the local fiber packing efficiency, characterized by the ratio of the local fiber volume fraction to the available fiber volume fraction V_f^i/V_a^i.
- Volume. This is information that will be necessary to update control volume saturations and, therefore, define flow front location during simulation.

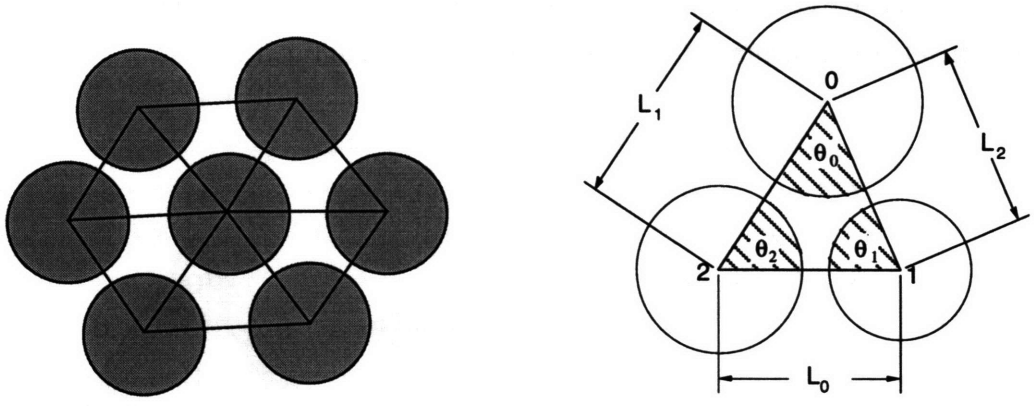


Figure 5.7 Generation of control volumes: (a) schematic of triangular cells in portion of fiber bundle, (b) nomenclature for analysis of an arbitrary unit cell.

As with fiber placement, control volume definition begins by calculating the various parameters that are defined within the (x,y)-plane defined by $z = k \cdot \Delta z$ where k is an integer ranging from $0 \le k < n_z$. Calculation of the parameters describing the interfaces between adjacent z -planes are then performed based on volume-averaged values.

Location of Control Volume Centroid

As shown in Figure 5.8(b), there are four control volumes associated with the unit cell that is formed by three fibers that are of arbitrary size and location (i.e., the triangle formed by the three fibers is not equilateral as is the case in uniform close packing of identical fibers). The centroid of the pore-centered control volume will not correspond to the centroid of the triangular unit cell. As shown in Figure 5.8, the control volume cross-section can be represented as the sum of a number of component areas, or geometric "primitives". Noting the well-known relationship [10] for the centroid of a composite area $(\bar{x},\bar{y})$

$$\bar{x} = \frac{\sum_i A_i \bar{x}_i}{\sum_i A_i} \qquad \bar{y} = \frac{\sum_i A_i \bar{y}_i}{\sum_i A_i} \tag{5.11}$$

where the summations are carried out over all the component areas, the numerators correspond to the sum of the first moments of area for each of the components, and the denominators are equal to the cross-sectional (axial) area available to the fluid for the pore element in question. Note that, as shown in Figure 5.8, all but the triangular cell areas are negative (i.e., they must be subtracted in order to produce the required shape). In order to avoid confusion and to maintain a certain notational aesthetic, the following discussion will refer specifically to the magnitudes of the "negative" areas in Eq. 5.11.

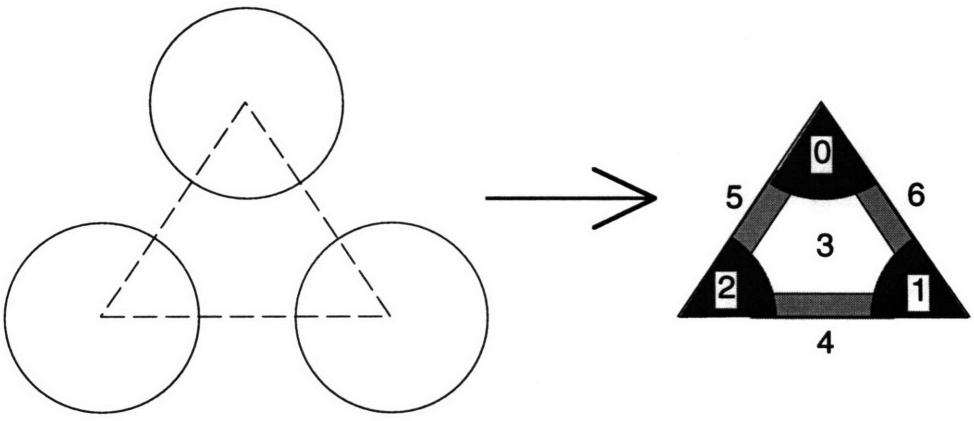


Figure 5.8 Decomposition of unit cell into component areas

Consider first the fiber sectors included in the unit cell. The area of each sector, $|A_i|$ where $i = 0,1,2$, is given by

$$|A_i| = \frac{\theta_i d_i^2}{8} \tag{5.12}$$

where θ_i is the included angle for fiber i shown in Figure 5.7(b) and d_i is the corresponding fiber diameter. The centroid of sector i lies on the line bisecting θ_i a distance [10]

$$\bar{r}_i = \frac{2d_i \sin(\theta_i/2)}{3\theta_i} \tag{5.13}$$

In order to determine the coordinates $(\bar{x}_i, \bar{y}_i)$ of the sector centroid, it is necessary to define the direction of the θ_i-bisector. This is done by noting that the desired direction corresponds to the direction of the resultant vector $\vec{R}$ obtained by adding the directed line segments $\vec{L}_{i,j}$ from vertex i to vertices $j \neq i$. Performing the necessary calculations yields

$$\vec{R} = R_x \hat{i} + R_y \hat{j} \tag{5.14}$$

where $(\hat{i}, \hat{j})$ are the unit vectors in the direction of the coordinate axes and

$$R_x = \sum_{j \neq i} (x_j - x_i) \qquad R_y = \sum_{j \neq i} (y_j - y_i) \tag{5.15}$$

Consequently, the unit vector $\hat{u}$ in the desired direction of $\vec{R}$ is

$$\begin{aligned} \hat{u} &= \frac{R_x}{|R|}\hat{i} + \frac{R_y}{|R|}\hat{j} \\ &= u_x \hat{i} + u_y \hat{j} \end{aligned} \tag{5.16}$$

Multiplying $\hat{u}$ by the centroid distance $\bar{r}_i$ yields the centroid coordinates $(\bar{x}_i, \bar{y}_i)$

$$\bar{x}_i = \bar{r}_i u_x \qquad \bar{y}_i = \bar{r}_i u_y \tag{5.17}$$

Next, consider the triangle with vertices corresponding to the center coordinates of fibers 0,1,2 in Figure 5.7(b). The coordinates of the triangle's centroid are $(\bar{x}_3, \bar{y}_3)$. The area of the triangle A_3 is given by [11]

$$A_3 = \frac{1}{2} Abs\left(\begin{vmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} \right) \tag{5.18}$$

where $Abs(\ )$ indicates absolute value and (x_i, y_i) refer to the coordinates of vertex i. The centroid of the triangle corresponds to the intersection of any two of the three bisectors of side L_i drawn from the opposite vertex i. Writing the equation for each of the bisectors yields

$$y = m_i(x - x_i) + y_i \tag{5.19}$$

where the slope m_i is given by

$$m_i = \frac{\bar{y}_{L_i} - y_i}{\bar{x}_{L_i} - x_i} \tag{5.20}$$

and $(\bar{x}_{L_i}, \bar{y}_{L_i})$ are the coordinates of the midpoint of side L_i. Solving for the intersection of any two bisectors from vertices a and b with finite m_i and non-zero difference $m_a - m_b$ yields

$$\bar{x}_3 = \frac{(m_a x_a - m_b x_b) + (y_b - y_a)}{m_a - m_b} \qquad \bar{y}_3 = \frac{m_a(y_b - m_b x_b) - m_b(y_a - m_a x_a)}{m_a - m_b} \tag{5.21}$$

Finally, consider the gap areas, $|A_i|$ where $i = 4,5,6$. Figure 5.9 defines the nomenclature for analysis of the gap region. As with the main triangular cell, the calculation of the gap area can be performed most efficiently by subtracting easily computed component areas from the larger region indicated in Figure 5.9. The first step requires selection of a width for the control volume corresponding to gap i. When the minimum fiber separation δ_o (i.e., the gap "height") of the gap element is $\delta_o < 0.30d$ the gap element width w is set equal to δ_o. For $\delta_o \geq 0.30d$, the gap element width is set arbitrarily to $w = 0.1\delta_o$, a "small" value. The key consideration is that the gap elements should be much smaller than the pore elements. This can come about quite "naturally" when the average fiber volume fraction V_f is high (corresponding to average $\delta_o/d << 1$) and the local variance of V_f is low. Selection of optimum values is left for future work.

Having defined the gap element width w, calculation of the required component areas is straightforward. Referring to Figure 5.9, the large area of region A, A_A, is given by

$$\begin{aligned} A_A &= wL_{1,2} \\ &= w\left[\delta_o + \tfrac{1}{2}(d_1 + d_2)\right] \end{aligned} \tag{5.22}$$

Calculation of areas A_B and A_C of the included sectors and the triangular fiber regions in Figure 5.9, repsectively, requires the value of the angles included by the sectors formed in each fiber by radii separated by the gap width w. This angle is seen to be

$$\alpha_i = 2\sin^{-1}(w/d_i) \tag{5.23}$$

for each fiber $i = 1,2$ in Figure 5.9. The total area of the included sectors, therefore, is seen to be

$$A_B = \frac{1}{8}\sum_{i=1}^{2} \alpha_i d_i^2 \tag{5.24}$$

Calculating the area of the two triangular regions straddling the included sector on fiber i yields

$$\begin{aligned} A_C^i &= (w/2)h_i \\ &= (w/4)d_i \cos(\alpha_i/2) \end{aligned} \tag{5.25}$$

and, therefore, the total area of the straddling regions C is

$$A_C = (w/4)\sum_{i=1}^{2} d_i \cos(\alpha_i/2) \tag{5.26}$$

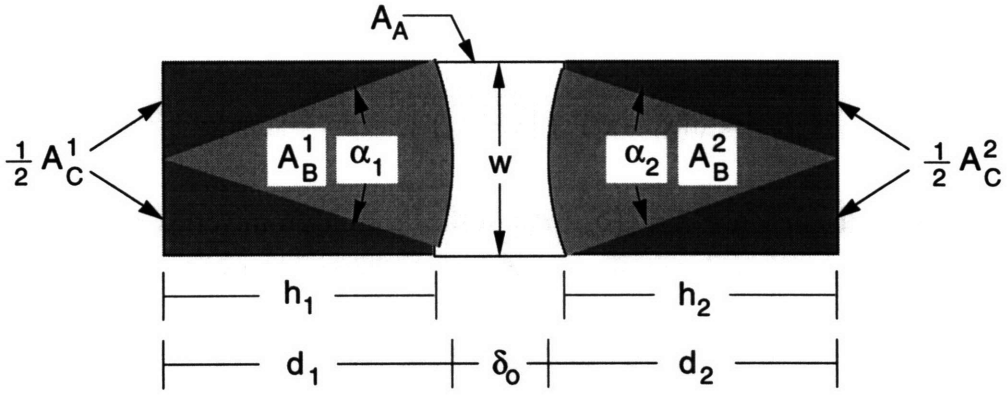


Figure 5.9 Nomenclature for gap area analysis

The gap area A_i^{gap} is now seen to be

$$\begin{aligned} A_i^{gap} &= A_C - (A_B + A_c) \\ &= w\left[\delta_o + \tfrac{1}{2}(d_1 + d_2)\right] - \frac{1}{8}\sum_{i=1}^{2}\alpha_i d_i^2 - (w/4)\sum_{i=1}^{2} d_i \cos(\alpha_i/2) \end{aligned} \tag{5.27}$$

for the general case where $d_1 \neq d_2$. As shown in Figure 5.8, only half of the gap element is seen to overlap with the triangular unit cell. Consequently, the appropriate value of A_i for use in Eq. 5.11 is

$$|A_i| = \frac{A_i^{gap}}{2} \tag{5.28}$$

In order to accommodate gap elements on the outer transverse boundaries of the model, the computer program calculates gap area parameters each time a given gap (defined by the two adjacent fibers 1 and 2 in Figure 5.9) is associated with a given pore (i.e., interior) element. Consequently, each gap region will be analyzed, at most, two times and the values of the gap element area and volume are updated after each analysis of the gap.

By virture of symmetry (and whether or not the gap is bounded on one or both sides by interior elements), the gap element centroid will lie in the center of the gap on the directed line segment joining the vertices of fibers 1 and 2, $\vec{L}_{1,2}$. Following a procedure similar to that for developing Eq. 5.17, one can define the unit vector $\hat{u}$ in the $\vec{L}_{1,2}$ direction

$$\begin{aligned} \hat{u} &= u_x \hat{i} + u_y \hat{j} \\ &= \frac{(x_2 - x_1)}{|L_{1,2}|}\hat{i} + \frac{(y_2 - y_1)}{|L_{1,2}|}\hat{j} \end{aligned} \tag{5.29}$$

where

$$|L_{1,2}| = \left|\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\right| \tag{5.30}$$

Multiplying $\hat{u}$ by the distance $\bar{r} = \frac{1}{2}(\delta_o + d_1)$ of the centroid from vertex 1 and translating to the global coordinate axes, yields the gap element centroid coordinates

$$\bar{x}_i = x_1^c + \bar{r}u_x \quad \bar{y}_i = y_1^c + \bar{r}u_y \tag{5.31}$$

where (x_1^c, y_1^c) are the global coordinates of fiber 1 in Figure 5.9.

Having determined the areas and centroid global coordinates for all of the geometric primitives, all that remains to calculate the global coordinates of the pore element centroid is substitution of the appropriate equations for the various A_i and $(\bar{x}_i, \bar{y}_i)$ into Eq. 5.11. Determination of the global coordinates of the gap element centroids is done by evaluation of Eq. 5.30.

Fiber Volume Fraction

The fiber volume fraction for cell i is equal to the area ratio given by

$$V_f^i = \frac{A_f}{A_{cell}} = \frac{1}{A_3}\sum_{j=0}^{2}|A_j| \tag{5.32}$$

where the included area $|A_j|$ of fiber j is given by Eqn. 5.12 and the total cell area, $A_{cell} = A_3$, is given by Eqn. 5.18. Fiber volume fraction statistics calculated by the program are all based on the cell values given by Eq. 5.32.

A related parameter for cell i is the available fiber volume fraction, V_a^i. The ratio V_f^i/V_a^i can be thought of as a measure of the packing efficiency relative to the configuration of the fibers within the cell. In addition, to being a measure of the arrangement of fibers in the cell (and, by way of the appropriate averages, the cross-section and the entire bundle model), the reference plane average, $\overline{V}_a$, is needed to estimate the fiber perturbation wavelength in Eq. 5.10.

For uniformly spaced fibers of the same diameter, d, the minimum gap, $\delta_o^{j,k}$, separating fibers (j,k) in the cell is the same for all three pairs and is given by

$$\delta_o = \delta_o^{j,k} = d\left[\left(\frac{V_a^i}{V_f^i}\right)^{1/2} - 1\right] \tag{5.33}$$

which can be solved for V_a^i. For this ideal case, it doesn't matter which pair of fibers is selected to determine V_a^i. In the more general case of non-uniform packing and arbitrarily sized fibers, solution of Eq. 5.33 for each pair of fibers in the cell would result in three different values for V_a^i. Consequently, it becomes necessary to evaluate V_a^i in a consistent manner from cell to cell.

Fortunately, the basic definition of V_a as the maximum possible fiber volume fraction for a given configuration provides the key insight. Defining configuration of the fibers in the cell by the set of values of the interior angles of the triangular cell (or, in the more

general case of arbitrarily shaped unit cell, the orientations of the vectors joining the fiber centers within the cell), V_a^i is seen to be reached once any one of the pairs of fibers within the cell touch. This definition can be written formally by noting that the length of side $L_{j,k}$ between fibers (j,k) is

$$\begin{aligned} L_{j,k} &= \tfrac{1}{2}\left(d_j + d_k\right) + \delta_o^{j,k} \\ &= \bar{d}_{j,k} + \delta_o^{j,k} \end{aligned} \tag{5.34}$$

which, upon rearranging terms, can be written as

$$\frac{L_{j,k}}{\bar{d}_{j,k}} = 1 + \frac{\delta_o^{j,k}}{\bar{d}_{j,k}} \tag{5.35}$$

Requiring that the fibers maintain their present configuration (i.e., the same positions relative to one another as measured by the interior angles of the cell or, equivalently, the orientations of the vectors joining fiber centers) enables one to write

$$\begin{aligned} \frac{V_a^i}{V_f^i} &= \min_{\text{cell } i}\left\{\frac{L_{j,k}}{\bar{d}_{j,k}}\right\} \\ &= 1 + \min_{\text{cell } i}\left\{\frac{\delta_o^{j,k}}{\bar{d}_{j,k}}\right\} \end{aligned} \tag{5.36}$$

which can be solved for V_a^i.

Volume

The volume, V_i, associated with element i is approximated by

$$V_i = A_z^i \cdot \Delta z \tag{5.37}$$

where A_z^i is the area of the element cross-section and Δz is the distance between successive layers in the z-direction (assumed constant). Note that, for the case of perfectly straight fibers, Eqn. 5.33 is exact. The relationship becomes approximate, however, when the fibers are allowed to be wavy.

Control Volume Interfaces

As a result of the discretization scheme illustrated in Figure 5.7, the model consists of two types of contol volume elements, pores and gaps. Each contol volume element will have two types of boundaries, intra-layer interfaces with neighbors of unlike type (i.e.,

pore-gap) and inter-layer interfaces with neighbors of like type (i.e., pore-pore and gap-gap). Important variables describing the interface between adjacent control volumes (i,j) are the interface area $A_{i,j}$, hydraulic radius $R_h^{i,j}$, effective free surface radius of curvature $R_c^{i,j}$, control volume centroid separation $L_{i,j}$, and conductance $h_{i,j}$. Calculation of the conductance $h_{i,j}$ will be discussed in Section 5.5. With the exception of $R_c^{i,j}$, the other interface variables are all used in the calculation of $h_{i,j}$ and, therefore, will be discussed together. Since $R_c^{i,j}$ is assumed to be related to the hydraulic radius $R_h^{i,j}$, it will also be discussed here.

Discretization of the pore space into control volumes can result in discontinuities in the values of geometric variables, such as element area and hydraulic radius, at the boundaries of adjacent elements. For axial (i.e., pore-pore, gap-gap) interfaces, this is handled by using volume-weighted averages of the variable, $x_{i,j}$, under consideration or, formally,

$$x_{i,j} = \frac{x_i V_i + x_j V_j}{V_i + V_j} \tag{5.38}$$

where x_k is the value of x based on control volume element k and V_k is the volume of element k. Consequently, $A_{i,j}$, $R_h^{i,j}$, and $R_c^{i,j}$ are calculated using Eq. 5.38 for axial interfaces once all of the in-plane control volumes have been defined.

Values of $A_{i,j}$, $R_h^{i,j}$, and $R_c^{i,j}$ for intra-layer (i.e., pore-gap) interfaces are calculated based on the values at the gap center line, where the separation of the two fibers shared by the adjacent pore and gap contol volumes is a minimum, $\delta_o^{i,j}$. Under the assumption of small $\delta_o^{i,j}$ (i.e., high fiber volume fraction), the apparent curvature across the gap is relatively small and this approximation is both reasonable and efficient. The resulting approximations are

$$A_{i,j} = \delta_o^{i,j} \Delta z \tag{5.39}$$

$$R_h^{i,j} = \frac{\delta_o^{i,j}}{2} \tag{5.40}$$

$$R_c^{i,j} = R_h^{i,j} \tag{5.41}$$

all which are the same as for two parallel plates separated by $\delta_o^{i,j}$.

5.4 Calculation of Conductance Matrix

The individual entries $h_{i,j}$ of the conductance matrix, **H**, are calculated (or read from a file) at the start of the simulation. Both the axial and transverse conductance values are based on the assumptions of low Reynolds number and steady, saturated flow between adjacent control volumes. As a simplifying assumption, the conductance values $h_{i,j}$ are treated as constants independent of control volume saturation.

5.4.1 Axial Conductance

Modeling the pore space between adjacent nodes (i,j) as a non-circular duct of hydraulic diameter, D_h, and assuming low Reynolds number and steady flow, one can write the pressure drop between nodes (i,j) as

$$\Delta p = f\frac{\rho L U^2}{2D_h} \tag{5.42}$$

where f is the Darcy friction factor and is given by [12]

$$f = \frac{c}{\mathrm{Re}_{D_h}} = \frac{c\mu}{\rho U D_h} \tag{5.43}$$

The value of c is dependent on the local geometry and $\mathrm{Re}_{D_h} = \rho U D_h/\mu$ is the Reynolds number based on the hydraulic diameter. As an approximation, c is set equal to 64 for the large interstitial "pores" and 96 for the inter-fiber "gaps". These values correspond to approximating the axial flow within the pores with flow in circular conduits and within the gaps with flow between two parallel, flat plates separated by a gap equal to the minimum separation between fibers (i,j). Substituting Eq. 5.43 into Eq. 5.42 and noting that $Q_{i,j} = A_{i,j}U_{i,j}$ yields

$$Q_{i,j} = \frac{2\left(D_h^{i,j}\right)^2 A_{i,j}}{c_{i,j}\mu L_{i,j}}\Delta p \tag{5.44}$$

Consequently, the axial conductance across the interface (i,j) is given by

$$h_{i,j} = \frac{2\left(D_h^{i,j}\right)^2 A_{i,j}}{c_{i,j}\mu L_{i,j}} = \left(\frac{32}{c_{i,j}}\right)\frac{\left(R_h^{i,j}\right)^2 A_{i,j}}{\mu L_{i,j}} \tag{5.45}$$

where

$$c_{i,j} = \begin{cases} 64 & \text{pore elements} \\ 96 & \text{gap elements} \end{cases} \tag{5.46}$$

5.4.2 Transverse Conductance

The values of the transverse conductances were calculated following the lubrication approach used in [13,14]. Their approach was generalized for this study in order to accomodate arbitrary variations in the fiber radii and, therefore, led to a relaxation of the symmetry requirements of their analyses. Figure 5.10 illustrates the geometry in the gap between fibers *i* and *j*.

A one-dimensional x-momentum balance across the gap assuming a quasi-steady, inertia-free (i.e., "lubrication") flow yields

$$0 = -\frac{dp}{dx} + \mu \frac{d^2u}{dx^2} \tag{5.47a}$$

where

$$u(\delta_i(x)) = u(\delta_j(x)) = 0 \tag{5.47b}$$

Separating variables, integrating Eq. 5.47(a) and applying the boundary conditions Eq. 5.47(b) results in the velocity profile, $u(y)$, at any x in the gap,

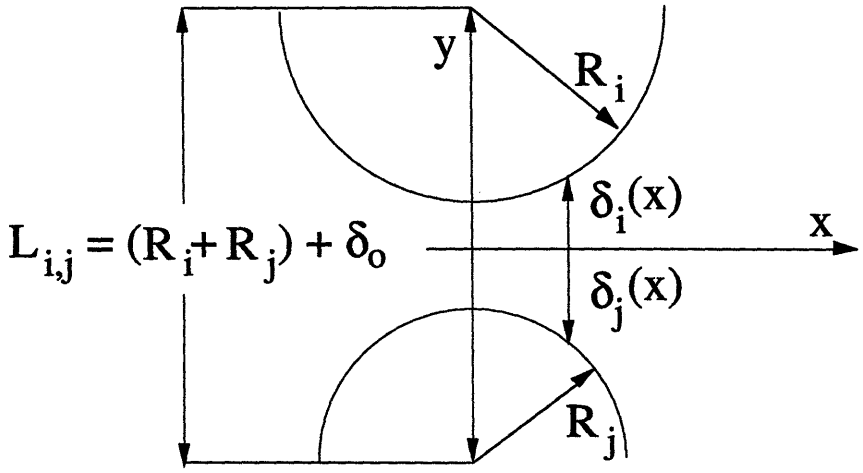


Figure 5.10 Inter-fiber gap geometry.

$$u(y) = \frac{1}{2\mu}\left(\frac{dp}{dx}\right)\left[y^2 - (\delta_i - \delta_j)y - \delta_i\delta_j\right] \tag{5.48}$$

where

$$\delta_i = \delta_i(x)$$
$$\delta_j = \delta_j(x)$$

Integrating Eq. 5.48 over the cross-sectional area of the gap (assuming unit depth in the z-direction) gives the local, volumetric flow rate at x

$$Q = Q(x) = -\frac{1}{12\mu}\left(\frac{dp}{dx}\right)\delta^3 \tag{5.49}$$

where

$$\delta = \delta(x) = \delta_i(x) + \delta_j(x).$$

Note that, by continuity, Q is constant across the gap. Separating variables and integrating from the center of the gap at $x_1 = 0$ to x_2 yields the pressure drop across the gap

$$\Delta p = -12\mu Q \int_{x_1=0}^{x_2} \delta^{-3} dx = \underbrace{-12\mu I(0,x_2)}_{1/h_{gap}} Q \tag{5.50}$$

and, as indicated in Eq. 5.50, the inverse of the conductance, h_{gap}, across the interface between the gap control volume centered at $x = 0$ and the adjacent pore control volume.

Evaluation of the integral, $I(0,x_2)$, in Eq. 5.50 requires knowledge of the functional form of $\delta(x)$. As shown in Figure 5.10, one can write

$$\delta(x) = \delta_i(x) + \delta_j(x) = L_{i,j} - \left[\left(R_i^2 - x^2\right)^{1/2} + \left(R_j^2 - x^2\right)^{1/2}\right] \tag{5.51}$$

for the gap height as a function of x. For arbitrary R_i and R_j evaluation of $I(0,x_2)$ must be approximate. Here this was done using Simpson's rule, although any of a number of approximate integration techniques could be used. This integration is performed only once in the simulation since the values of the conductance matrix are assumed constant.

5.5 Scaling of Equations

During solution of the system of finite difference equations, Eq. 5.4, the governing equation and related boundary conditions are non-dimensionalized. The scaling factors

chosen are based on the bundle-scale values of the variables in Eq. 5.4, which is rewritten below as

$$\left(\langle \mathbf{H} \rangle_{L_z} \mathbf{h}\right) P_{L_z} p = Q_{in} q \tag{5.52}$$

where

$$\langle \mathbf{H} \rangle_{L_z} = \frac{\pi \left(\langle R_h \rangle_{L_z} \right)^4}{8 \mu L_z} \tag{5.53}$$

is an average axial conductance based on the volume-weighted, tow average, axial hydraulic radius. Defining the tow length pressure scale as

$$P_{L_z} = \frac{Q_{in}}{\langle \mathbf{H} \rangle_{L_z}} \tag{5.54}$$

one can then write

$$\left(\frac{\langle \mathbf{H} \rangle_{L_z} \cdot P_{L_z}}{Q_{in}} \right) \mathbf{h} p = q \tag{5.55}$$

but, since

$$\left(\frac{\langle \mathbf{H} \rangle_{L_z} \cdot P_{L_z}}{Q_{in}} \right) = 1 \tag{5.56}$$

one can rewrite Eq. 5.55 as

$$\mathbf{h} p = q \tag{5.57}$$

Having defined the pressure scale P_{L_z}, Eq. 5.5 becomes

$$p_c^{i,j} = \frac{P_c^{i,j}}{P_{L_z}} = \frac{2 \gamma \cos \theta_i}{R_c^{i,j} \cdot P_{L_z}} \tag{5.58}$$

5.6 Definition of Solution Domain for Current Time Step

Since the advancing flow front continuously incorporates new control volumes into the set of saturated control volumes, it is necessary to redefine the domain for analysis at the start of each time step. As illustrated schematically in Figure 5.11 this involves determining whether each node and its associated control volume in the model is behind, on, or beyond the flow front and whether empty and partially filled control volumes are in front of the global flow front or within entrapped voids.

5.6.1 Determination of Node Status

The status of every node is determined at the beginning of each time step. This is done by scanning through the global list of nodes in the model and noting that the filling status of node i implies certain necessary conditions on the saturations of its neighbors j. These conditions are summarized in Table 5.1 and also serve as definitions for the various types of nodes in the calculation. Note that the node classifications summarized in Table 5.1 do not explicitly include nodes within entrapped voids. Identification and incorporation of void nodes are considered in detail in Section 5.6.2.

Table 5.1 Node Status Criteria

Status	Criteria
Not in solution domain	$sat_j = 0$ for all j and i is not a source node
At front	$0 < sat_i < 1$ or $sat_i = 0$ and {at least one $sat_j = 1$ or i is a source node}
Next to front	$sat_i = 1$ and at least one $sat_j = 1$
Behind front (full)	$sat_i = 1$

5.6.2 Void Identification

Voids are identified using **B**readth **F**irst **S**earch and **T**raversal (BFST) of the undirected graph (i.e., list) formed by the nodes behind the first fully dry plane (i.e., the vent plane defined in Figure 5.11). BFST is a basic search and sorting method and is described in [15]. The magnitude of the advantage gained by definition of the vent plane is dependent on the problem geometry and the time (i.e., fraction of all possible nodes filled) into the simulation. In the case of axial flow considered here, the increase in search due to judicious definition of the vent plane efficiency is greatest early in the simulation.

Using BFST, the nodes in the original graph are grouped into user-defined sub-groups. In this application, the nodes behind the vent plane are sorted in two steps. In the first step, the nodes are divided into two categories of linked (i.e., neighboring) groups of nodes: fully saturated nodes and dry nodes behind the vent plane. In the second step, the groups of dry nodes are then searched to identify and eliminate those groups that can be connected to nodes in the vent plane. The remaining groups of dry nodes are those that are behind the global flow front and are not connected to the vent plane. These are the nodes that make up the entrapped voids.

Currently, once the simulation identifies nodes and their corresponding control volumes as belonging to a void, all of the interface conductances to neighboring nodes are set equal to zero. As a result, the saturations of the control volumes contained within the void regions are held constant throughout the remainder of the simulaiion. This means that the void volumes currently calculated are upper bounds since the voids are not allowed to shrink due to increases in the surrounding fluid pressure. In addition, the voids are stationary. This is not a serious problem if, as discussed in Section 4.2, the local balance of viscous to capillary forces (i.e., the capillary number $Ca = \mu U/\gamma$) is below the threshold required for mobilization of the majority of voids.

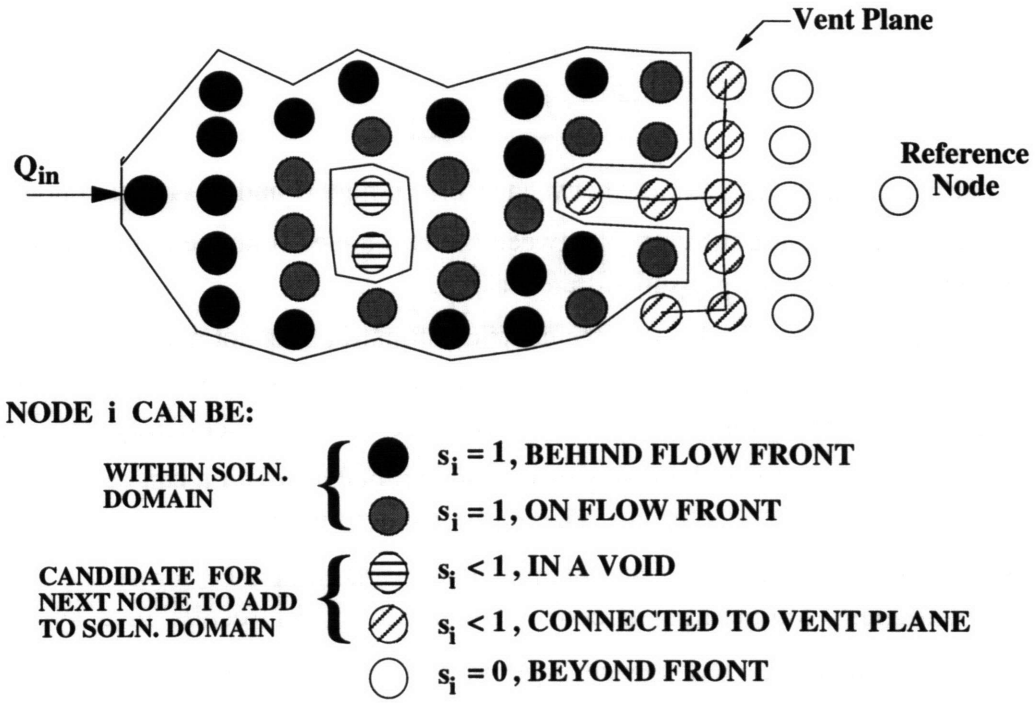


Figure 5.11 Schematic of node classification.

5.7 Solution of Governing Equation

Solution of the governing system of equations, Eq. 5.57, is performed iteratively using relaxation [16,17]. The resulting formula for the kth iteration for the pressure at node i is

$$p_i^{(k)} = (1-\omega)p_i^{(k-1)} + \omega\left[\frac{q_{ext}^{i\leftarrow} + \sum_j a_{i,j}h_{i,j}\left(p_j^{(m)} - p_c^{i,j}\right)}{\sum_j a_{i,j}h_{i,j}}\right] \tag{5.59}$$

where the initial trial solution, $p_i^{(0)}$, corresponds to the pressure obtained in the previous time step, $q_{ext}^{i\leftarrow}$ is the net external volumetric flow into node i, ω is the overrelaxtion parameter ($\omega = 1$ results in Gauss-Seidel iteration, $1.2 < \omega < 1.6$ is typical in successive overrelaxation, SOR), and

$$a_{i,j} = \begin{cases} 0 & sat_j < 1 \text{ and } p_i^{(k-1)} - \left(p_j^{(m)} - p_c^{i,j}\right) < 0 \\ 1 & \text{all other } (i,j) \end{cases} \tag{5.60}$$

Following conventional notation, the superscript (m) in Eq. 5.60 refers to the value of p_j calculated in the mth iteration of Eq. 5.59. For both the Gauss-Seidel and SOR methods, $p_j^{(m)}$ should be the most recently calculated value of p_j. Since Eq. 5.59 is evaluated for each node i in ascending numerical order, the rule for $p_j^{(m)}$ is

$$p_j^{(m)} = \begin{cases} p_j^{(k)} & j < \mathrm{i} \\ p_j^{(k-1)} & j > \mathrm{i} \end{cases} \tag{5.61}$$

Equation 5.60 is a statement that flow between saturated node i and unsaturated node j is allowed only if the presure difference between the two nodes is above a minimum value determined by the interfacial pressure difference $p_c^{i,j}$.

A consequence of Eq. 5.60 is that the coefficient matrix $\mathbf{h}$ is not necessarily constant throughout the entire solution process since its entries are dependent on the pressures at nodes i and j and the interfacial pressure difference $p_c^{i,j}$. Consequently, the system $\mathbf{h}p = q$ becomes non-linear and convergence to an acceptable solution can be slower than for the linear system with $\mathbf{h}$ constant. The reason for defining the coefficients $a_{i,j}$ using Eq. 5.60 in the first place is that one obtains a strict mass balance across the solution domain boundaries. Maintaining $\mathbf{h}$ constant with all $a_{i,j} = 1$ results in a system of equations that satisfy a net mass balance at the price of introducing artificial influxes at the advancing flow front for nodes i and j where $p_i - \left(p_j - p_c^{i,j}\right) < 0$.

At the end of each iteration, k, the maximum solution error, $e_{\max}^{(k)}$, is defined as

$$e_{\max}^{(k)} = \max_i \left\{ \left| p_i^{(k)} - p_i^{(k-1)} \right| \right\} \tag{5.62}$$

The equation solver is stopped if the maximum error is less than a user-defined tolerance, $e_{\max}^{(k)} \le Tol$, or the number of iterations exceeds a maximum limit, $k > k_{\max}$. The results presented in this thesis were obtained with $Tol = 10^{-10}$ and $k_{\max} = 5 \times 10^4$.

In order to accomodate the possibility that unsaturated node j is part of a void with a pressure obeying the ideal gas law, all nodal pressures in the equation solver routine are absolute. Prior to returning to the calling routine, the pressures are returned to values relative to a user-defined reference point.

5.8 Selection of time step and advancement of flow front

Selection of the time increment, Δt, by which to advance the simulation to the next time step is based on the current pressure distribution. As in [1], the maximum allowable time increment is set to the time required to add (i.e., fill) one additional node to the analysis domain, or

$$\Delta t = \min_{i \in \{\text{at front}\}} \left\{ t_{fill}^i = \frac{(1 - sat_i) V_i}{q_i^{\leftarrow}} \right\} \tag{5.63}$$

where t_{fill}^i, sat_i, V_i, and $q_i^{\leftarrow}$ are the filling time, current saturation, volume and net flow into control volume i. In order to ensure that only one node was added to the solution domain when analyzing nominally ideal fiber arrays, a small amount of fiber waviness $\left(\varepsilon = 10^{-2}\right)$ was introduced into the model.

Once the new time increment has been calculated, the flow front is advanced by updating the saturations of all the control volumes i where $i \in \{\text{at front}\}$. Formally, this yields

$$sat_i^{t+\Delta t} = sat_i^t + \frac{\Delta t q_i^{\leftarrow}}{V_i} \tag{5.64}$$

where Δt is given by Eq. 5.63. Note that, for a given pressure distribution, some nodes can have a net flow out of the control volume (i.e., $q_i^{\leftarrow} < 0$). If $\Delta t > t_{drain}^i$, where t_{drain}^i is the time required to drain contol volume i, then $sat_i^{t+\Delta t}$ calculated using Eq. 5.64 can become negative. Consequently, the program adds the following condition to Eq. 5.64

$$sat_i^{t+\Delta t} = 0 \qquad \text{for } q_i^{\leftarrow} < 0 \text{ and } \Delta t > t_{drain}^i \tag{5.65}$$

where

$$t_{drain}^i = \frac{sat_i V_i}{|q_i^{\leftarrow}|} \tag{5.66}$$

With the calculation of the new control volume saturations completed, the current values of the pressure and saturation variables are written into the appropriate variables for the previous time step,

$$\begin{aligned} p_i^t &= p_i^{t+\Delta t} \\ sat_i^t &= sat_i^{t+\Delta t} \end{aligned} \tag{5.65}$$

and the simulation is ready to repeat the process for the next time step.

5.9 References

1. M. Blunt,, M., and King, P., 1991, "Relative Permeabilites from Two- and Three-Dimensional Pore-Scale Network Modelling", *Transport in Porous Media*, **6**:407-433.

2. Levine, S., Reed, P., and Shutts, G., 1977, "Some Aspects of Wetting/Dewetting of a Porous Medium", *Powder Technology*, **17**:163-181.

3. Larson, R., Scriven, L., and Davis, H., 1977, "Percolation Theory of Residual Phases on Porous Media", *Nature*, **268**(4):409-413.

4. Larson, R., Davis, H., and Scriven, L., 1981, "Displacement of Residual Nonwetting Fluid From Porous Media", *Chemical Engineering Science*, **36**:75-85.

5. Sheidegger, A., 1974, *The Physics of Flow Through Porous Media*, 3rd ed., University of Toronto Press, Toronto.

6. Cai, Z., and Gutowski, T., 1992, "The 3-D Deformation Behavior of a Lubricated Fiber Bundle," *Journal of Composite Materials*, **26**(8):1207-1237.

7. Gutowski, T., and Dillon, G., 1992, "The Elastic Deformation of Lubricated Carbon Fiber Bundles: Comparison of Theory and Experiments," *Journal of Composite Materials*, **26**(16):2330-2347.

8. Kim, Y., McCarthy, S., and Fanucci, J., 1990, "Compressibility and Relaxation of Fiber Reinforcements During Composite Processing", in *Proceedings of the SPE 48th Annual Technical Conference ANTEC '90*, pp. 1252-1256.

9. Dillon, G., private communication.

10. Beer, F., and Johnston, E., 1981, *Mechanics of Materials*, McGraw-Hill Book Co., New York, pp. 574-583.

11. Tallarida, R., 1992, *Pocket Book of Integrals and Mathematical Formulas*, 2nd ed., CRC Press, Boca Raton, p. 25.

12. White, F., 1986, *Fluid Mechanics*, 2nd ed., McGraw-Hill Book Co., New York, p. 322.

13. Lundström, T., 1991, *SICOMP Technical Report 91-001: Micromechanical Analysis of the Permeability of a Fibrous Preform.*

14. Gebart, B., 1992, "Permeability of Unidirectional Reinforcements for RTM", *Journal of Composite Materials*, 26(8):1100-1133.

15. Horowitz, E., and Sahni, S., 1978, *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, MD, 263-269.

16. Conte, S., and de Boor, C., 1980, *Elementary Numerical Analysis: An Algorithmic Approach*, 3rd ed., McGraw-Hill, New York.

17. Hildebrand, F., 1956, *Introduction to Numerical Analysis*, 2nd ed., Ch 10, Dover, New York.

6 EXPERIMENTAL OBSERVATIONS

The results of flow visualization experiments performed in this thesis are presented and discussed. These results can be divided into two main groups. The first group of results documents observations of flow in commercial fabrics and illustrate characteristics of flow at the various length scales present within typical fiber preforms. The second set of results are concerned with a more detailed examination of the flow and void formation at the fiber scale. These results include observations of flow in model composites described in Section 3.4.

6.1 Observations of Flow In Commercial Fabrics

Observation of flow in the commercial unidirectional fabrics (ANCAREF and VECTORPLY samples, see Appendix A) clearly showed the existence of two distinct regions of flow within the mold, the gap (or interface) between the preform and the tool and the interior of the preform fabric. Figures 6.1 and 6.2 indicate the advancing flow fronts in both regions. Note that the definition of the two fronts is a direct function of the amount of dry scattering fibers between the front and the CCD camera and the depth of focus of the imaging optics. Unless otherwise noted, the optical system was always focused on the top surface of the preform as viewed through the observation window. Consequently, the interface front is much more sharply defined than the interior front.

Consider first the ANCAREF sample, shown in Figure 6.1, which is composed of loose bundles of aligned fibers sandwiched between two thin "veil" layers. The interface front is characterized by the irregular surface formed by the tool (in this case the glass observation window) and the top surface of the fabric (which is not the top layer of fibers but, rather, the veil layer holding the loose bundles of fibers in place). The interface flow fronts observed are typical of the observations of the "macro voids" reported in [1,2] in random fabrics. This is probably because the veil material for the ANCAREF sample is very much like a thin layer of random mat placed on top of the aligned fibers. Some of these random fibers can be seen in Figure 6.1. Also shown clearly in Figure 6.1 are the interior voids which are not analyzed in [1,2].

As shown in the sequence, the voids formed in the interface are fairly large compared to those within the interior of the fabric. Consequently, using the arguments behind the development of Eq. 4.30, the interface voids should be easier to flush out (i.e., via "bleeding") than the interior voids. The voids that occur in the interface region are the ones that are responsible for the high amount of secondary processing required for parts with

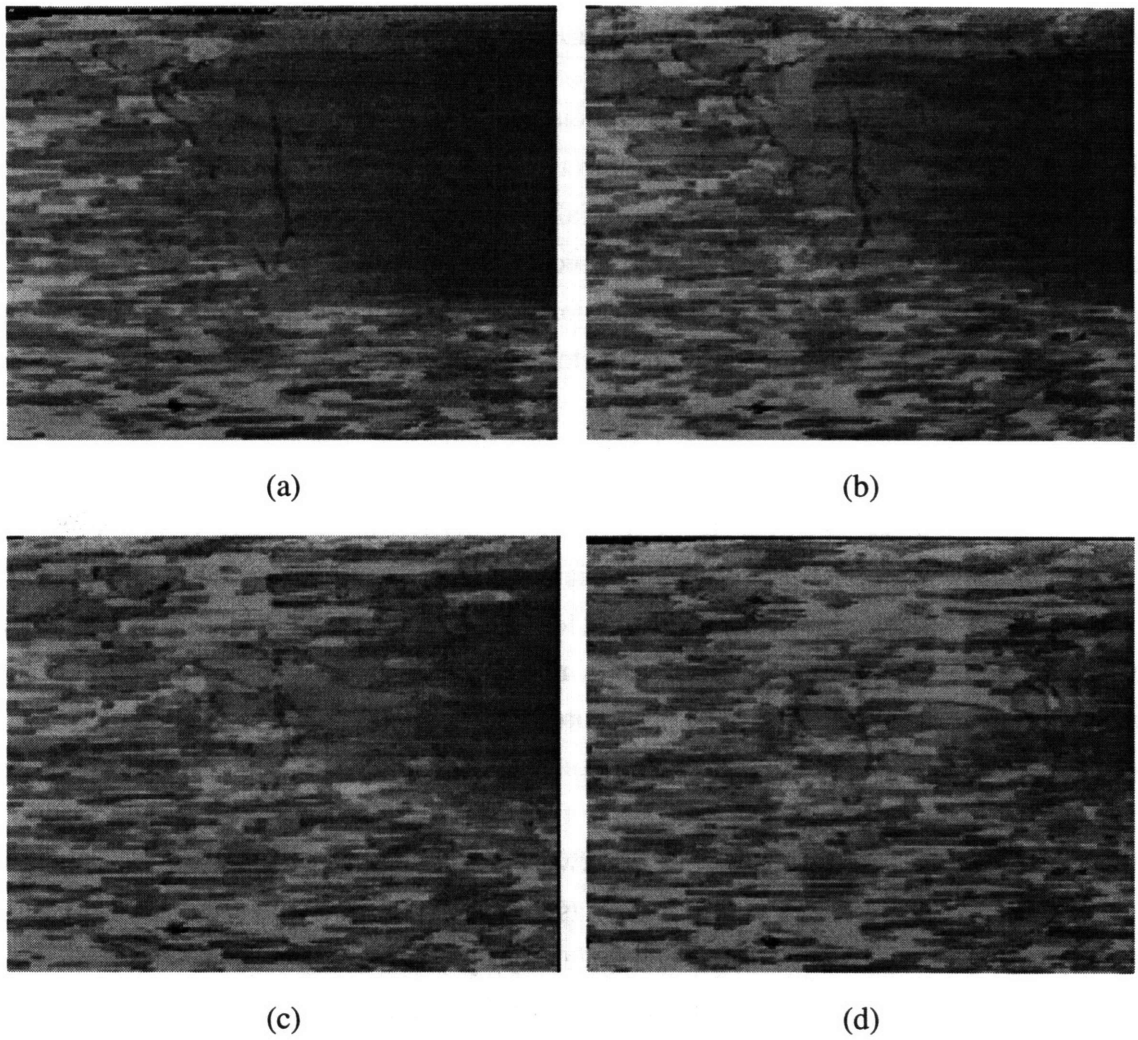


Figure 6.1 Fiber-scale flow within a unidirectional fabric: (a) $t = 0$, (b) $t = 0.3s$, (c) $t = 0.6s$, (d) $t = 0.9s$.

customer-visible surfaces. Elimination of this class of voids can, consequently, produce significant labor savings for such parts.

As in the ANCAREF sample, the interface and interior flow fronts are clearly visible in the VECTORPLY sample, shown in Figure 6.2, which is composed of well-defined tows held in place by cross-stitching. Unlike the ANCAREF sample, however, the interface region created between the tool and the VECTORPLY fabric has relatively large variations in the transverse (i.e., thickness) dimension because of the typical tow shape in this fabric. With the fabric sample compressed, as in the experiment shown in Figure 6.2, the interface region corresponding to the center of a tow has a thickness on the order of the average interior fiber separation. In the region corresponding to the center of the inter-tow channel, the interface thickness is greatest.

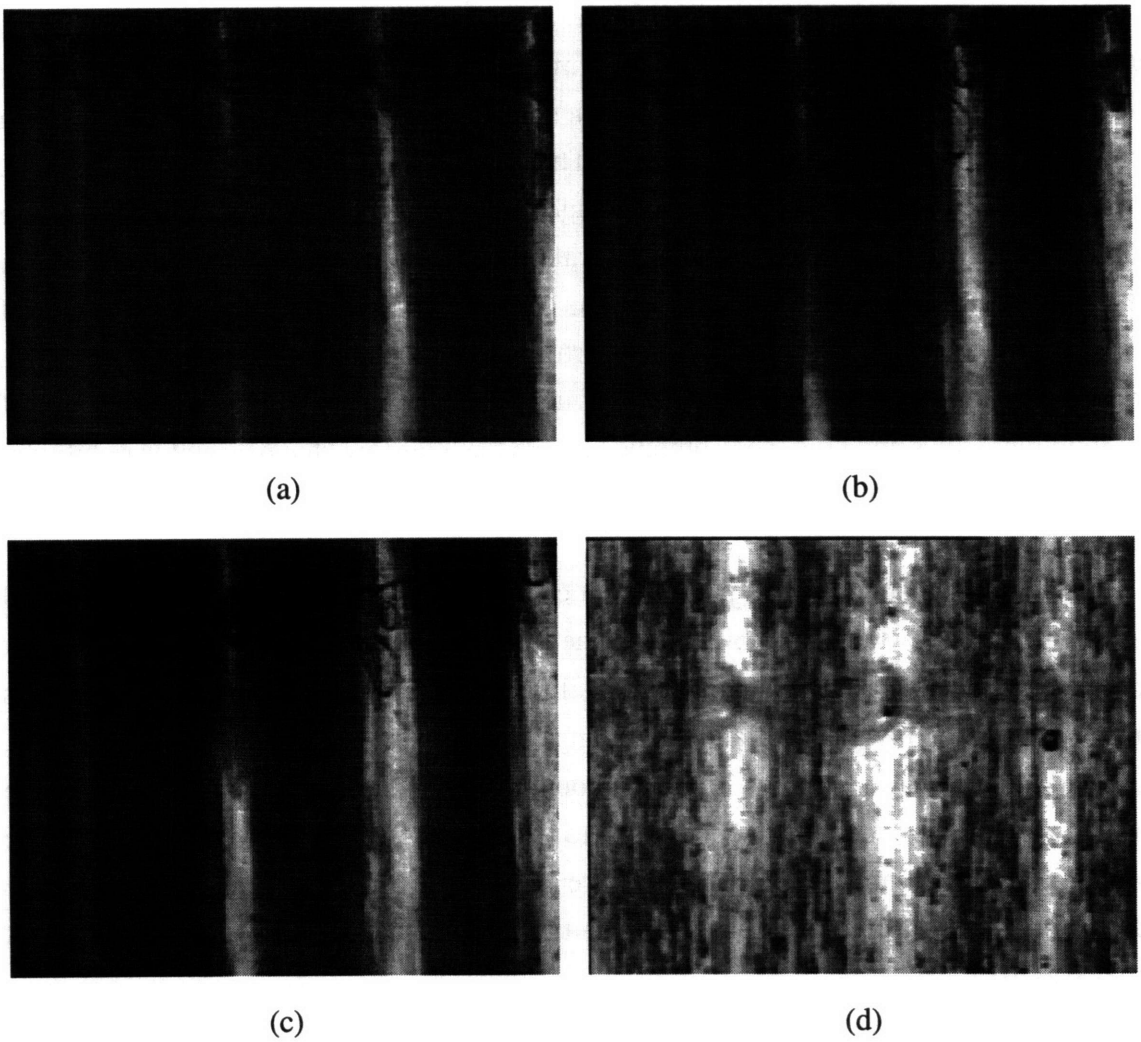


Figure 6.2 Fiber-scale flow within a stitched, unidirectional fabric: (a) $t = 0$, (b) $t = 0.2s$, (c) $t = 0.4s$, (d) image taken at a slightly different position (downstream) after impregnation.

As shown in the sequence in Figure 6.2, the interface voids are typically formed when the flow front within the inter-tow channel runs into the cross-stitching and turns back upon itself, entrapping relatively large air bubbles. If the channel flow is fast enough (i.e., the effective channel capillary number is greater than the minimum for void mobilization through the cross-stitching) the bubble can be forced through the cross-stitching. Depending on the speed of the fluid and the channel/cross-stitching geometry, the bubble will pass through whole or will break up into smaller bubbles before continuing downstream to the next constriction. There the process begins again. Voids formed within the tows look like the interior voids within the ANCAREF sample shown in Figure 6.1.

Comparing the flow in the various fabrics, it is clear that the unidirectional case is an archetype for flow in the more complicated materials. Another way of saying this is that there is a certain amount of separation of phenomena at the various length scales present within typical reinforcement materials. This separation, however, is not complete. Consider, for example, the sequence of images in Figure 6.2 showing infiltration of the VECTORPLY material. Two main length scales can be seen immediately upon inspection of the material, the fiber and tow scales. As in the previous discussions, the sources of variability at the fiber scale come from perturbations in fiber diameter, surface finish, waviness, and packing. At the tow scale, variables include the resulting shape and size of the individual tows and how they combine to form the inter-tow channels and variability in the joining cross-stitching. It is the effects of these variables that are most plainly seen in Figure 6.2.

As the impregnation process progresses it can be seen that the resulting axial and radial flows within the tow are a combination of the flow in the channel extending into the tow and the original axial flow within the tow. Failure to incorporate the significant axial flows present within the tow, as is often the case in so-called "simplified" analyses such as [3], can lead to large errors in predicting the amount of time necessary for full wet-out of the tow. Applying the model developed in [3] to the case shown in Figure 6.2 produced the typical flow front profile shown in Figure 6.3. Note the discrepancy between the calculated and measured profiles. The resulting ratio of measured to predicted wet-out times was 10:1.

The effects of fiber scale variability on the tow scale flow can be seen in the resulting tow scale flow front profile. As local (i.e., fiber scale) variability can lead to finger formation and subsequent instabilities leading to air entrapment, the tow scale front is forced to flow around the entrapped voids, which act as local obstacles to flow if the local capillary number Ca is not high enough to flush the voids out of the tow.

Comparing the observed Ca with the critical values estimated using Eqs 4.8 and 4.10 and the mobilization criteria Ca_m, Eq. 4.30 with Eqs 4.33 and 4.34, shows that surface tension forces are significant and that the local viscous forces are not sufficient to flush out all of the voids generated.

Examining the flow in fabrics with even greater structural complexity (Figure 6.3, Sea-Glass and Figure 6.4, BGF) illustrates the basic nature of flow in the unidirectional fabric and the effective downward separation of scales (i.e., in general, flow at the larger length scales is affected by flow at the smaller length scales and not vice versa). As demonstrated in Figure 6.5, changing the tow-scale structure of the fabric by deforming it in the trellising rack described in Chapter 3, did very little to change to void entrapment within the tows.

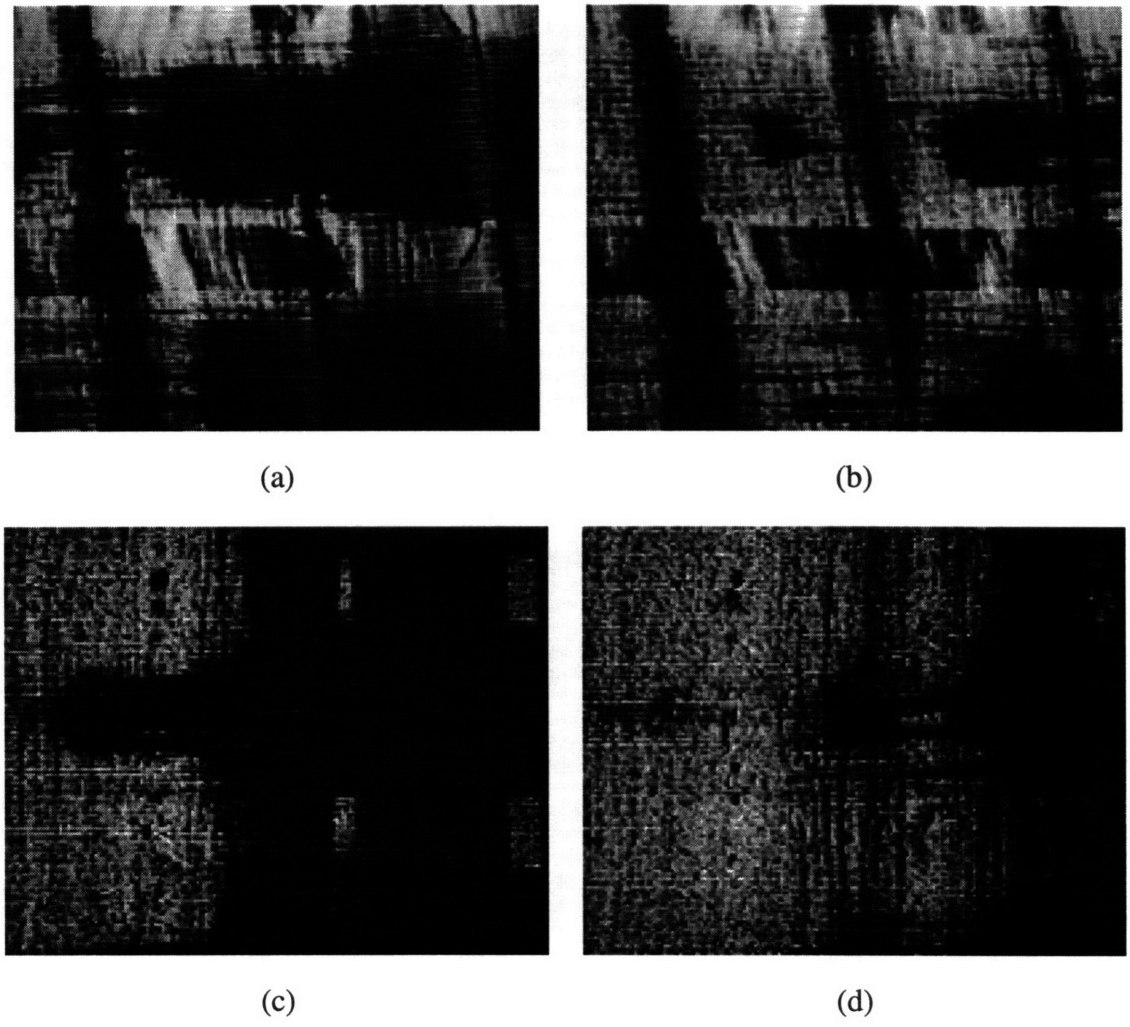


Figure 6.3 Fiber- and tow-scale flow within a 0°-90° weave fabric. (a) and (b) polyester resin with $\Delta t = 10s$, (c) and (d) C_6H_5Br with $\Delta t = 2/30s$.

Once the tow was surrounded by fluid, the air trapped within the tow was compressed and partitioned by impregnating fluid until the air and fluid pressures balanced everywhere within the tow. At the tow scale, however, the effect on void entrapment and mobilization was quite dramatic. Note that the observed tow scale void formation occurred in the inter-tow channels and was minimized in the fabric deformation modes that minimized the anisotropy of the tow-scale microstructure [4].

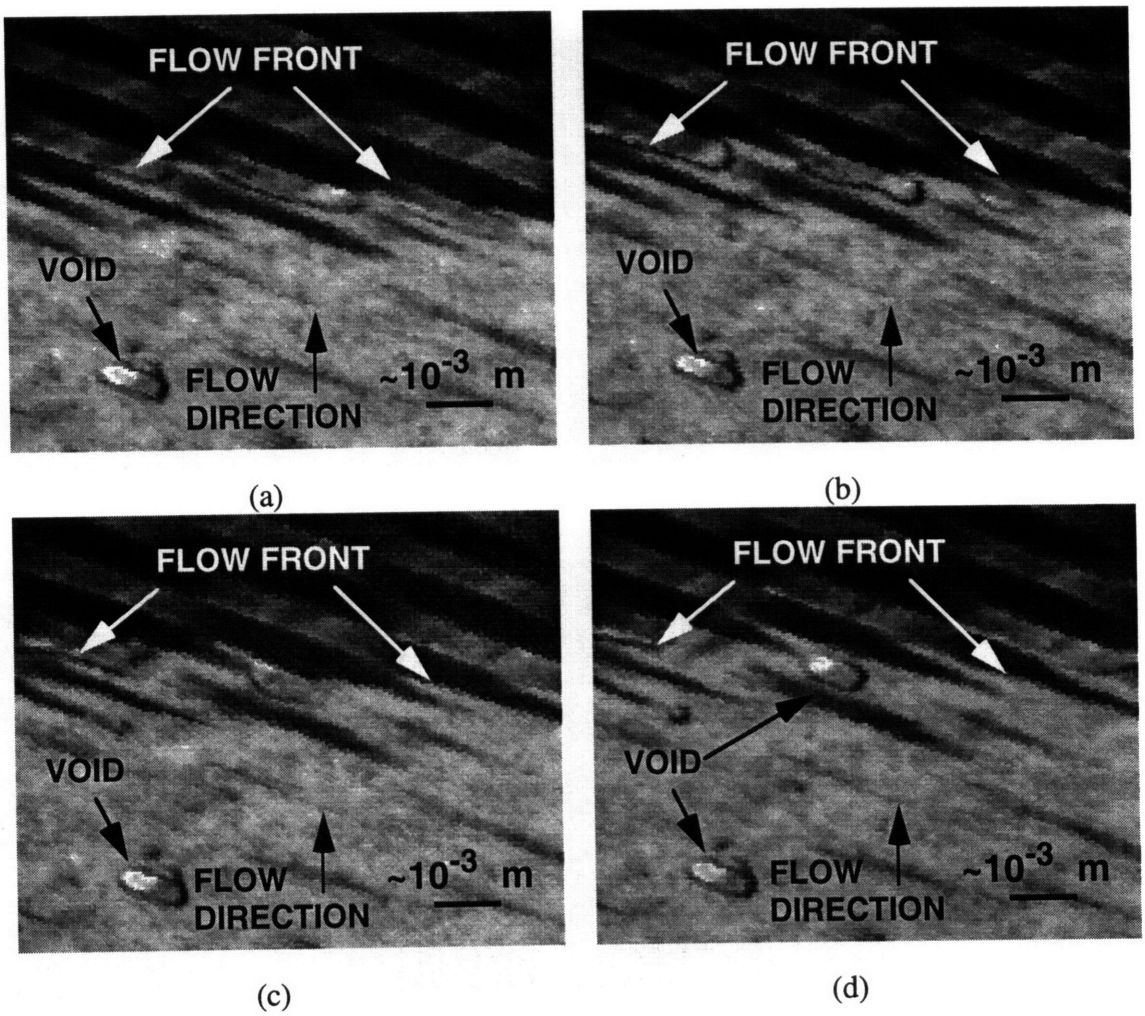


Figure 6.4 Fiber-and tow-scale flow within eight-harness satin weave fabric: (a) $t = 0$, (b) $t = 1/30\,s$, (c) $t = 2/30\,s$, (d) $t = 3/30\,s$.

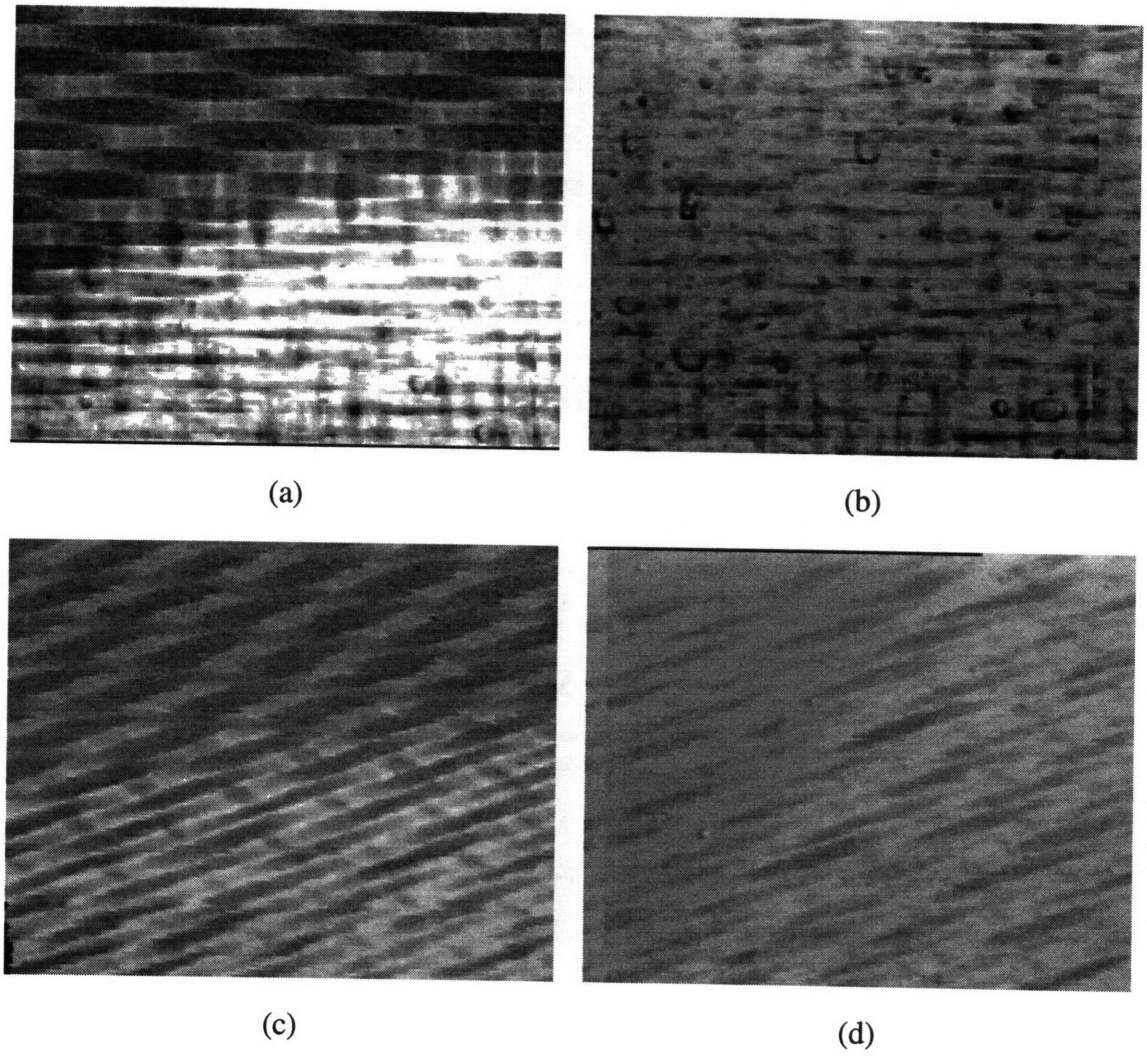


Figure 6.5 Effect of changing tow-scale structure on flow within eight-harness satin weave fabric: (a) and (b) undeformed fabric, $\theta_{warp} = 90°$, (c) and (d) $\theta_{warp} = 40°$.

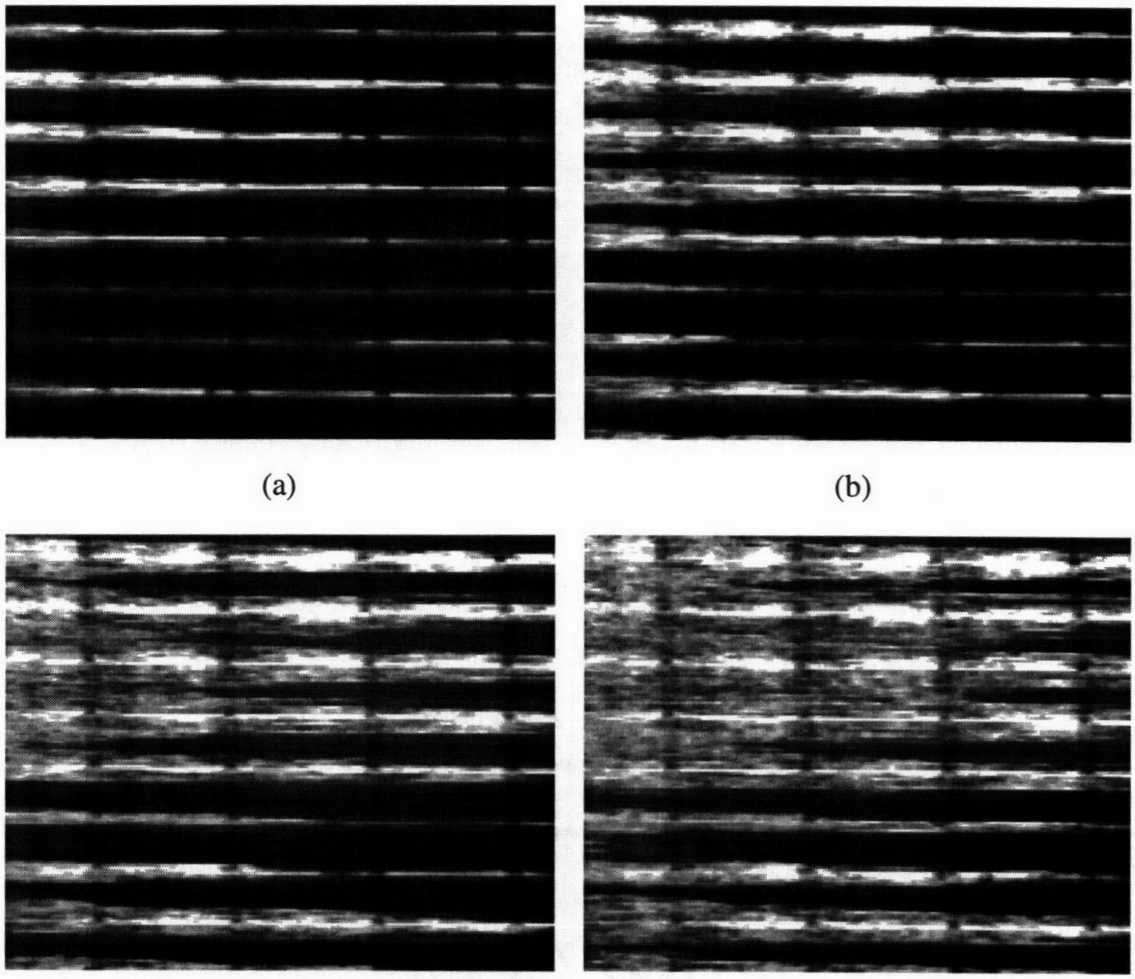


Figure 6.6 Effect of variation in tow separation on part-scale flow within a stitched, unidirectional fabric: (a) $t = 10s$, (b) $t = 20s$, (c) $t = 30s$, (d) $t = 40s$.

As another illustration of the dependence of flow at the larger scales to variability at the smaller scales, consider Figure 6.6 which shows an image taken during impregnation of the VECTORPLY fabric. Figure 6.6 clearly shows the effect of inter-tow channel (tow scale) variability on the part scale flow front profile. These experimental observations lead to the natural conclusion that a fuller understanding of the flow within the preform requires a systematic approach from the smallest to the largest length scales. Note that the traditional continuum approach begins analysis over some averaging volume typically larger than the fiber scale. Although this approach has been successful in applications such as selection of machine capacity and estimation of part-scale flows, the current need to increase the predictive capabilities of process models requires more detailed information for model refinement and validation.

Another characteristic observed in the commercial fabrics is the presence of fingering at the advancing flow front. At the fiber scale, this is similar to the "capillary fringe" analyzed in unsaturated soil flows [5]. Although preform variability was seen to cause local fluctuations in finger length (i.e., speed relative to bulk front speed), the observed fingers were typically stable in the sense of Eq. 4.18.

6.2 Fiber-Scale Void Entrapment

The observed interior voids are a problem in the development of structural components since they result in degraded mechanical properties and resistance to environmental elements, such as moisture. Ideally, these interior voids would be eliminated or, at least, minimized. Using an analysis similar to that used to develop Eq. 4.30, one can show that, theoretically, it should be possible to flush out voids if their formation cannot be prevented. Indeed, this is the result that is reported in [6,7] for random fabrics and in [8,9] for soils. These results raise the question of whether this is possible (that is to say practical) in more structured reinforcements that would be of interest in advanced composites manufacturing. Observations of voids formed in the preform-tool interface suggest this to be possible under certain circumstances [2]. In order to address the issue of the internal voids, it was decided to study the formation of these voids in greater detail by examining the fiber-scale flow within typical preforms in depth.

Using the setup described in Section 3.4, observations were made of globally axial flow at the fiber scale. These experiments were all performed using the same fluid (Cargille #5095). Initial experiments were performed with a constant pressure drop across the sample length (i.e., constant inlet pressure). This resulted in a continuously decreasing front speed, U, since

$$U = K\frac{|\Delta p|}{L} \tag{6.1}$$

where K, $|\Delta p|$, and L are the axial permeability, applied pressure drop, and wet sample length, respectively. Consequently, the capillary number, $Ca = \mu U/\gamma$, generated during each experiment varied along the length of the specimen. In all of these experiments, it was noted that void entrapment was minimal near the inlet and became significant as the flow front advanced (and slowed down).

Before attempting to relate void entrapment to flow conditions, it is necessary to understand the relationship of void entrapment to the amount of the fingering present at the flow front and the local speed of wetting (expressed in terms of some form of

dimensionless capillary number. An argument for this grouping can be seen in the development of Eqs 4.8 and 4.10). As shown in Figure 6.7, the formation of voids at the flow front occurred near the base of an advancing finger. A relationship between the local capillary number on the amount of fingering was proposed in Eq. 4.18, the condition for the existence of a stable finger of length z_f/δ.

The sequence shown in Figure 6.7 is typical. Fiber-scale void entrapment was seen to begin with the growth of fingers beyond the bulk front. These fingers would form the advancing meniscus into a complex surface which, depending on the local packing structure, would be affected by the fingers formed between the nearest two or three fibers to any given pair of adjacent fibers (i.e., in hexagonal and square packing the fibers would naturally group into unit cells as in Figure 5.7 of three and four fibers, respectively). For the range of capillary numbers in which fingering and void entrapment was observed $\left(Ca < 10^{-3}\right)$ the fingers would be most likely located within the inter-fiber gaps with the finger bases located in the adjacent pores (as in the case of capillary wetting in vertical [10,11] and horizontal [12] fiber bundles).

As the flow front advanced through the specimen, the length of the fingers would, on average, increase. Since the front speed was also seen to be decreasing at the same time, this suggested that the finger length might be a function of the bulk wetting speed as hypothesized in Eq. 4.18. As the fingers would progress through the sample, local variations in pore space geometry would cause individual fingers to deviate from the average, bulk wetting speed.

In the case of constant inlet pressure flow, this meant that the growth rate of individual fingers would be at times faster or slower than the average. In industrial practice, another source of local variability might be caused by local variations in fiber surface chemistry caused by dissolving sizing [13] or small dirt particles. This potential source of variability was effectively eliminated in these experiments because the large fibers had no sizing and, as described in Section 3.4, multiple observations were made on the same sample after initial wetting and drainage of the sample which decreased the probability of significant variation in fiber surface chemistry.

With each finger advancing through the sample, its growth rate oscillating rapidly about some average value due to the variability of the inter-fiber pore space, the complex surface of the advancing meniscus formed within each local unit cell would, in turn, try to maintain a low-energy (i.e., stable) configuration. This trend from unstable and metastable to stable states would cause the meniscus to jump occasionally from one configuration to another, entrapping small pockets of air in the process. These types of meniscus jumps are often referred to as "Haines jumps" in the literature [9,14-16].

Consider the sequence in Figure 6.7. As stated before, the sequence is typical. Note the progression of the transverse disturbance in the meniscus indicated in the figure. The time interval between images is 1/30s, the sampling rate of the video camera. Note that the initial perturbation is located on the surface of only one finger. As the original disturbance grows, another disturbance on the adjacent finger also begins to form. This adjacent finger is connected to the first finger by the surface formed by the advancing meniscus. By the next frame, the two disturbances have advanced to the center of the adjacent pore and have joined up, entrapping a small pocket of air.

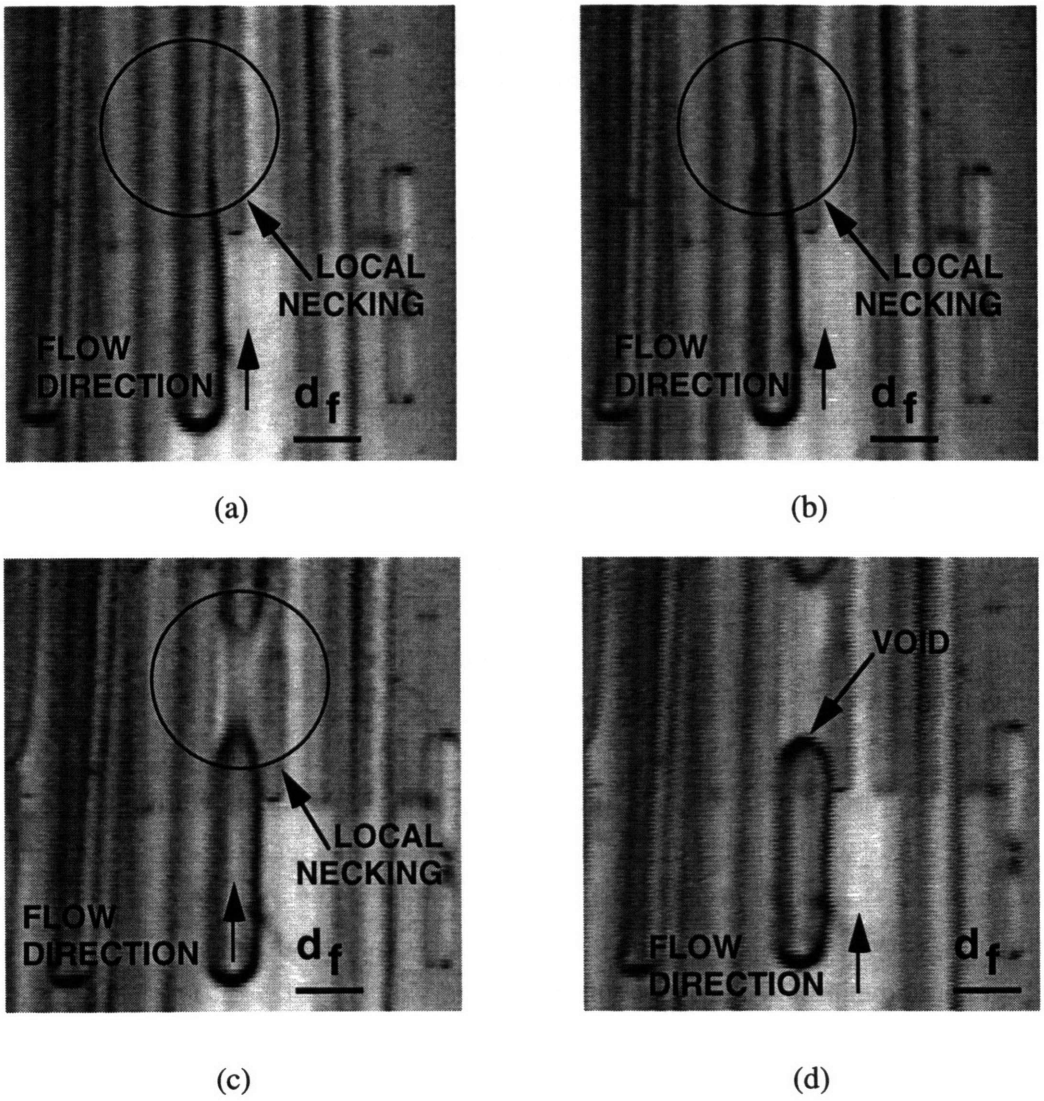
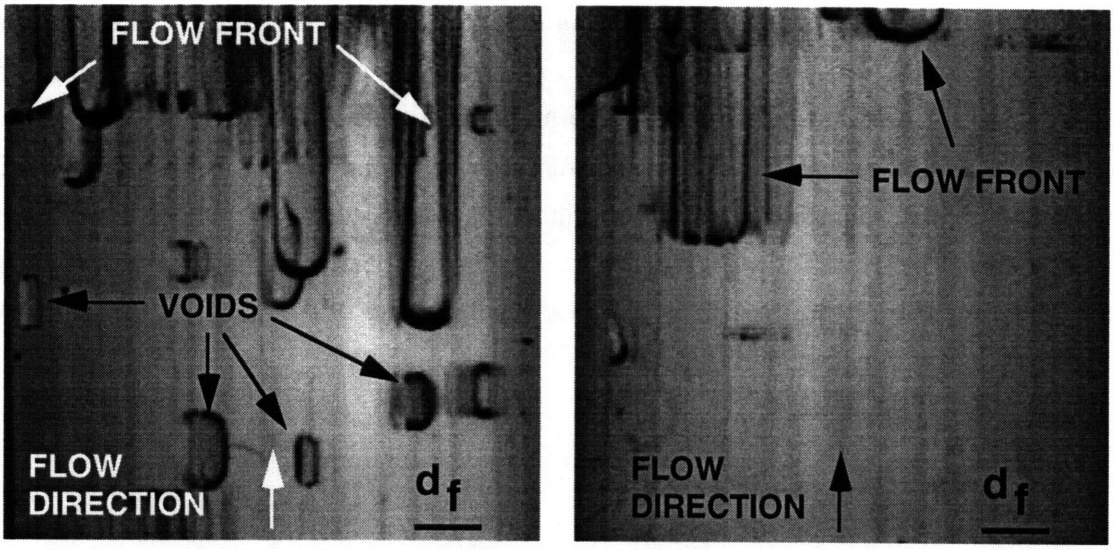


Figure 6.7 Fiber-scale void formation: (a) $t = 0$, (b) $t = 1/30s$, (c) $t = 2/30s$, (d) $t = 3/30s$.



(a) (b)

Figure 6.8 Fiber-scale void formation regimes: (a) $U < U_c$, (b) $U > U_c$.

As indicated qualitatively in Figure 6.8, the observed amount of void entrapment suggested a correlation with the speed of fiber wetting. Based on the arguments in Chapter 4 and results reported for random fabrics [2,6,7], this correlation should, in fact, be with respect to some measure of the relative importance of the local viscous and surface tension forces (i.e., some capillary number, Ca). This correlation should also be consistent with the conditions for flow front fingering since the actual entrapment mechanism was seen to be dependent on finger growth and local instability. If correct, the condition for stable fingering developed in Chapter 4 (Eq. 4.18) should then help to explain the correlation suggested by the result indicated in Figure 6.8.

In order to facilitate investigation of the formation of voids and the importance of fingering and capillary number, Ca, the impregnation experiments were modified from constant inlet pressure to constant flow rate processes. This enabled a greater number of observations of flow at a given wetting speed (and corresponding Ca) during each experiment. Varying the speed of the global front (by varying the mass flow rate into the model) generated capillary numbers within the interval $10^{-5} < Ca < 10^{-2}$. Measurements were then made of the finger length, z_f, the standard deviation of in individual finger lengths, σ_{z_f}, and the extent, Δz_f, of the region in which flow front fingering occurred. Figure 6.9 illustrates the measurement geometry. A computer program was written to analyze images of the flow front in order to obtain the fingering data. The program is described and listed in Appendix D.

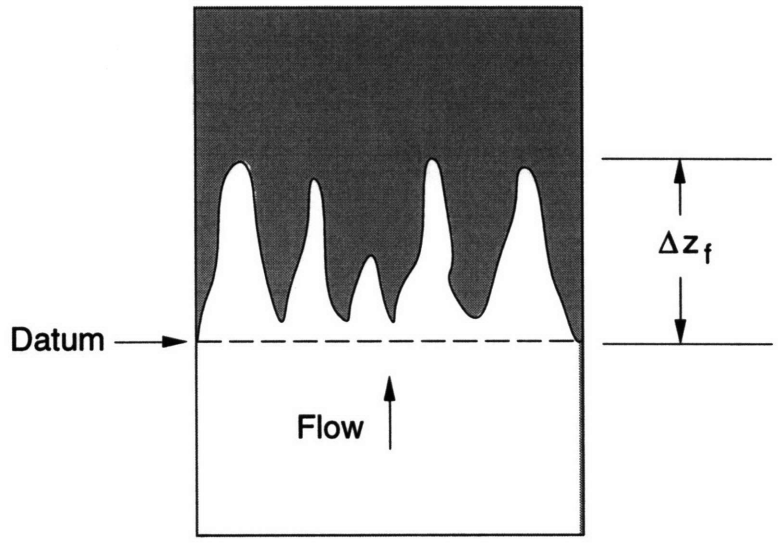
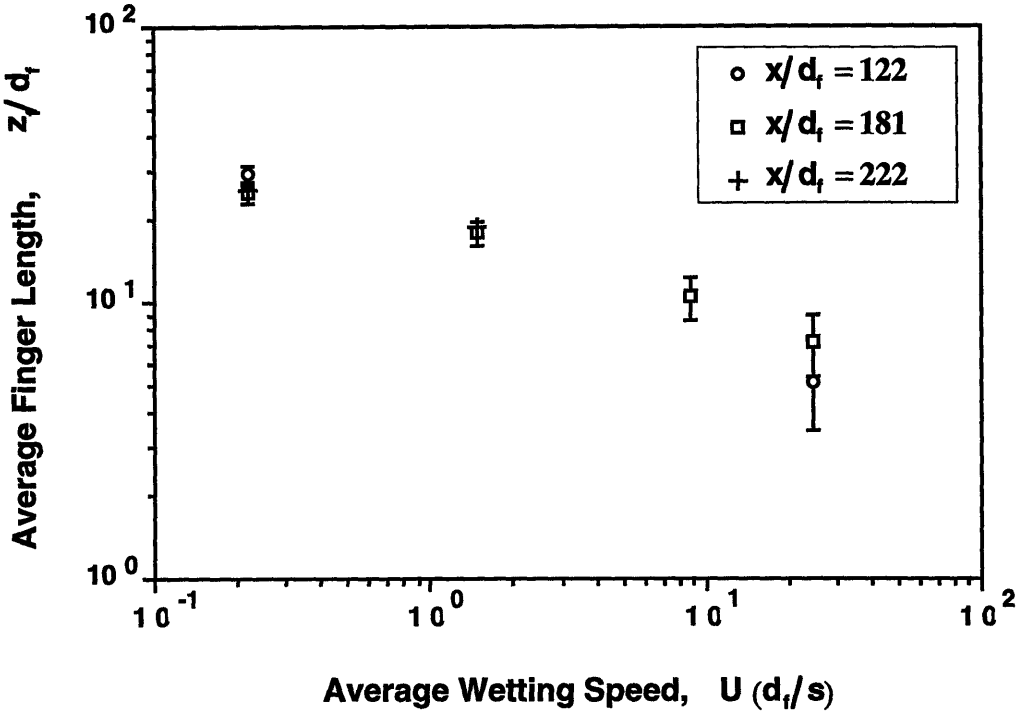


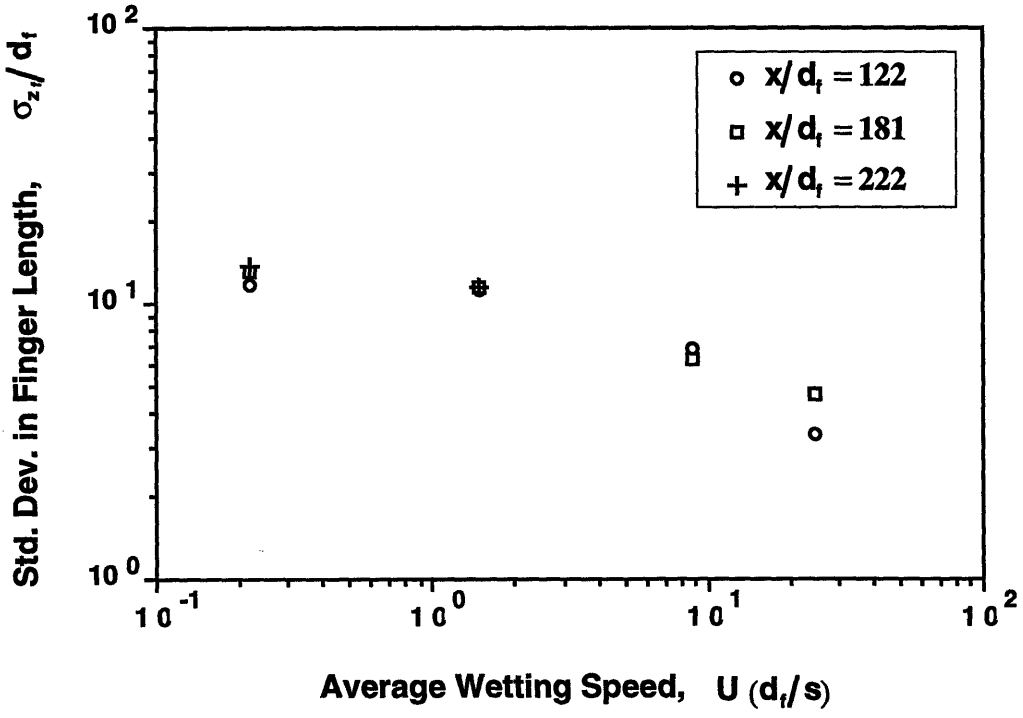
Figure 6.9 Definition of wetting range for analysis of fiber-scale fingering. The datum for the measurements corresponds to the bulk wetting front.

Figure 6.10 shows the fingering vs. wetting speed behavior observed in one sample at various distances from the flow inlet (measured in terms of fiber diameter). This result is typical. Since $\sigma_{z_f} \sim z_f$ for all wetting speeds, U, considered and measurement of the fingering range, Δz_f, was much quicker and gave results of the same order of magnitude, analysis of the fingering behavior was subsequently based on Δz_f. Figure 6.11 shows the resulting measurements of the fingering range, Δz_f, vs. wetting speed, U, for samples with various average fiber volume fractions, V_f. The uncertainty in the values of Δz_f is $\pm 1.7 d_f$.

In order to compare the measured fingering behavior with that predicted using Eq. 4.23, the fingering data should be plotted against some characteristic capillary number, $Ca = \mu U / \gamma$, (based here on the average wetting speed U). Figures 6.11-6.13 show the comparison for the data plotted in Figure 6.10. The darkened circles in Figures 6.11-6.13 represent measured values corrected for gravity effects using Eq. 4.25.



(a)



(b)

Figure 6.10 Flow front fingering vs. average wetting speed, U : (a) average finger length and (b) standard deviation in finger length.

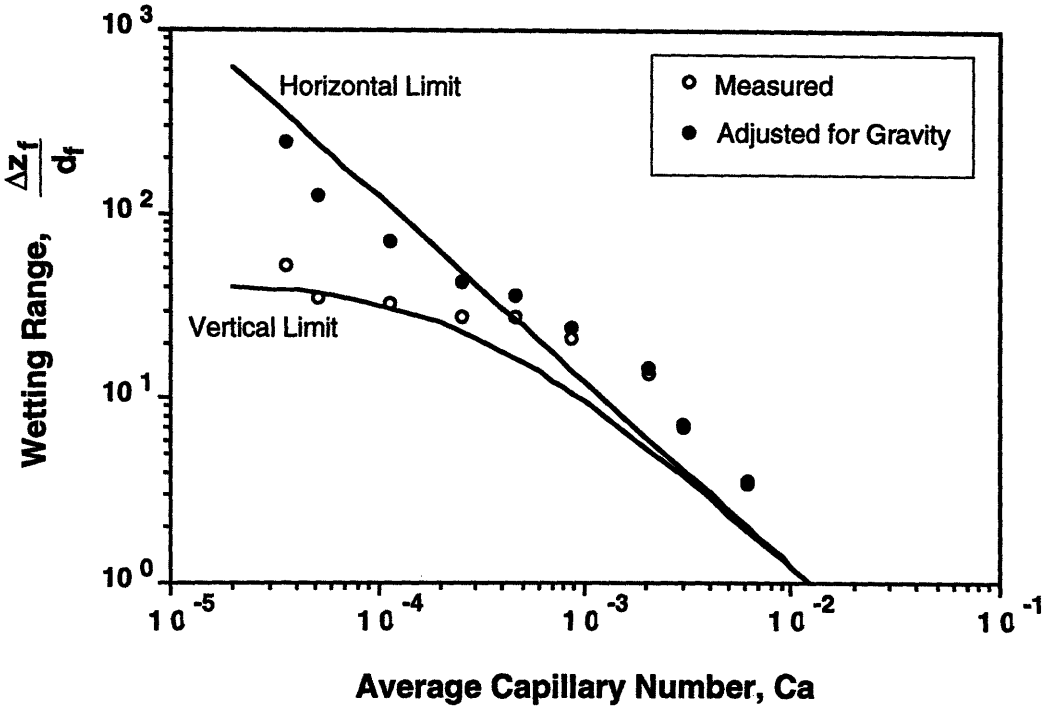


Figure 6.11 Comparison of measured flow front fingering vs. average capillary number, Ca , and order-of-magnitude estimate for $V_f = 0.59$

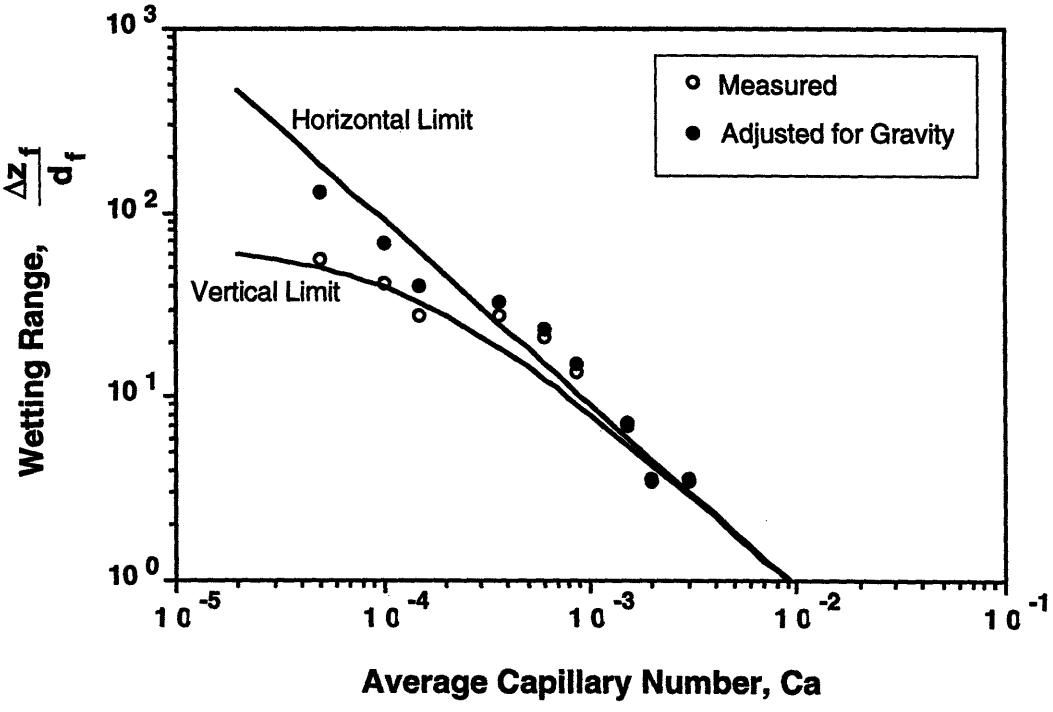


Figure 6.12 Comparison of measured flow front fingering vs. average capillary number, Ca , and order-of-magnitude estimate for $V_f = 0.67$

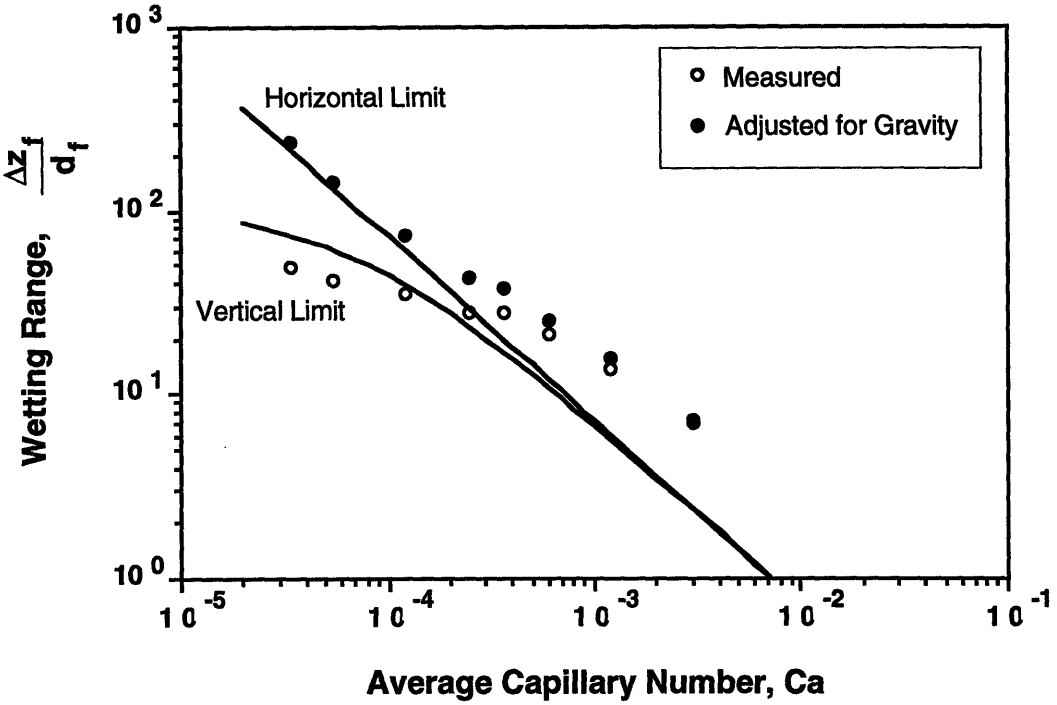


Figure 6.13 Comparison of measured flow front fingering vs. average capillary number, Ca , and order-of-magnitude estimate for $V_f = 0.73$

6.3 References

1. Molnar, J. Trevino, L., and Lee, L., 1989, "Mold Filling in Structural RIM and Resin Transfer Molding" *Proceedings of the 44th Annual Conference, Composites Institute, The Society of the Plastics Industry.*

2. Patel, N., and Lee, L., 1995, "Effects of Fiber Mat Architecture on Void Formation and Removal in Liquid Composite Molding", *Polymer Composites*, **16**(5):386-399.

3. Chan, A., and Morgan, R., 1992, "Modeling Preform Impregnation and Void Formation in Resin Transfer Molding of Unidirectional Composites", *SAMPE Quarterly*, **23**:48-52.

4. Ueda, S., 1993, *Visualization of the Flow in Complex Shapes Made by Resin Transfer Molding*, S.M. thesis, Massachusetts Institute of Technology.

5. Bear, J., 1972, *Dynamics of Fluids in porous Media*, Ch. 9, Dover, New York.

6. Mahale, A., Prud'Homme, R., and Rebenfeld, L., 1992, "Quantitative Measurement of Voids Formed During Liquid Impregnation of Nonwoven Multifilament Glass Networks Using an Optical Visualization Technique," *Polymer Engineering and Science*, **32**(5):319-326.

7. Mahale, A., Prud'Homme, R., and Rebenfeld, L., 1993, "Characterization of Voids Formed During Liquid Impregnation of Non-Woven Multifilament Glass Networks as Related to Composite Processing", *Composites Manufacturing*, **4**(4):199-204.

8. Larson, R., Scriven, L., and Davis, H., 1977, "Percolation Theory of Residual Phases on Porous Media", *Nature*, **268**(4):409-413.

9. Larson, R., Davis, H., and Scriven, L., 1981, "Displacement of Residual Nonwetting Fluid From Porous Media", *Chemical Engineering Science*, **36**:75-85.

10. Princen, H., 1969, "Capillary Phenomena in assemblies of Parallel Cylinders. I. Capillary Rise Between Two Cylinders", *Journal of Colloid and Interface Science*, **30**(1):69-75.

11. Princen, H., 1969, "Capillary Phenomena in assemblies of Parallel Cylinders. II. Capillary Rise in Systems with More Than Two Cylinders", *Journal of Colloid and Interface Science*, **30**(3):359-371.

12. Princen, H., 1970, "Capillary Phenomena in assemblies of Parallel Cylinders. III. Liquid Columns Between Horizontal Parallel Cylinders", *Journal of Colloid and Interface Science*, **34**(2):171-184.

13. Chen, J., Backes, D., and Jayaraman, K., "Dynamics of Binder Displacement in Liquid Molding", *Polymer Composites*, **17**(1):23-33.

14. Levine, S., Reed, P., and Shutts, G., 1977, "Some Aspects of Wetting/Dewetting of a Porous Medium", *Powder Technology*, **17**:163-181.

15. Ng, K., Davis, H., and Scriven, L., 1978, "Visualization of Blob Mechanics in Flow Through Porous Media", *Chemical Engineering Science*, **33**:1009-1014.

16. Wilkinson, D., 1986, "Percolation Effects in Immiscible Displacement", *Physical Review A*, **34**(2):1380-1391.

7 FIBER-SCALE FLOW SIMULATION

The results of computations using the model outlined in Chapter 5 are presented and discussed. Calculations were made to simulate the fiber-scale flow in model fiber arrays. The goal of the numerical simulations was to approximate the behavior observed in the experiments in order to develop a tool for understanding how variability at the fiber scale affects void formation within typical fiber preforms used in advanced composites applications. In addition, this objective was to be accomplished using modest computer resources, a workstation such as might be found on an engineer's desk, as opposed to a supercomputer [1] or a series of processors working in parallel [2,3]. Axial flow in ideal arrays of fibers was modeled in order to evaluate the performance of the computer model. Unit cell variability was introduced by adding a characteristic waviness to the fibers in the model. Initial results presented here show that this variability led to entrapment of voids within the unit cell based models.

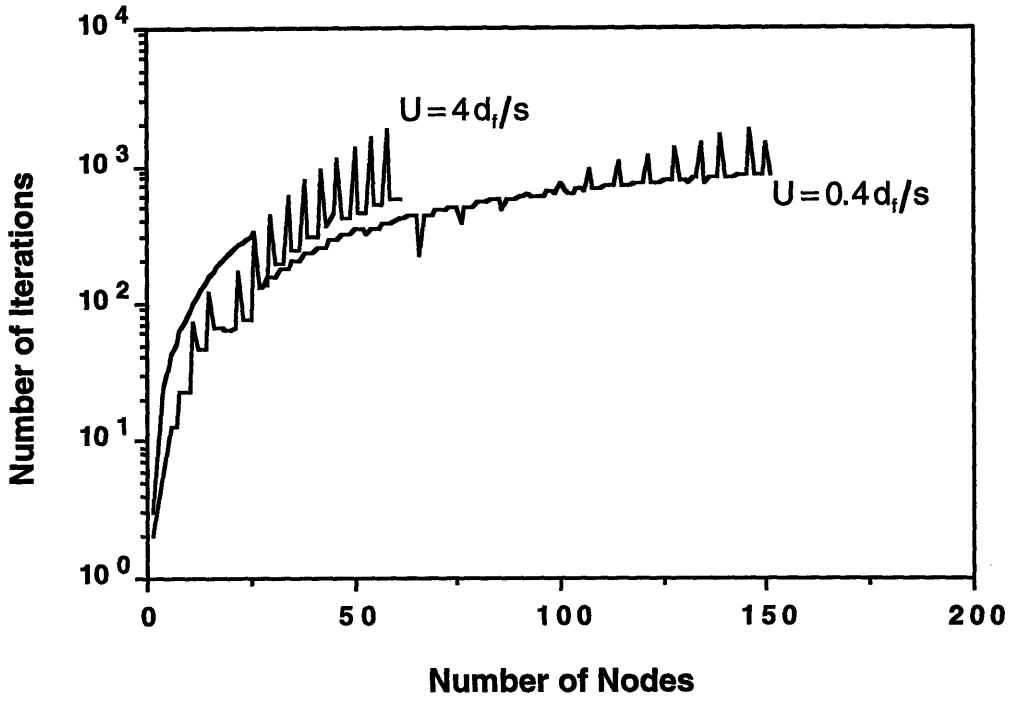
7.1 Flow in Ideal Fiber Arrays

Using symmetry, flow in an ideal array (i.e., hexagonal or square packing of identical, straight fibers) can be modeled using only one unit cell, such as the cell shown in Figure 5.7(b). This greatly reduces the size (and, consequently, the required solution time) of the model. The results reported here are for unit cells in hexagonally packed arrays of various fiber volume fractions. Table 7.1 summarizes the cell geometries considered. The models each consisted of 400 control volumes, corresponding to a length of 100 fiber diameters and an axial increment of one fiber diameter.

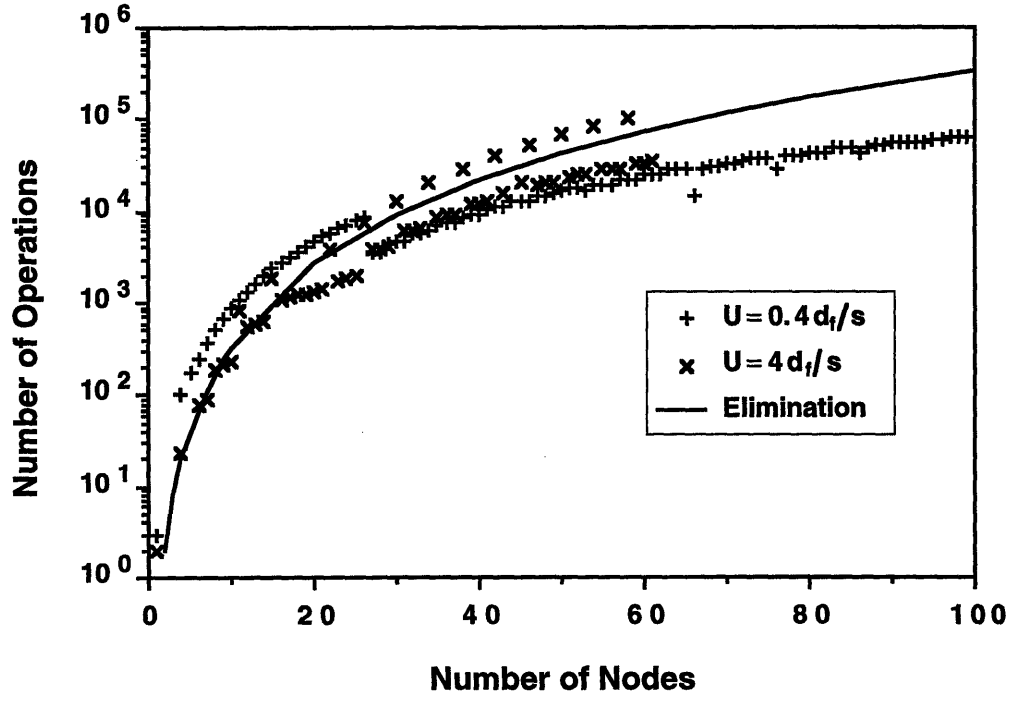
Although the resulting models of the fiber bundle take up relatively little memory, the resulting efficiency of the pressure solution at each time step was seen to need improvement. Figure 7.1 illustrates the convergence behavior of the model's equation solver. The number of iterations required for the pressure solution to converge is plotted against the number of nodes within the solution domain (i.e., all nodes i such that $sat_i = 0$) for various wetting speeds, U, for $V_f = 0.67$ in Figure 7.1(a). Closer

Table 7.1 Model Cell Parameters

V_f	δ/d_f	V_{gap}/V_{pore}
0.59	0.240	0.16
0.67	0.163	0.09
0.73	0.115	0.05
0.80	0.065	0.02



(a)



(b)

Figure 7.1 Convergence of pressure solution for $V_f = 0.67$: (a) number of iterations vs. number of nodes within solution domain for various average wetting speeds, U, (b) comparison of estimated operation count with elimination.

inspection of the convergence data indicated that the spikes in the plots correspond to time steps when the newest control volume added to the solution domain represented a pore in the unit cell. This can be seen by noting that the curves are smooth as the model transitions from its initial to steady state since this involves only the addition of identical gap elements. The curves then begin oscillating as the advancing flow front incorporates the three gap elements (which because of the perturbations introduced in packing geometry are added in succession) and then the next pore element which maintains the steady-state front configuration. As indicated in the third column of Table 7.1, the discrepancy in node volumes was significant in all cases considered with the disparity in node volumes increasing with increasing fiber volume (as would be expected since the fiber separation will be decreasing). A more uniform mesh would, most likely, decrease the magnitude of the observed spikes in the iterations versus number of nodes plot since the resulting changes in the solution domain with each time step would be more uniform.

A more meaningful metric for evaluating the model's equation solver would be a comparison of the computational "cost" of the current algorithm with some benchmark. Such a metric would be the number of operations required to solve the system of n simultaneous equations (one for each full node at time t), which can be estimated using the product of n and the number of iterations required to converge within the specified solution tolerance. Figure 7.1(b) shows the comparison of the approximate operation count for the first 100 nodes in Figure 7.1(a) and an estimate [4] of the number of operations required for elimination on a system with an $n \times n$ coefficient matrix with half bandwidth, w ($\#op's \approx \frac{1}{3}w(w-1)(3n-2w+1)$, in general, and $\#op's \approx \frac{1}{3}n(n-1)(n+1) \approx \frac{1}{3}n^3$ when the coeficient matrix is full, or $w = n$). Figure 7.1(b) shows the estimate for elimination on a full coefficient matrix. The comments made above regarding spiking in the convergence data for the model apply here also.

The fact that the iterative solution used in the computer model is so close to the estimate for elimination should not be surprising. Addition of the third dimension to the model geometry results in a coefficient matrix that, although sparse, has a very broad bandwidth. Furthermore, introduction of the pressure- and saturation-dependent coefficients, $a_{i,j}$, into the conductance matrix, **H**, by way of Eq. 5.59 makes the system of equations nonlinear. This said, however, improvement of the pressure solver in the model should be possible with judicious discretization of the model geometry and selection of solution tolerances.

As shown in Figure 7.2, the calculated front profile, characterized by the wetting range Δz_f (which in the case of the ideal arrays considered is the same as the average finger length), would converge to a limiting value as a function of the wetting speed, U (non-dimensionalized by the fiber diameter, d_f). The results are typical. The initial "ramp-up"

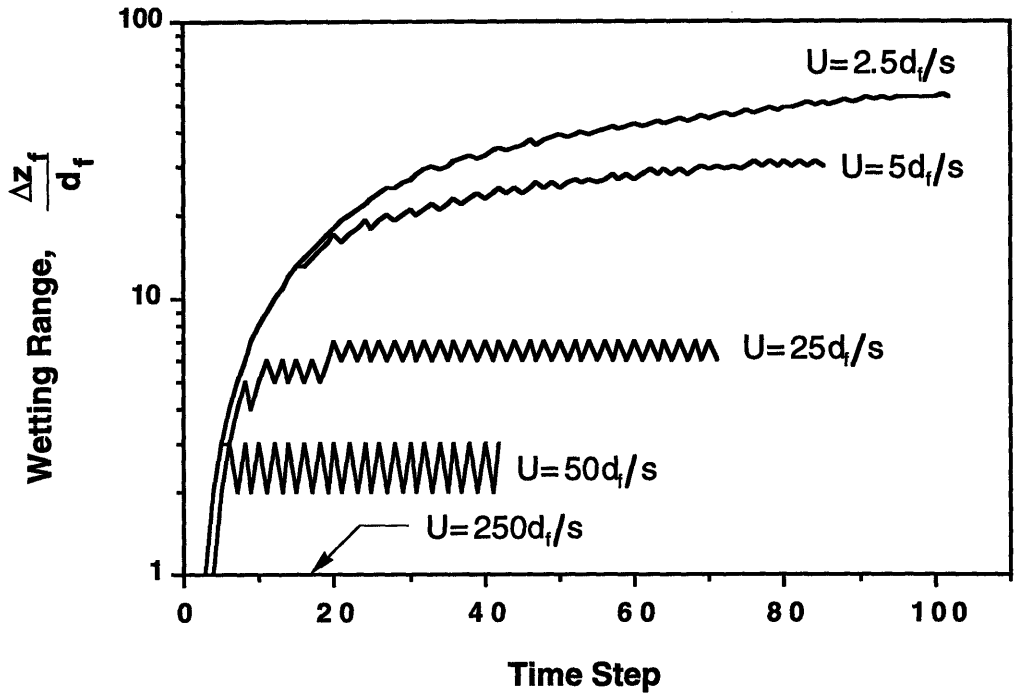
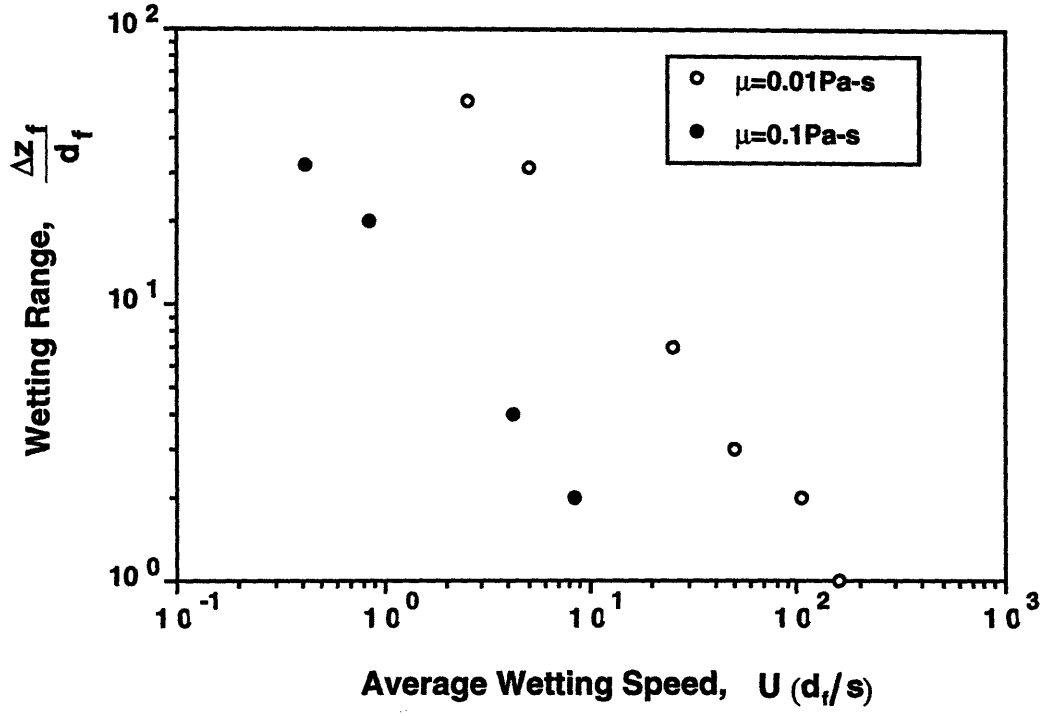


Figure 7.2 Convergence of calculated flow front fingering vs. average wetting speed, U, for $V_f = 0.67$.

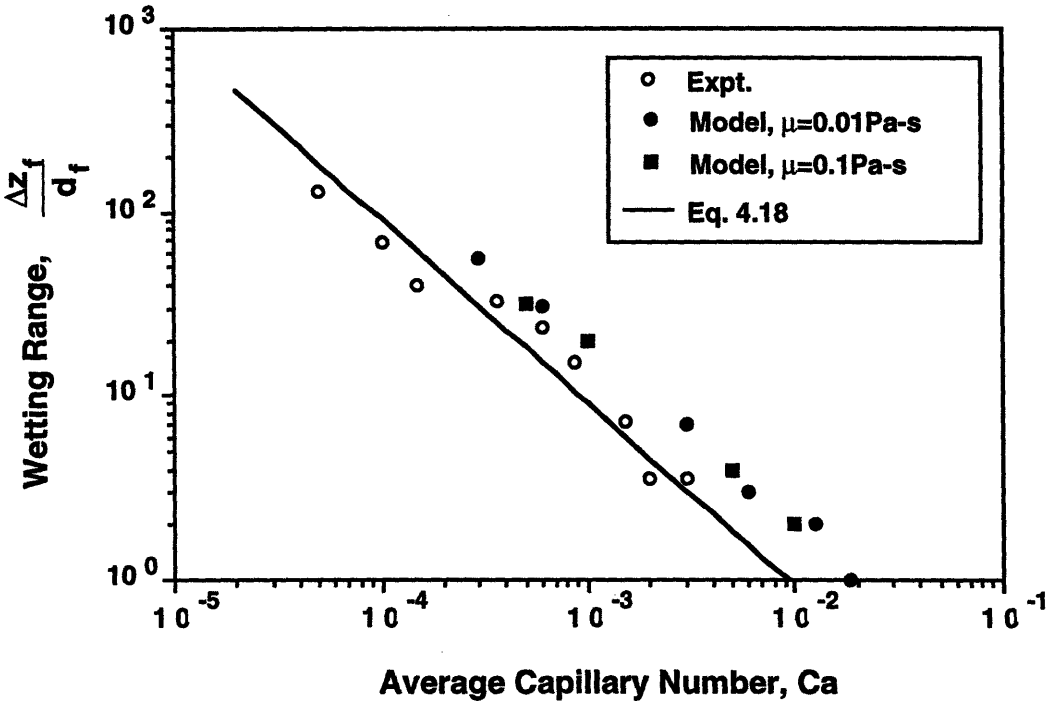
corresponds to the model's transition from the initial condition of all $sat_i = 0$ at $t = 0$ to the steady state configuration. The oscillations around the steady state values are due to the discretization of the model (i.e., the front can advance only in one-diameter increments).

Plotting the steady state wetting range versus the wetting speed in Figure 7.2, produces the plot shown in Figure 7.3(a). For comparison, a similar set of data for the identical case with a more viscous fluid (viscosity ten times greater than the case shown in Figure 7.2) is plotted on the same graph. As shown in Figure 7.3(b), the two curves collapse into one when plotted against the capillary number, $Ca = \mu U/\gamma$. This result is unsurprising on the basis of the arguments behind the development of Eq. 4.18, which is also plotted in Figure 7.3(b) for comparison. Consequently, fingering results for the other cases are presented in terms of the capillary number, Ca, in Figures 7.4 to 7.6. Experimental values using the model fibers, corrected for the effect of gravity using Eq. 4.25, are also plotted for comparison.

The values of the various estimates for the wetting range are seen to be within the same order of magnitude. The discrepancies in the various values can be attributed to discretization of the computer model, the assumptions on pore space geometry in the development of Eqs 4.18 and 4.25, and on the average nature of the measured values of the wetting range, Δz_f.



(a)



(b)

Figure 7.3 Calculated flow front fingering for $V_f = 0.67$: data plotted against (a) average wetting speed, U, and (b) average capillary number, $Ca = \mu U/\gamma$.

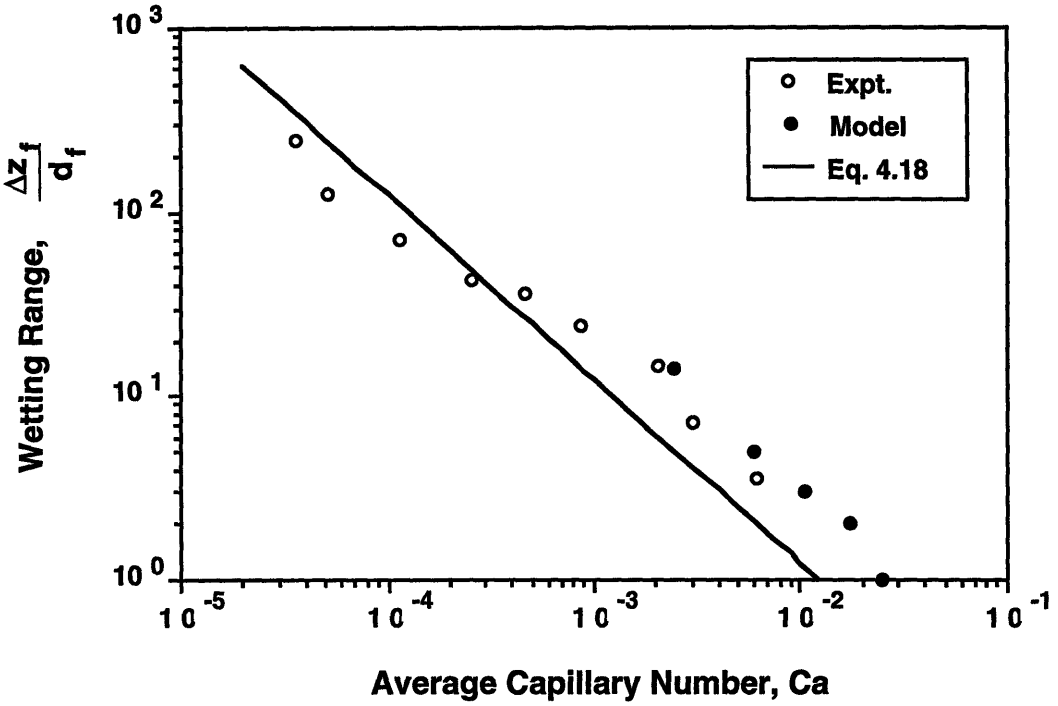


Figure 7.4 Comparison of calculated and measured flow front fingering vs. capillary number, Ca, and order-of-magnitude estimate for $V_f = 0.59$.

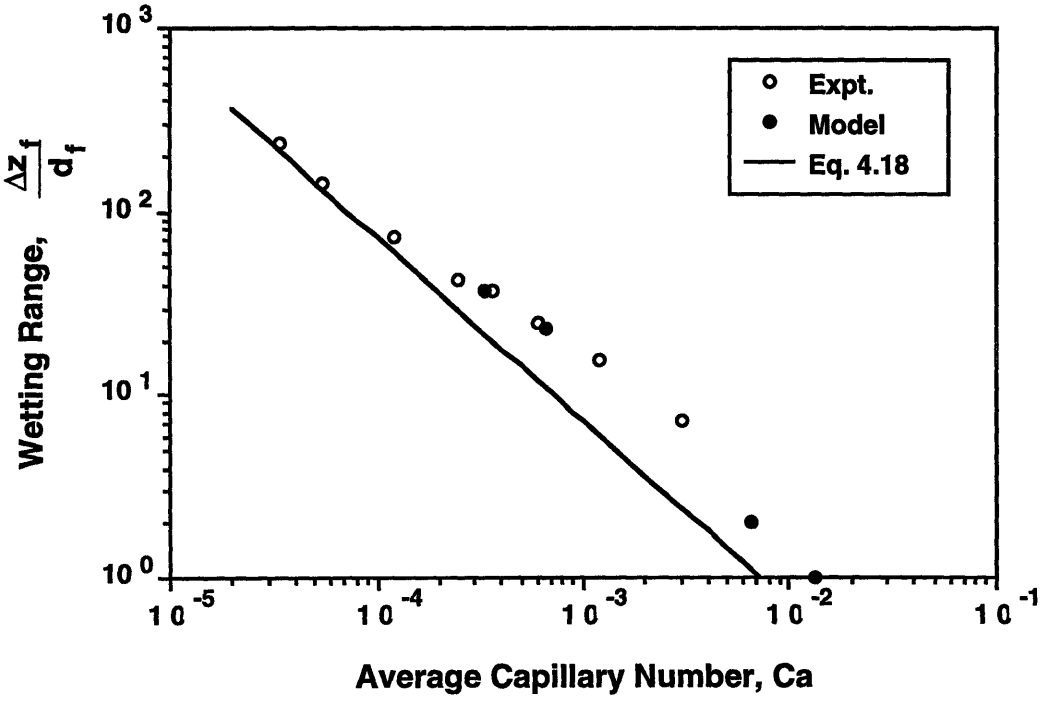


Figure 7.5 Comparison of calculated and measured flow front fingering vs. capillary number, Ca, and order-of-magnitude estimate for $V_f = 0.73$.

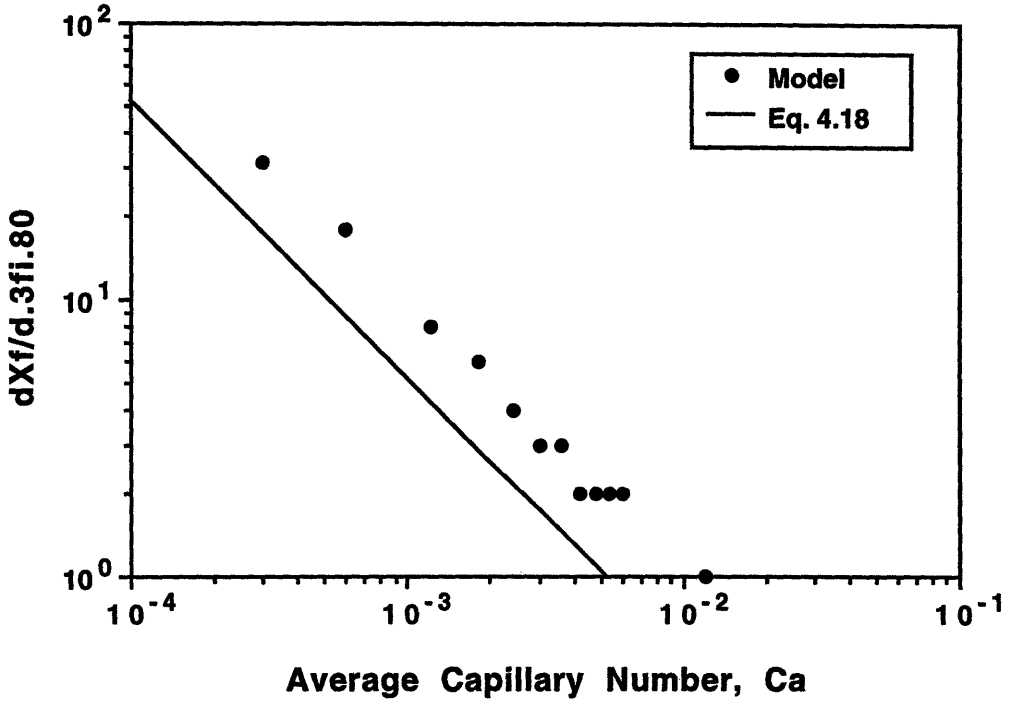


Figure 7.6 Comparison of calculated flow front fingering vs. average capillary number, Ca , and order-of-magnitude estimate for $V_f = 0.80$.

7.2 Fiber-Scale Void Formation

None of the simulations of flow within ideal unit cells created conditions that led to void entrapment. In the ideal arrays (and for the range of Ca considered), the front configuration would always tend to the steady state configuration characterized by formation of fingers within the inter-fiber gaps followed by flow within the adjacent pore. This result should be expected, since (as modeled using Eq. 5.5, the Young-Laplace equation) the effect of the capillary pressure at the fluid surface is inversely proportional to the local surface curvature. In an ideal physical system, axial flow within a gap will be favored over transverse flow from the gap to the adjacent pore. Axial variation of the gap height (and, therefore, the curvature of the advancing meniscus) will be small compared to the tranvsverse variation in the inter-fiber channel. For flow from the gap to the pore, the curvature of the adjacent fibers will lead to a continuously decreasing meniscus curvature and, therefore, a decreasing interfacial pressure drop to drive the flow.

The current control volume model does not reflect this effect. The radii of curvature for the axial and transverse flows within the gap are both assumed to be equal to half the minimum fiber separation, δ_o, in effect overpredicting the tendency for transverse flow from gaps to pores. In spite of this, however, no voids were formed in simulations on ideal arrays. Consequently, in order to simulate void entrapment in the model, the ideal geometry was perturbed as outlined in Section 5.3.1.

Table 7.2 summarizes the cell parameters for one set of models generated for various nominal values of the perturbation factor, ε, (defined in Eq. 5.9) for a unit cell with nominal fiber volume fraction, $V_f = 0.67$. Notice that increasing the perturbation over the nominal fiber separation ($\varepsilon > 1$) decreased the average fiber volume fraction while also increasing the variability in fiber separation, as expressed by the standard deviation shown in the fourth column of Table 7.2. The opposite effect is seen (for this set of models) for smaller perturbations ($\varepsilon < 1$). Figure 7.7 illustrates the resulting periodic nature of the average, cross-section fiber-volume fraction along the length of two models, $\varepsilon = 0.5$ and $\varepsilon = 1$. Figure 7.8 shows the resulting variation in node volume along one period of cross-section variation for the model with $\varepsilon = 1$.

Simulations, such as those for the ideal cells, were run on these models for various Ca. Void entrapment was observed only in the models with $\varepsilon \geq 1$. The voids were located within the pores and were formed when fluid flowed transverse to the fibers from adjacent gaps. Figure 7.9 compares the development of the wetting range in the model with $\varepsilon = 1$ and the ideal unit cell. As shown in the figure, one of the consequences of entrapping the void is that the extent of the wetting range, Δz_f, is temporarily reduced. With the local extent of the fingering reduced, the probability of a new void forming is also reduced since, as noted in the experimental observations, shorter fingers will typically have less opportunity for cross flow.

Table 7.2 Cell Parameters for Perturbed Model

ε	$\overline{V}_f$	$\overline{\delta}_o/d_f$	$\sigma_{\frac{\delta_o}{d_f}}$
0.01	0.67	0.163	~0
0.50	0.69	0.149	0.026
0.75	0.69	0.146	0.035
1.00	0.64	0.193	0.044
1.50	0.65	0.177	0.074

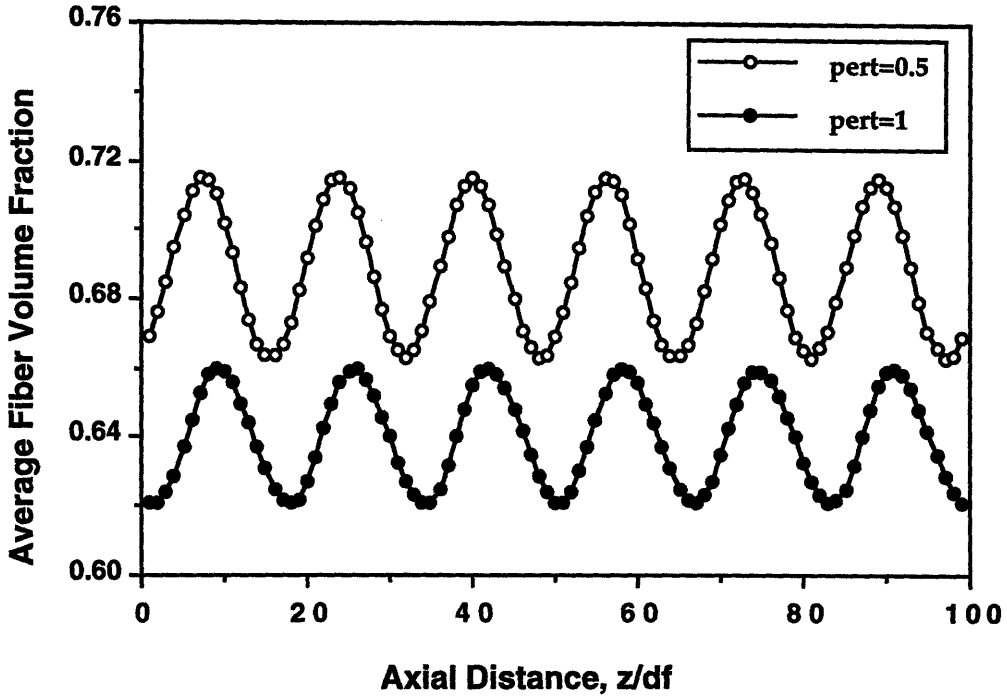


Figure 7.7 Variation of average, cross-section fiber volume fraction along length of model for $\varepsilon = 0.5$ and $\varepsilon = 1$.

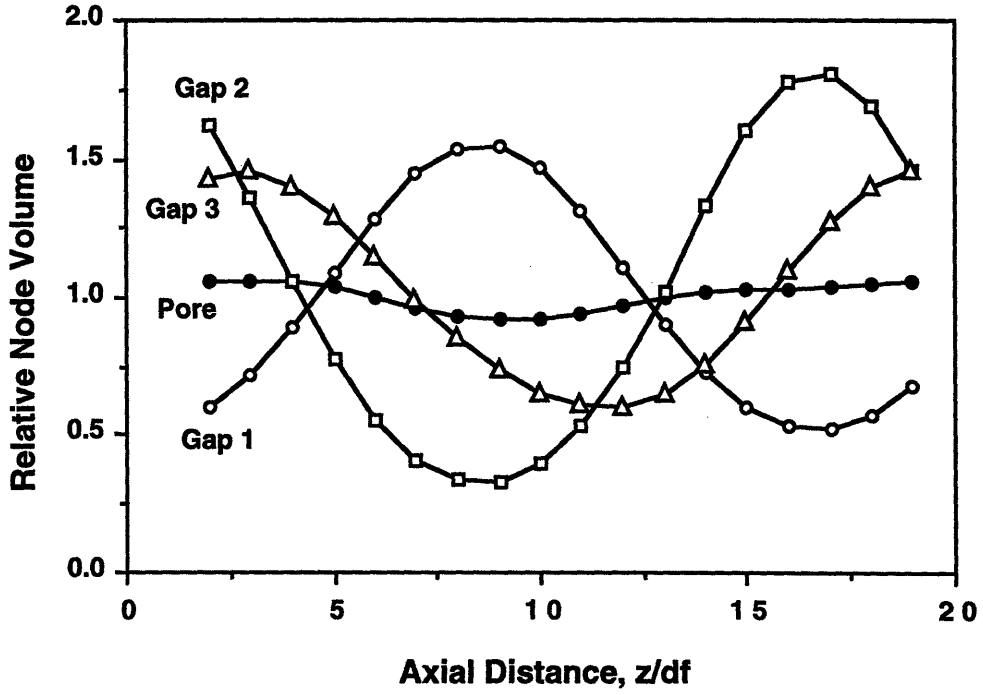


Figure 7.8 Variation of node volume relative to mean along one period of fiber waviness for $\varepsilon = 1$.

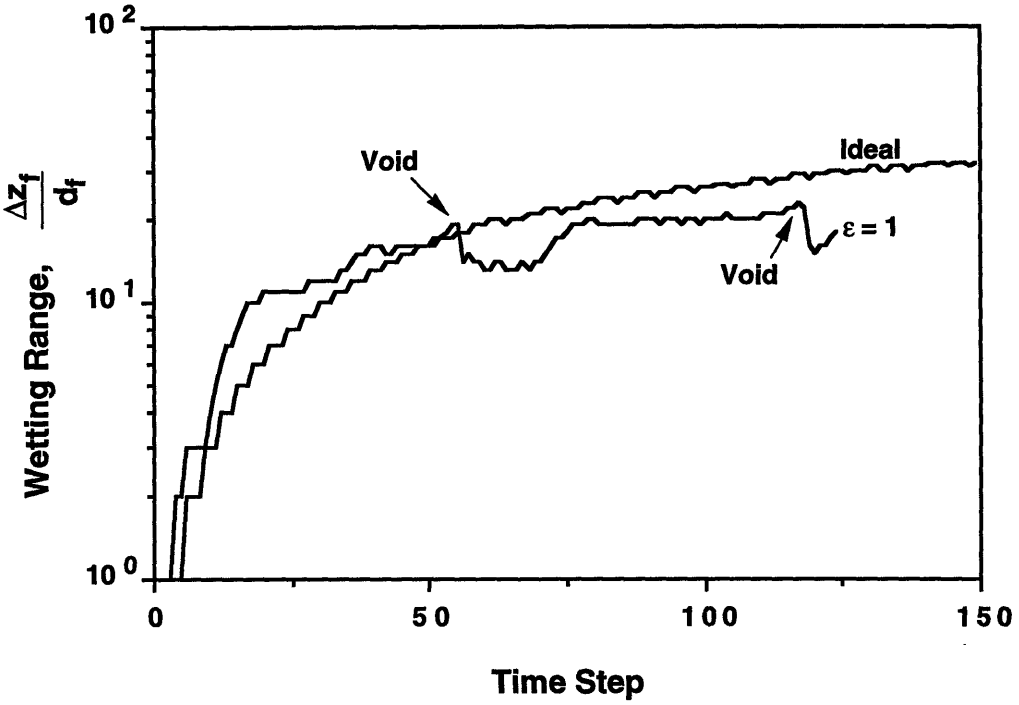


Figure 7.9 Comparison of flow front development in perturbed model with $\varepsilon = 1$ and ideal unit cell for $V_f = 0.67$ and $Ca = 5 \times 10^{-4}$.

7.3 References

1. Fickie, K., 1994, "Manufacturing Simulation", *AHPCRC Bulletin*,**4**(3), http://www.arc.umn.edu/html/publications/bulletins/v4n3/v4n3.html

2. Cruz, M., 1993, *A Parallel Monte-Carlo Partial-Differential-Equation Procedure for the Analysis of Muticomponent Random Media*, PhD. dissertation, Massachusetts Institute of Technology.

3. Ghaddar, C., 1995, *Parallel Analytico-Computational Methods for Multicomponent Media: Application to Thermal Composites and Porous-Media Flows*, PhD. dissertation, Massachusetts Institute of Technology.

4. Strang, G., 1988, *Linear Algebra and Its Applications*, 3rd ed., Harcourt Brace Jovanovich, Publishers, San Diego, p.55.

8 CONCLUSIONS

The objective of this thesis was to reduce the trial-and-error in fiber impregnation tooling and process design by developing tools with which designers can investigate the effects of variability within the fiber preform. In order to meet this goal, a method to directly observe and quantify the flow within the preform was developed. Based on matching the refractive indices of the fibers and a simulated resin, this flow visualization method can be combined with photometric measurements and image processing to quantitatively describe the impregnation process at the various length scales within the preform. Examples of flow visualization results at the fiber and tow scales were presented to illustrate the method. In order to generalize the experimental results, a computer model of preform impregnation at the fiber and tow scales was developed. The model, based on an approximate representation of the inter-fiber pore space as a three-dimensional network of control volumes, provides a design tool that can be used to study various assumptions on the causes and effects of variability within the preform at this scale. The results of a study of void entrapment during unidirectional flow at the tow scale were presented as an example.

The main process related results are summarized below. In addition, possible areas of further research and implications on processing strategies for void-free parts are discussed.

Fiber and Tow-Scale Void Entrapment

Observation of flow within typical preforms indicated that the basic mechanisms for void formation were dependent on both the length scale of interest and the structure of the fabric. The length scale was seen to affect both the relative importance of forces acting on the advancing fluid surface (i.e., viscous, surface tension, and gravity forces) and the types of preform variability that were important in the development of voids. The specific structure of the fabric was seen to have a dramatic effect on the complexity of the flow within the fabric: a generalization being that the more complex the flow, the greater the probability for void entrapment. At the fiber scale, two types of flows were observed. These flows were globally axial flow along the fiber axes and a combination of axial and radial flow within discrete fiber bundles.

Globally axial flow within unidirectional preforms was the least complicated flow observed. In this case, nonuniformity of the pore space created by the fibers led to the formation of fingers at the advancing flow front. As discussed in Chapter 4, the length of

the fingers was seen to be determined by the degree of preform variability and the balance among the viscous, surface tension, and gravity forces acting on the advancing interface. Fingers were shown to form even in ideal arrays of fibers because of the resulting variability of the pore space transverse to the fiber axes. An order of magnitude analysis and experiments showed that finger length scaled inversely with the capillary number, $Ca = \mu U/\gamma$.

Observation of the formation of individual voids at this scale indicated that fingers extending beyond a certain length would become unstable, forming secondary surface perturbations along the length of the finger. These perturbations could then dissipate or grow and form transverse liquid bridges with adjacent fingers and entrap air. Computer simulation of the process indicated that the formation of the liquid bridges between adjacent fingers was dependent on finger length (and, therefore, the factors affecting finger growth) and the amount of variability along the major fiber axes (i.e., in the direction of global flow).

Flow in fabrics with discrete tows was seen to be composed of two types, an approximately one-dimensional flow in the gaps between adjacent tows and a combined axial and radial flow within the individual tows. The inter-tow flow will be discussed below since it is characterized by the tow length scale. The flow within the tow, on the other hand, is characterized by both the fiber and tow length scales.

For global flow along the major tow axis, the resulting flow consisted of a predominantly axial flow in the center of the tow (as in the unidirectional flow discussed above) and a combination of radial and axial flow toward the outer surface of the tow. As with the predominantly axial flow, the nonuniformity of the pore space within the tow was seen to lead to the formation of flow front fingering. Now, however, both axial and radial fingers were observed. Since the permeability of the tow is typically much greater in the axial rather than the radial direction, the radial fingers were seen to dissipate in the axial direction. This resulted in local flows that were either upstream or downstream relative to the global flow.

Voids were seen to be formed by two mechanisms in this case. The first, primarily toward the center of the tow, was essentially the same as observed in the predominantly axial flow in the uniaxial fiber beds. The second mechanism was caused by the dissipation of adjacent radial fingers. Voids would be formed when axial fingers moving in opposite directions would entrap pockets of air. As with the previous case, the entrapment of voids was most likely due to axial variations in the pore space leading to axial variability in local values of the tow's radial permeability.

At the tow scale, nonuniformity in the gaps between adjacent tows or in the gap formed between the fabric and the tooling were seen to cause fingering at the flow front. Since the flow in these gaps was seen not to be exactly one-dimensional, voids would form when advancing fingers would encounter obstacles, such as cross-stitching, which would then lead to a backflow and the entrapment of air pockets. Since the scale of gap variability could be much larger than within the inter-fiber pores, the voids formed at this length scale could be much larger than those formed between individual fibers. Further comparison of these voids with the fiber-scale voids showed that the fiber-scale voids were cylindrical (with diameter on the order of the inter-fiber pore diameter) while the tow-scale voids were either spherical (when located between adjacent, discrete tows) or irregularly shaped (when located between the preform surface and the tooling).

Void removal

Although the mechanisms for void formation were seen to be different at the fiber and tow scales, the same basic mechanism was seen to be at work for void removal. As discussed in Chapter 4, an effective void mobilization strategy must recognize that

- variability within the preform results in local constrictions in the pore space that trap voids in place. At the fiber scale this variability can be due to fiber waviness, variations in fiber diameter, and nonuniform fiber packing. At the tow scale, preform variability can be generated by nonuniform tow packing and cross-stitching and veil materials that hold the tows in place. In addition, the packing of round fibers and tows generates variability in the pore space transverse to the fiber and tow axes.

- mobilization of a void through a constriction requires increasing the local pressure gradient to overcome the surface tension resisting deformation of the void. Large voids (i.e., in the direction of the mobilizing pressure gradient), low interfacial surface tension, and minimized preform variability in the direction of the mobilizing pressure gradient all facilitate void mobilization.

- an important consequence of the difference in the magnitudes of the fiber and tow length scales is that mobilization of voids at the tow scale is easier than at the fiber scale.

Processing Implications

The void entrapment and mobilization behavior summarized above has significant implications for the production of void-free parts. Since the type of flow observed during the process is also dependent on the preform structure and orientation relative to the direction of global flow, general processing rules cannot be very specific. Nonetheless, based on the relative ease of mobilizing voids at the various length scales within the preform, it can be stated that a general processing strategy should avoid fiber-scale void

entrapment and, if necessary, should employ a void flushing step to remove voids occurring at the tow and preform-tooling gap scales.

For axial flow within uniaxial fabrics without distinct tows, experiments and calculations showed that fiber-scale voids were minimized when the local capillary number, Ca, was above a certain critical level. This critical value of Ca was seen to be dependent on both fluid properties and the amount of variability within the inter-fiber pore space. In terms of process variables, higher impregnation rates should be favored in order to avoid fiber-scale voids. As discussed in Section 4.1.2, however, the impregnation rate should be below the upper limit imposed by considerations of meniscus stability. An additional benefit of using a high impregnation rate is that larger voids formed at the scale of the preform-tooling gap will be flushed out of the mold.

For flow in fabrics with discrete tows, avoiding formation of fiber-scale voids is complicated by the combination of axial and radial flows within the tows. Here the processing challenge is to balance the axial flow within the center of the tow with the radial flow coming into the center from the inter-tow channels. In order to achieve this balance, the speed of the advancing fluid front within the adjacent channel should match the axial flow within the tow. This can be achieved by slowing the global impregnation rate. Care must be taken, however, to ensure that the axial flow within the tow generates local capillary numbers high enough to satisfy the criterion for void-free axial flow. Consequently, selection of the proper impregnation rate requires optimization of the balance between axial fingering within the tow and the tow-scale fingering within the adjacent channel.

APPENDIX

APPENDIX A Physical Properties of Test Materials

Table A.1 Determination of Fiber Refractive Index

Illumination Wavelength	632.8 nm
Interface	Fiber-Air
Index Based on Immersion	1.5537
Index based on Measured Transmittance and (7)	1.5501
Discrepancy in Refractive Index Values	- 0.23%

Table A.2 Physical Properties of Commercial Fabrics

Fabric	Type	Average/Tow V_f	Nominal Fiber Diameter	Nominal Tow Diameter
ANCAREF J600	Unidirectional	0.6/--	12 ± 1 μm	–
Bean Fiber Glass, VECTORPLY V18	Stitched, Unidirectional	0.45/0.6	15 ± 1 μm	1.5 mm
Fiber Glass Evercoat Co., Sea-Glass Cloth	0°- 90° Weave	0.5/0.7	9 μm	5 mm
BGF Industries Style 3783 8HS E-glass	8-Harness, Satin Weave	0.54/	9 μm	0.5 mm

APPENDIX B Properties of Test Fluids

Table B.1 Cauchy Equation Coefficients of Model Fluids (Wavelength in Å)

Fluid	a_o	a_1	a_2
Cargille Fluid #5095	1.531	793580	4.53897×10^{12}
Cargille Fluid #1057	1.53288	758214	2.95923×10^{12}
C_6H_5Br	1.5323	762856	3.69347×10^{12}

Table B.2 Physical Properties of Test Fluids

	Water	Polyester	C_6H_5Br	Cargille Fluid #1057	Cargille Fluid #5095
Specific Gravity	1	1.11	1.50	1.097	0.953
Viscosity (Pa•s)	10^{-3}	1	10^{-3}	0.32	10^{-2}
Surface Tension (x 10^3 N/m)	72	40	38	42	42
Contact Angle (on glass)	~0°	35°	30°	25°	25°
Refractive Index	1.3307	1.5556	1.55365	1.5537	1.5537
Refractive Index Mismatch	-14.4%	0.13%	~0	~0	~0

APPENDIX C Program Listing for Fiber-Scale Flow Analysis

C.1 Header File

```
/* NetDefs.h: Header file for simulation of fiber-scale flow */


#include "stdio.h"
#define fTol 1e-3   /* Assumed tolerance for floating point operations */
#define NC 6        /* Node coordination number (i.e., # neighbors + 1)
                       This parameter is a function of "packing" structure */
#define PI 3.1459



typedef struct FiberInfo     *fiberptr;
typedef struct PackingInfo       *packptr;
typedef struct GapInfo           *gapptr;
typedef struct FrontInfo     *frontptr;
typedef struct FaceInfo          *faceptr;
typedef struct NodeInfo          *nodeptr;
typedef struct NeighborInfo      *neighborptr;
typedef struct SourceInfo       *sourceptr;
typedef struct voidnodetype      *vnodeptr;
typedef struct GroupInfoType   *groupinfo;


struct FiberInfo {
    int       i;        /* Fiber number */
    float          x[3];          /* Coordinates of fiber center */
    float        phi[2];       /* Fiber orientation[0] and phase[1] angles */
    float          d;        /* Fiber diameter */
    fiberptr  next;
};

struct PackingInfo {
    int  cell;              /* tow sub-domain/cell (assumed triangular) */
    int  vertex[3];
    float     Az;               /* pore area normal to tow axis */
    float     L[3];             /* length of side opposite vertex[i] */
    float     angle[3];         /* Interior angle for vertex[i] */
    float     xc;            /* Coordinates of cell centroid */
    float     yc;
    float     x[3];             /* Coordinates of midpoint of side[i] */
    float     y[3];
    packptr  next;
};

struct FrontInfo {
    int         node;
    frontptr  next;
};

struct FaceInfo {
    int       F;
    int       u;
    int       v;
```

```
    float       A;
    float       L;
    float       H;
    float       Rcap;    /* Assumed radius of curvature for fluid interface */
    faceptr     next;
};

struct GapInfo {
    int         i;
    int         v1;
    int         v2;
    gapptr      next;
};

struct NodeInfo {
    int         node;
    float       x[3];
    float       V;
    float       Vf;
    float       Rh;
    float       sat;
    float       sat0;
    float       Q;
    float       P;
    float       P0;
    float       Ptemp;
    int         flag;
    neighborptr neighbor;
    nodeptr     next;
};

struct NeighborInfo {
    int         f;
    int         n;
    neighborptr prev;
    neighborptr next;
};

struct SourceInfo {
    int         i;
    float       Q;
    sourceptr   next;
};

struct voidnodetype {
    int         node;
    vnodeptr    prev;
    vnodeptr    next;
};

struct GroupInfoType {
    int         label;
    groupinfo   prev;
    groupinfo   next;
    vnodeptr    list;
};
```

```
void Step(int nXsect,float dz,sourceptr Source,frontptr Front,frontptr Next2Front,
      int inNode,int refnode,double *t,double *dt,double *Dt,nodeptr node,faceptr face,
      float surfT,char *voidfile,float *vol,float Pref,groupinfo voidgroup,int *numvoid,
      char *solndat,float Pscale,float Pwave);

void Solver(nodeptr node,faceptr face,sourceptr Source,frontptr Behind,frontptr AtBehind,
       frontptr Next2Front,frontptr AtFront,groupinfo voidgroup,int inNode,int refnode,
       float Qin,float surfT,float Pref,double t,char *solndat,float Pscale,float Pwave);


float getPc(nodeptr nptr,faceptr fptr,nodeptr node,faceptr face,float surfT,float Pscale,
       int refnode);

float getAvgPc(nodeptr nptr,nodeptr node,faceptr face,float surfT,float Pscale,int refnode);

void newH(nodeptr node,faceptr face,sourceptr Source,float surfT,float Pscale,int refnode,
      float dt);

int CheckList(int i, frontptr list);

void EdgeCheck(int i, float *a, int *El,int *NextNode, nodeptr node);
void FrontCheck(sourceptr Source,nodeptr node,faceptr face,frontptr AtFront,frontptr AtBehind,
        frontptr Behind,frontptr Next2Front,int refnode);

void Geometry(float Dnom,float beta,float dz,int nz,float visc,fiberptr fiber,
        packptr basepack,nodeptr node,faceptr face,int *nXsect,int *fXsect,
        int *inNode,int *refnode,float Pref,float pert,float deltaD,float *avgD,
        float *stdevD,float *avgVf,float *stdevVf,sourceptr XsectVf,float *avgGap,
        float *stdevGap,float *Va,float *baseGap,float *Lscale,int *numfibers,int *cellnum,
        float *Kz,float *Kr,float *Apore);

float myrandom();
int evenOdd();
float dist(float *P1,float *P2);
void sectorCentroid(int v0,int v1,int v2,float *x,float *y,float *d,float *angle,
           float *xc, float *yc);
void gapInfo(int v0,int v1,int v2,float *x,float *y,float *xc,float *yc,
        float *L,float *d,float *A,float *delta,float *P,float *Rh,float *Vf);
gapptr newGap(int i,int v1,int v2);
void addGap(gapptr gap,int i,int v1,int v2);
int checkGap(gapptr gap,int u,int v,int k,int nXsect);
gapptr   getGap(gapptr gap,int u,int v,int k,int nXsect);
float TransverseH(int f1,int f2,float minGap,float *d);
void Scaling(nodeptr node,faceptr face,float avgD,float *Apore);


fiberptr newFiber(int i,float *x,float d);
void addFiber(fiberptr fiber, int i,float *x,float d);
fiberptr getFiber(int i, float z,fiberptr fiber);
fiberptr Scan2EndFiber(fiberptr fiber);
void printFiber(fiberptr fiber);
void ReadFiberData(fiberptr fiber,char *filename,int *numfibers);
void perturbFibers(int nz,float dz,int numfibers,fiberptr fiber,float pert,
     float deltaD,float *avgD,float *stdevD,float Lscale,float avgGap);

packptr newCell(int cell);
void addCell(packptr basepack, int cell,int *vertex);
```

```
packptr getCell(int cell, packptr basepack);
packptr Scan2EndCell(packptr basepack);
void printVertex(packptr basepack);
void ReadVertexData(packptr basepack,int *cellnum,char *filename);

frontptr newFrontNode(int i);
void addFrontNode(frontptr list, int i);
void rmvFrontNode(int i, frontptr list);
void renewFrontNode(frontptr list);
void printList(frontptr list);

void NewTime(sourceptr Source,double t,double *tmin,nodeptr node,faceptr face,
        frontptr AtFront,float surfT,float Pscale,int refnode,groupinfo voidgroup);
void FrontFlow(nodeptr iptr,sourceptr Source,float *Qnet,nodeptr node,faceptr face,
         float surfT,groupinfo voidgroup,float Qs,float Pscale,int refnode);

void NewSat(sourceptr Source,double Dt,nodeptr node,faceptr face,frontptr AtFront,
       float surfT,float Pscale,int refnode,groupinfo voidgroup);


sourceptr newsource(int i, float Q);
void addsource(sourceptr list, int i, float Q);
void ReadSourceData(sourceptr Source,char *filename);
void ChangeSource(sourceptr list, int i, float Q);
float GetSource(int i,sourceptr list);
int CheckSource(int i, sourceptr list);


void VoidNode(int nXsect,int inNode,float dz,frontptr AtFront,frontptr AtBehind,
              frontptr Behind,frontptr FrontEl,frontptr Front,groupinfo voidElem,
              nodeptr node,faceptr face,frontptr Next2Front,double t,char *fvoidfile,
              float *vol,groupinfo voidgroup,int *numvoid);

void UpdateQ(frontptr AtBehind,frontptr Next2Front,sourceptr Source,nodeptr node,
        int refnode,faceptr face,float surfT,float Pref,float *Qout);
void FrontQ(nodeptr iptr,int refnode,nodeptr node,faceptr face,float surfT,
      float Pref,sourceptr Source);

void addPc(frontptr Next2Front,int refnode,nodeptr node,faceptr face,float surfT,
      float Pref,sourceptr Source);

void printSourceList(sourceptr list);

void fPrintFront(frontptr Front,float dz,nodeptr node,char *filename);

void fFrontGrowth(double t,float satVol,frontptr Front,frontptr Full,float dz,int nz,
           nodeptr node,char *filename,float *minZ,float *avgZ,float *maxZ,
           int numfibers,float visc,sourceptr XsectVf,float avgHz,float Dnom,int inNode);

void fPrintXsectData(float time,int step,int nz,float dz,nodeptr node,char *filename,
         char option);
void fPrintFaceData(float time,int step,faceptr face,char *filename);
void fprintList(frontptr list);
void FileRead(void *Var,int size,char *filename);
void FileWrite(void *Var,int size,char *filename);
void fPrintName(char *filename,char* option,int step);
char *fReadName(char *input);
```

```
void printvoidList(groupinfo label);
vnodeptr newVnode(int i);
void addVnode(vnodeptr list, int i);
vnodeptr Scan2EndVnode(vnodeptr list);
void printVnodeList(vnodeptr list);
int CheckVnodeList(int i, vnodeptr list);
vnodeptr GetVnode(int i, vnodeptr list);
void rmvnode(int i, vnodeptr list);
groupinfo newgroup(int i);
void addgroup(groupinfo list, int i);
groupinfo Scan2EndGroup(groupinfo label);
void printGroupLabel(groupinfo list);
void printGroupList(groupinfo group);
int CheckGroupLabel(int i, groupinfo list);
int CheckGroupList(int i,groupinfo group);
groupinfo GetGroup(int i, groupinfo group);
void rmvGroup(int i, groupinfo list);

void LinkGroups(groupinfo group,groupinfo voidgroup);
void Search(int first,groupinfo group,vnodeptr visit,vnodeptr newvisit);
void Explore(int i,groupinfo group,vnodeptr visit,vnodeptr newvisit,
             vnodeptr queue);
int GetNext2Visit(groupinfo group,vnodeptr visit);
int CheckVector(int goal,int *vector,int size);
void GroupRenumber(groupinfo group);

faceptr newFace(int F,int u,int v);
void addFace(faceptr face,int F,int u,int v);
faceptr Scan2EndFace(faceptr face);
faceptr getFace(int F,faceptr face);
faceptr   findFace(faceptr face,int u,int v);
void printFace(faceptr face);
void ReadFaceData(faceptr face,char *filename,int *nXsect,
    int *fXsect,int *numSlice);
faceptr addSlices(faceptr base,int nXsect,int fXsect,int numSlice);

nodeptr newNodeData(int n);
void addNodeData(nodeptr node,int n);
nodeptr Scan2EndNodeData(nodeptr node);
nodeptr getNode(int n,nodeptr node);
void printNodeData(nodeptr node);
void printNodeNeighbors(nodeptr node);
nodeptr MakeNodeList(faceptr face);

neighborptr newNeighbor(int f,int n);
void addNeighbor(neighborptr neighbor,int f,int n);
neighborptr Scan2EndNeighbor(neighborptr neighbor);
void printNeighbor(neighborptr neighbor);

void saveModel(int nXsect,int fXsect,int numfibers,int inNode,int refnode,float avgD,
          float stdevD,float avgVf,float stdevVf,float Va,float baseGap,
          float Lscale,float avgGap,float stdevGap,nodeptr node,faceptr face,
          char *paramfile,char *nodefile,char *nbrfile,char *facefile,
          char *porefile,sourceptr XsectVf,int cellnum,float Kz,float Kr,float Apore);
void readModel(int *nXsect,int *fXsect,int *numfibers,int *inNode,int *refnode,float *avgD,
          float *stdevD,float *avgVf,float *stdevVf,float *Va,float *baseGap,
          float *Lscale,float *avgGap,float *stdevGap,nodeptr node,faceptr face,
          char *paramfile,char *nodefile,char *nbrfile,char *facefile,sourceptr XsectVf,
```

```
        int *cellnum,float *Kz,float *Kr,float *Apore);

void mystrcat(char *str1,char *str2,char *newstr);
```

C.2 Main Program

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void main()
{
    int       i,tcount,nXsect,fXsect,nz,maxcount,inNode,refnode,numvoid,numfibers,
                cellnum;
    double      t,dt,Dt;
    float       Dnom,visc,surfT,satVol,Volume,vol,Lscale,dz,beta,x[3],avg,
              avg2,stdev,voidvol,Qin,Pin,Pref,pert,deltaD,vol2,avgD,
              stdevD,avgVf,stdevVf,Pc,stopVol,Pscale,avgRh,avgHz,minZ,avgZ,maxZ,
                avgGap,stdevGap,Va,baseGap,Pwave,Kz,Kr,Apore,Ca;
    FILE        *fp,*data,*fPdat,*fvoid;
    sourceptr   Source,OutFlow,srcptr,XsectVf;
    frontptr  Front,Next2Front,Full;
    char        ans[40],Pname[40],satname[40],input[40],output[40],nodeout[40],
              faceout[40],timefile[40],model[5][40],stdmodel[5][40],chr,frontfile[40],
                voidfile[40],Pfile[40],solndat[40];
    groupinfo   voidgroup;
    nodeptr     node,tempnode,nptr;
    faceptr     face,fptr;
    neighborptr  tempN;
    fiberptr  fiber;
    packptr     basepack;

    /* Set "seed" for random generator */
    srand(time(0) * getpid());
  /*    srandom(time(0) * getpid());  */        /* If not Sun */

     /* Initialize variables */
    mystrcat(".param","",stdmodel[0]);
    mystrcat(".node","",stdmodel[1]);
    mystrcat(".nbr","",stdmodel[2]);
    mystrcat(".face","",stdmodel[3]);
    mystrcat(".pore","",stdmodel[4]);

    for(i=0;i<3;i++) x[i] = 0.0;
    inNode = refnode = cellnum = 0;
    avgD = stdevD = avgVf = stdevVf = Va = baseGap = Lscale = Pwave = Apore = Ca = 0;

    XsectVf =  newsource(-1,0.0);  /* Initialize Xsect stats on Vf */

    pert = 0;           /* Perturbation of fiber position in x-y plane */
```

```
    deltaD = 0;         /* Perturbation of fiber diameter at z */

    numfibers = 0;

    minZ = avgZ = maxZ = avgGap = stdevGap = Kz = Kr = 0;
    /* Note:  avgGap and stdevGap are normalized by Dnom */

    nptr = node = tempnode = NULL;
    fptr = NULL;
    Front = Next2Front = Full = NULL;
    tempN = NULL;


    Front = newFrontNode(-1);      /* Initialize list of global front nodes */
    /* Initialize list of full nodes next to global front */
    Next2Front = newFrontNode(-1);


    voidgroup = newgroup(0);       /* Initialize list of voids */

/* Read data on model parameters from input text file */
    avgD = dz = visc = surfT = beta = Pref = 0;
    nz = maxcount = nXsect = fXsect = inNode = refnode = 0;
    printf("Enter name of fiber data input file (Default = 0, model.dat): ");
    scanf("%s",input);
/*  mystrcat("0","",input);*/
    printf("%s\n",input);

    if(*input=='0') mystrcat("model.dat","",input);

    printf("%s\n",input);

    /* Open text file (with line breaks) for output */
    if ((fp = fopen(input, "r")) ==NULL) {
     printf("cannot open file\n");
     exit(1);
    }

    /* Read average (nominal) fiber diameter (avgD), average fiber waviness
        (beta), axial length of CV's (dz),
        number of "slices" (nz), fluid properties:  viscosity (visc) and
        surface tension (surfT) and total volumetric nominal capillary number(Ca). */
    fscanf(fp,"%f %f %f %f %f %d %f %f %f %d %f",&Dnom,&beta,&pert,&deltaD,&dz,
            &nz,&visc,&surfT,&Ca,&maxcount,&Pref);

    /* Close input file */
    fclose(fp);

    printf("dz = %.3f  nz = %d  visc = %.2e  surfT = %.2e Ca = %.2e\n",dz,nz,
        visc,surfT,Ca);
    printf("beta = %.2f\tmaxcount = %d\tPref = %.2e\n",beta,maxcount,Pref);
    printf("pert = %.2f\tdeltaD = %.2f\n",pert,deltaD);


    /* Enter name of void growth data file. */
    fp = NULL;
    printf("Enter name of void growth data file (Default = 0, void.dat): ");
    /*scanf("%s",voidfile);*/
```

```
mystrcat("0","",voidfile);
printf("%s\n",voidfile);
if(*voidfile=='0') mystrcat("void.dat","",voidfile);

/* Open text file (with line breaks) for input */
if ((fp = fopen(voidfile, "w")) ==NULL) {
  printf("cannot open file\n");
  exit(1);
}

fclose(fp);

/* Enter name of average pressure growth data file. */
fp = NULL;
printf("Enter name of average pressure growth data file (Default = 0, P.dat): ");
/*scanf("%s",Pfile);*/
mystrcat("0","",Pfile);
printf("%s\n",Pfile);
if(*Pfile=='0') mystrcat("P.dat","",Pfile);

/* Open text file (with line breaks) for input */
if ((fp = fopen(Pfile, "w")) ==NULL) {
  printf("cannot open file\n");
  exit(1);
}

fclose(fp);

/* Enter name of front growth data file. */
fp = NULL;
printf("Enter name of front growth data file (Default = 0, front.dat): ");
/*scanf("%s",frontfile);*/
mystrcat("0","",frontfile);
printf("%s\n",frontfile);
if(*frontfile=='0') mystrcat("front.dat","",frontfile);

/* Open file for input */
 if ((fp = fopen(frontfile, "w")) ==NULL) {
    printf("cannot open file");
    exit(1);
  }

fclose(fp);


 /* Enter name of solution summary data file. */
 fp = NULL;
 printf("Enter name of (pressure) solution summary data file (Default = 0, soln.dat): ");
 /*scanf("%s",solndat);*/
 mystrcat("0","",solndat);
 printf("%s\n",solndat);
 if(*solndat=='0') mystrcat("soln.dat","",solndat);

 /* Open file for input */
 if ((fp = fopen(solndat, "w")) ==NULL) {
        printf("cannot open file");
        exit(1);
  }
```

```
fclose(fp);


basepack = newCell(-1);              /* Skip header */

node =  newNodeData(-1);       /* Skip header */
face = newFace(-1,-1,-1);      /* Skip header */
fiber = newFiber(-1,x,-1.0);          /* Skip header */

mystrcat("n","",input);
printf("Use previous model (y or n)?: ");
scanf("%s",input);

if(*input=='y'){
  /* Enter model data file prefix */
  printf("Enter model data file prefix (Default = 0, model): ");
  scanf("%s",input);
  /*mystrcat("0","",input);
    printf("%s\n",input);*/
  if(*input=='0') mystrcat("model","",input);

  for(i=0;i<4;i++) mystrcat(input,stdmodel[i],model[i]);

  readModel(&nXsect,&fXsect,&numfibers,&inNode,&refnode,&avgD,&stdev,&avgVf,&stdevVf,
        &Va,&baseGap,&Lscale,&avgGap,&stdevGap,node,face,model[0],
        model[1],model[2],model[3],XsectVf,&cellnum,&Kz,&Kr,&Apore);

}

else{
  printf("stdevGap = %e\n",stdevGap);
  Geometry(Dnom,beta,dz,nz,visc,fiber,basepack,node,face,&nXsect,&fXsect,
        &inNode,&refnode,Pref,pert,deltaD,&avgD,&stdev,&avgVf,&stdevVf,
        XsectVf,&avgGap,&stdevGap,&Va,&baseGap,&Lscale,&numfibers,&cellnum,&Kz,&Kr,
        &Apore);
}

printf("avgGap = %e  stdevGap = %e  Lscale = %e\n",avgGap,stdevGap,Lscale);


mystrcat("n","",input);
printf("Save model (y or n)?: ");
scanf("%s",input);

if(*input=='y'){
  /* Enter model data file prefix */
  printf("Enter model data file prefix (Default = 0, model): ");
  scanf("%s",input);
  /*mystrcat("0","",input);
    printf("%s\n",input);*/
  if(*input=='0') mystrcat("model","",input);

  for(i=0;i<5;i++) mystrcat(input,stdmodel[i],model[i]);

  saveModel(nXsect,fXsect,numfibers,inNode,refnode,avgD,stdev,avgVf,stdevVf,Va,
        baseGap,Lscale,avgGap,stdevGap,node,face,model[0],model[1],
        model[2],model[3],model[4],XsectVf,cellnum,Kz,Kr,Apore);
```

```
    }


    /* Incorporate viscosity into interface conductance, fptr->H. */
    fptr = face->next;          /* Skip header */
    for(; fptr!=NULL; fptr = fptr->next){
      fptr->H /= visc;
    }


    mystrcat("n","",input);
    printf("Continue with simulation (y or n)?: ");
    scanf("%s",input);
    if(*input=='n') exit(1);

    Qin = surfT*Apore*Ca/visc;
    Source = newsource(-1,0.0);
    addsource(Source,inNode,1);  /* Scale Source by Qin */

    srcptr = NULL;
    OutFlow = newsource(-1,0.0);
/*  for(i=(nz-1)*nXsect;i<nz*nXsect;i++) addsource(OutFlow,i,0.0);*/
/*  At present, assume all outflows are at refnode */

/*************************************************************************/
/* Read Source data from input file */
/*  Source = newsource(-1,0.0);
    printf("Enter name of Source input file (Default = 0, Source.dat): ");
    scanf("%s",input);
    if(*input=='0') input = "Source.dat";

    ReadSourceData(Source,input);*/
/*  printSourceList(Source);*/

/* Read Outflow data from input file */
/*  OutFlow = newsource(-1,0.0);
    printf("Enter name of OutFlow input file (Default = 0, OutFlow.dat): ");
    scanf("%s",input);
    if(*input=='0') input = "OutFlow.dat";

    ReadSourceData(OutFlow,input);*/
/*  printSourceList(OutFlow);*/
/*************************************************************************/
/*Initialize Q-vector.  Calculate tow/pore volume, avgRh and Pscale. */
    Volume = avgRh = avgHz = Pscale = 0;
    nptr = node->next;          /* Skip header */
    for(;nptr != NULL; nptr = nptr->next){
      nptr->P0 = nptr->P = Pref;
      if(CheckSource(nptr->node,Source)) nptr->Q = GetSource(nptr->node,Source);
      if(nptr->node==inNode){
        nptr->sat0 = nptr->sat = 1;
/*
        nptr->flag = 1;
*/
      }
      else{

/*      if(!nptr->x[2]){
```

```
          nptr->sat0 = nptr->sat = 1;
          nptr->flag = 1;
        }
*/
       if((nptr->node!=refnode)&&(nptr->node!=inNode)){
         Volume += nptr->V;
         avgRh += nptr->Rh*nptr->V;  /* Define avgRh as a volume-weighted average */
        }
      }
     }

     avgRh /= Volume;
     avgHz = 3.1459*pow(avgRh,4)/(8*visc*nz*dz*Dnom);  /* Tow Length Scale */
/*
      avgHz = 3.1459*pow(avgRh,4)/(8*visc*Dnom); */ /* Fiber Scale */

     if(Lscale) Pwave = Qin/(3.1459*pow(avgRh,4)/(8*visc*Lscale*Dnom)); /* Waviness scale */

     if(avgHz){
      Pscale = Qin/avgHz;
      /*  Scale interface conductance, fptr->H, by avgHz. */
      fptr = face->next;          /* Skip header */
      for(; fptr!=NULL; fptr = fptr->next) fptr->H /= avgHz;
     }
     else{
      printf("Error: division by zero.  avgHz = 0.\n");
      exit(1);
     }

     /*Initialize node pressures to Pref/Pscale.*/
     Pref /= Pscale;
     Pwave /= Pscale;
      nptr = node->next;              /* Skip header */
      for(;nptr != NULL; nptr = nptr->next) nptr->P0 = nptr->P = nptr->Ptemp = Pref;

     Pc = 0;
     nptr = getNode(refnode,node);
     Pc = getAvgPc(nptr,node,face,surfT,Pscale,refnode);
     printf("avgPc = %e at node: %d\tavgHz = %e\tPscale = %e\n",Pc,nptr->node,avgHz,Pscale);
     printf("avgRh = %e\n",avgRh);


     /*  Write avgRh,avgHz,AvgPc and Pscale,avgGap,stdevGap, to solndat file as 1st line. */
     /*  Write Va,baseGap,Lscale to solndat as 2nd line */
     fp = NULL;
      /* Open file for input */
      if ((fp = fopen(solndat, "w")) ==NULL) {
            printf("cannot open file");
             exit(1);
       }

     fprintf(fp,"%e\t%e\t%e\t%e\n",avgRh,avgHz,Pc,Pscale);
     fprintf(fp,"%e\t%e\t%e\n",Va,baseGap,Lscale);

      fclose(fp);
```

```
    t = 0.0;        /* Initial value of simulation time */
    dt = Dt = 0;        /* Current iteration time step and last non-zero time step */
    tcount = 0;


    /*Read data from files and initialize*/
    chr = 'n';
    printf("Data from files (y or n)? ");
/*  scanf("%c",&chr); */
    printf("%c\n",chr);

    if(chr=='y'){
     printf("yesiree bob!\n");

     /* Enter names of input files */
     printf("Enter name of Pressure file (Default = 0, Pdat): ");
     /*scanf("%s",Pname);*/
     mystrcat("0","",Pname);
     printf("%s\n",Pname);
     if(*Pname=='0') mystrcat("Pdat","",Pname);
    /*printf("%s\n",Pname);*/

     printf("Enter name of saturation file (Default = 0, satdat): ");
    /*scanf("%s",satname);*/
     mystrcat("0","",satname);
     printf("%s\n",satname);
     if(*satname=='0') mystrcat("satdat","",satname);
    /*printf("%s\n",satname);*/

    /*FileRead(P,sizeof(P),Pname);
     FileRead(sat,sizeof(sat),satname);*/

     tempnode = node->next;             /* Skip header */
     for(;tempnode != NULL; tempnode->next){
      tempnode->P0 = tempnode->P;
      tempnode->sat0 = tempnode->sat;
     }
    }

    /* Enter names of Output text files */
    printf("Enter name of node data output file (Default = 0, nData): ");
/*  scanf("%s",nodeout);*/
    mystrcat("0","",nodeout);
    printf("%s\n",nodeout);
    if(*nodeout=='0') mystrcat("nData","",nodeout);

    printf("Enter name of face data output file (Default = 0, fData):");
/*  scanf("%s",faceout);*/
    mystrcat("0","",faceout);
    printf("%s\n",faceout);
    if(*faceout=='0') mystrcat("fData","",faceout);


    do{
     /* Initialize list of global front nodes */
     Front = newFrontNode(-1);
     /* Initialize list of full nodes next to global front */
     Next2Front = newFrontNode(-1);
```

```
        /* Initialize list of full nodes */
        Full = newFrontNode(-1);
        vol = 0;     /* Reset value of void volume for next iteration. */
        numvoid = 0;

        voidgroup = newgroup(0);     /* Initialize list of voids */
        printf("t = %.2e\t",t);
        Step(nXsect,dz,Source,Front,Next2Front,inNode,refnode,&t,&dt,&Dt,
            node,face,surfT,voidfile,&vol,Pref,voidgroup,&numvoid,solndat,Pscale,Pwave);
        if(dt!=1e6) t += dt;
        tcount++;
        printf("tcount = %d\n",tcount);

        if(tcount<maxcount+1){


       /*   input = "tempname";
          printList(Front);
          fPrintName("Fron","t",tcount);
          output = fReadName(input);
          printf("output = %s\n",output);
          fPrintFront(Front,dz,node,output);*/

/*
          mystrcat("tempname","",input);
          mystrcat(nodeout,"",output);
          fPrintName(output,"s",tcount);
          output = fReadName(input);
          fPrintXsectData(t,tcount,nz,dz,node,output,'s'); */


/*        fPrintFaceData(t,tcount,face,faceout);*/

          nptr = getNode(inNode,node);
          Pin = nptr->P;

          avg = avg2 = stdev = satVol = stopVol = voidvol = vol2 = 0;
          nptr = node->next;                    /* Skip header */
          for(; nptr->node != inNode; nptr = nptr->next){
            if(nptr->sat==1) addFrontNode(Full,nptr->node);
             satVol += nptr->sat*nptr->V;

            if(CheckGroupList(nptr->node,voidgroup)){
             voidvol += (1-nptr->sat)*nptr->V;
             vol2 += pow((1-nptr->sat)*nptr->V,2);
             }

            if(nptr->sat<0){
             printf("Error:  sat[%d] = %f\n",nptr->node,nptr->sat);
             exit(1);
             }


/*           printf("P[%d] = %.2e  f = %d  sat[%d] = %e\n",nptr->node,nptr->P, nptr->flag,
                nptr->node,nptr->sat);
*/
/*           if(nptr->sat) printf("P[%d] = %.2e  f = %d  sat[%d] = %.2f%%\n",nptr->node,nptr->P,
                        nptr->flag,nptr->node,100*nptr->sat);*/
```

```
        }

        printf("numvoid = %d\n",numvoid);
        printf("Pin = %.2e\tPref = %.2e\tPin-Pref = %.2e\n",Pin,Pref,Pin-Pref);


        fp = NULL;
        /* Open file for input */
        if ((fp = fopen(Pfile, "a")) ==NULL) {
          printf("cannot open file");
          exit(1);
        }
        fprintf(fp,"%e\t%e\n",t,Pin);
        fclose(fp);


      }

      printf("Tow Volume = %e Saturation = %.2f%% Void Fraction = %.2f%%\n",
          Volume,100*satVol/Volume,100*voidvol/Volume);
/*    fprintf(fvoid,"%e\t%e\t%e\t%e\n",t,Volume,100*satVol/Volume,100*voidvol/Volume);*/


      fFrontGrowth(t,satVol,Front,Full,dz,nz,node,frontfile,&minZ,&avgZ,&maxZ,
            numfibers,visc,XsectVf,avgHz,Dnom,inNode);

      printf("%%Dry volume = %.2f%%\n",100*(1-(satVol+voidvol)/Volume));

      stopVol = Volume - satVol + voidvol;

    } while ((tcount<maxcount)&&((stopVol>=0)&&(minZ<50)));  /* Stopping criteria */

    /*fclose(fvoid);*/


    printf("Save data (y or n)? ");
/*  scanf("%s",ans);*/
    mystrcat("y","",ans);
/*  printf("%s\n",ans);     */
/*  mystrcat("n","",ans);*/
    if(*ans=='y') {
      printf("You betcha, bucko!\n");

      /* Enter names of output files */
      printf("Enter name of Pressure file (Default = 0, Pdat): ");
      /*      scanf("%s",Pname);*/
      mystrcat("0","",Pname);
      printf("%s\n",Pname);
      if(*Pname=='0') mystrcat("Pdat","",Pname);
      /*      printf("%s\n",Pname);*/

      printf("Enter name of saturation file (Default = 0, satdat): ");
      /*      scanf("%s",satname);*/
        mystrcat("0","",satname);
      printf("%s\n",satname);
      if(*satname=='0') mystrcat("satdat","",satname);
      /*      printf("%s\n",satname);*/
```

```
        /*      FileWrite(P,sizeof(P),Pname);
           FileWrite(sat,sizeof(sat),satname);  */

    }

    printf("t = %e  tcount = %d\n",t,tcount);

}
```

C.3 Definition of Model Geometry and Conductance Matrix

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void Geometry(float Dnom,float beta,float dz,int nz,float visc,fiberptr fiber,
        packptr basepack,nodeptr node,faceptr face,int *nXsect,int *fXsect,
        int *inNode,int *refnode,float Pref,float pert,float deltaD,float *avgD,
        float *stdevD,float *avgVf,float *stdevVf,sourceptr XsectVf,float *avgGap,
        float *stdevGap,float *Va,float *baseGap,float *Lscale,int *numfibers,int *cellnum,
        float *Kz,float *Kr,float *Apore)
{

    int       i,j,k,F,vertex[3],a,b,gapcount,u,v,numgap;
    float         x[3],y[3],d[3],D3,P,Rh[4],avgRh,Vf[4],sum,m[3],Xmid[3],
              Ymid[3],xc[7],yc[7],A[7],Az,sumA,X[3],Xc,Yc,Zc,towVf,Vf2,sumVf,sumVz,sumVr,
              L[3],angle[3],Qx,Qy,delta[3],hx,gap2,minGap,maxGap,avgd,Cpore,Cgap,C;
    fiberptr  tempfiber;
    packptr       tempb;
    gapptr        gap,tempg;
    nodeptr       nptr,uptr,vptr;
    faceptr       fptr;
    neighborptr   nbru,nbrv;
    FILE          *fp;
    char          *name,*input;


    printf("In BaseGeometry\n");

/* Initialize local variables */
    tempfiber = NULL;
    tempb = NULL;
    name = NULL;
    input = NULL;
    nptr = uptr = vptr = NULL;
    fptr = NULL;
```

```
    nbru = nbrv = NULL;

    /* Initialize shape factors */
    Cpore = 0.5;
    Cgap = 1.0/3.0;
    C = 0;

    hx = 0;
    numgap = 0;
    minGap = 1e6;
    maxGap = 0;

    D3 = pow(Dnom,3);
    k = *numfibers = *cellnum = gapcount = *nXsect = *fXsect = F = u = v = 0;
    Xc = Yc = Zc = *avgVf = towVf = Vf2 = sumVf = *stdevVf = *avgGap = *stdevGap = gap2 = 0;
    *Va = *Kz = *Kr = sumVz = sumVr = 0;
    for(i=0;i<3;i++){
        vertex[i] = -1;
        L[i] = angle[i] = x[i] = y[i] = d[i] = delta[i] = -1.0;
    }

    for(i=0;i<7;i++){
        A[i] = xc[i] = yc[i] = -1.0;
    }


/* Read vertex data from input text  file */
    /* Enter name of fiber data input file */
    input = NULL;
    printf("Enter name of vertex data input file (Default = 0, vertex.dat): ");
/*  scanf("%s",input);*/
    input = "0";
    printf("%s\n",input);
    if(*input=='0') input = "vertex.dat";

    ReadVertexData(basepack,cellnum,input);

/* Read base fiber data from input text file */
    /* Enter name of fiber data input file */
    input = NULL;
    printf("Enter name of fiber data input file (Default = 0, fiber.dat): ");
/*  scanf("%s",input);*/
    input = "0";
    printf("%s\n",input);
    if(*input=='0') input = "fiber.dat";

    ReadFiberData(fiber,input,numfibers);


     /* Compute cell info in the "base" layer */
     tempb = basepack->next;                    /* Skip header */
    for(;tempb!=NULL;tempb=tempb->next){

     /* Rh[i] and Vf[i] are values corresponding to side[i] of cell.  Rh[3]
       and Vf[3] correspond to pore values. */
     *baseGap = 1e6;
     avgd = 0;
     for(i=0;i<4;i++) Rh[i] = Vf[i] = 0;
```

```
for(i=0;i<3;i++){
  tempfiber = getFiber(tempb->vertex[i],X[2],fiber);
  vertex[i] = tempb->vertex[i];
  x[i] = tempfiber->x[0];
  y[i] = tempfiber->x[1];
  d[i] = tempfiber->d;
}

/* Calculate lengths of triangle's (i.e., cell's) sides */
L[0] = sqrt((x[2]-x[1])*(x[2]-x[1]) + (y[2]-y[1])*(y[2]-y[1]));
L[1] = sqrt((x[2]-x[0])*(x[2]-x[0]) + (y[2]-y[0])*(y[2]-y[0]));
L[2] = sqrt((x[0]-x[1])*(x[0]-x[1]) + (y[0]-y[1])*(y[0]-y[1]));

/* Calculate cross-sectional area of triangular cell */
A[3] = 0.5*fabs(x[0]*(y[1]-y[2])-y[0]*(x[1]-x[2])+(x[1]*y[2]-x[2]*y[1]));

/* Calculate fiber volume fraction -> pore (i.e., axial flow) area */
/* Calculate interior angles of triangle using law of cosines */
angle[0] = acos(0.5*(L[1]*L[1]+L[2]*L[2]-L[0]*L[0])/(L[1]*L[2]));
angle[1] = acos(0.5*(L[0]*L[0]+L[2]*L[2]-L[1]*L[1])/(L[0]*L[2]));
angle[2] = acos(0.5*(L[1]*L[1]+L[0]*L[0]-L[2]*L[2])/(L[1]*L[0]));

sum = 0.0;
for(i=0;i<3;i++){
  /* Area of included fiber sector. Negative because will later
     subtract when determining pore centroid location. */

  A[i] = -0.125*angle[i]*d[i]*d[i];
  sum -= A[i];
}

Vf[3] = sum/A[3];              /* Local fiber volume fraction  */

*avgVf += Vf[3];    /* Use pore-based values of Vf for stats */
Vf2 += Vf[3]*Vf[3];
sumVf += Vf[3];


/* Calculate fiber gap, delta[i], on side[i] of the triangular cell */
delta[0] = L[0] - 0.5*(d[1]+d[2]);
delta[1] = L[1] - 0.5*(d[0]+d[2]);
delta[2] = L[2] - 0.5*(d[0]+d[1]);

for(i=0;i<3;i++){
  if(delta[i]>maxGap) maxGap = delta[i];
  if(delta[i]<minGap) minGap = delta[i];
  if(delta[i]<(*baseGap)) *baseGap = delta[i];
}

if(*baseGap==delta[0]) avgd = 0.5*(d[1]+d[2]);
else if(*baseGap==delta[1]) avgd = 0.5*(d[0]+d[2]);
else avgd = 0.5*(d[0]+d[1]);

*Va += Vf[3]*(1+(*baseGap)/avgd)*(1+(*baseGap)/avgd);

/* Check for fiber overlap */
for(i=0;i<3;i++){
```

```
        if(delta[i]<0){
          printf("Error:  fiber overlap.  Program aborted.\n");
          exit(1);
        }
      }

    }

     *avgVf /= *cellnum;
     printf("Base stat's:   avgVf = %.3f  minGap = %e  maxGap = %e\n",*avgVf,minGap,maxGap);

     *stdevVf = (Vf2 - sumVf*sumVf/(*cellnum))/(*cellnum);
     if(*stdevVf>fTol*fTol) *stdevVf = sqrt(*stdevVf);
    else *stdevVf = 0;


    printf("Base stdevVf = %e cellnum = %d\n",*stdevVf,*cellnum);

    *Va /= *cellnum;

    *baseGap = sqrt((*Va)/(*avgVf)) - 1;

    printf("Base avg Va = %e avg gap = %e\n",*Va,*baseGap);

    if(pert) *Lscale = beta*(*baseGap);
    else *Lscale = 0;

    printf("Entering perturbFibers\n");
    perturbFibers(nz,dz,*numfibers,fiber,pert,deltaD,avgD,stdevD,*Lscale,*baseGap);
    printf("perturbFibers completed\n");

    gap = newGap(-1,0,0);                    /* Skip header */

    *stdevVf = *avgVf = sumVf = Vf2 = maxGap = 0;
    minGap = 1e6;

for(k=0;k<nz;k++){
    gapcount += *cellnum;
    X[2] = k*dz;
    avgRh = 0.0;
    *avgVf = 0;

    /* Compute cell info */
    tempb = basepack->next;                          /* Skip header */
    for(;tempb!=NULL;tempb=tempb->next){

        /* Rh[i] and Vf[i] are values corresponding to side[i] of cell.  Rh[3]
             and Vf[3] correspond to pore values. */
        for(i=0;i<4;i++) Rh[i] = Vf[i] = 0;

        for(i=0;i<3;i++){
            tempfiber = getFiber(tempb->vertex[i],X[2],fiber);
            vertex[i] = tempb->vertex[i];
            x[i] = tempfiber->x[0];
            y[i] = tempfiber->x[1];
            d[i] = tempfiber->d;
        }
```

```
        /* Calculate lengths of triangle's (i.e., cell's) sides */
        L[0] = sqrt((x[2]-x[1])*(x[2]-x[1]) + (y[2]-y[1])*(y[2]-y[1]));
        L[1] = sqrt((x[2]-x[0])*(x[2]-x[0]) + (y[2]-y[0])*(y[2]-y[0]));
        L[2] = sqrt((x[0]-x[1])*(x[0]-x[1]) + (y[0]-y[1])*(y[0]-y[1]));

        /* Calculate cross-sectional area of triangular cell */
        A[3] = 0.5*fabs(x[0]*(y[1]-y[2])-y[0]*(x[1]-x[2])+(x[1]*y[2]-x[2]*y[1]));

        /* Calculate fiber volume fraction -> pore (i.e., axial flow) area */
        /* Calculate interior angles of triangle using law of cosines */
        angle[0] = acos(0.5*(L[1]*L[1]+L[2]*L[2]-L[0]*L[0])/(L[1]*L[2]));
        angle[1] = acos(0.5*(L[0]*L[0]+L[2]*L[2]-L[1]*L[1])/(L[0]*L[2]));
        angle[2] = acos(0.5*(L[1]*L[1]+L[0]*L[0]-L[2]*L[2])/(L[1]*L[0]));

        sum = 0.0;
        for(i=0;i<3;i++){
            /* Area of included fiber sector. Negative because will later
            subtract when determining pore centroid location. */

            A[i] = -0.125*angle[i]*d[i]*d[i];
            sum -= A[i];
        }

        Vf[3] = sum/A[3];           /* Local fiber volume fraction  */

        Az = (1-Vf[3])*A[3];  /* Local pore area in axial flow direction */
        *Apore += Az;

        /* Calculate fiber surface (perimeter) in cell */
        P = 0.0;
        for(i=0;i<3;i++) P += 0.5*angle[i]*d[i];

    /*  Rh[3] = Az/P;           */  /* Local hydraulic radius for axial flow */

/* Calculate centroid of pore area using first moments of component primitives */

    /* Calculate centroid of triangular cell formed by vertices 1,2,3 */
        /* Calculate slopes of vertex to side bisector lines.  Centroid is at
            intersection of bisectors (note: need to only use 2 besectors) */

        /* Calculate midpoint coordinates for side[i] opposite vertex[i] */
        Xmid[0] = 0.5*(x[2]+x[1]);
        Ymid[0] = 0.5*(y[2]+y[1]);

        Xmid[1] = 0.5*(x[2]+x[0]);
        Ymid[1] = 0.5*(y[2]+y[0]);

        Xmid[2] = 0.5*(x[0]+x[1]);
        Ymid[2] = 0.5*(y[0]+y[1]);


        /* Calculate slope, m[i], from vertex[i] to (Xmid[i],Ymid[i]) */
        for(i=0;i<3;i++){
            /* Calculate m[i] only if it is finite */
            if(Xmid[i] - x[i]) m[i] = (Ymid[i] - y[i])/(Xmid[i] - x[i]);
        }

        /* Pick 2 of the bisectors based on
```

```
    a)  slope m[i] finite
    b)  difference of slopes m[a]-m[b] is non-zero */

a = b = -1;
i = 0;
do{
    if(Xmid[i] - x[i]) a = i;
    i++;
} while ((a<0) && (i<3));

i = 0;
do{
    if(i!=a){
        if(Xmid[i] - x[i]){
            if(m[a] - m[i]) b = i;
        }
    }
    i++;
} while ((b<0) && (i<3));

/* Centroid of triangle formed by vertex fibers 1,2,3 */
xc[3] = (m[a]*x[a]-m[b]*x[b] + y[b]-y[a])/(m[a]-m[b]);
yc[3] = (m[a]*(y[b]-m[b]*x[b]) - m[b]*(y[a]-m[a]*x[a]))/(m[a]-m[b]);


/* Locate included sector centroid of each fiber in cell */
sectorCentroid(0,1,2,x,y,d,angle,xc,yc);
sectorCentroid(1,0,2,x,y,d,angle,xc,yc);
sectorCentroid(2,0,1,x,y,d,angle,xc,yc);

/* Calculate fiber gap, delta[i], on side[i] of the triangular cell */
delta[0] = L[0] - 0.5*(d[1]+d[2]);
delta[1] = L[1] - 0.5*(d[0]+d[2]);
delta[2] = L[2] - 0.5*(d[0]+d[1]);

for(i=0;i<3;i++){
 if(delta[i]>maxGap) maxGap = delta[i];
 if(delta[i]<minGap) minGap = delta[i];
}

/* Check for fiber overlap */
for(i=0;i<3;i++){
 if(delta[i]<0){
   printf("Error:  fiber overlap.  Program aborted.\n");
   exit(1);
  }
}

/* Locate centroid coordinates of "gap" element between fibers j and k
on side[i] */

gapInfo(0,1,2,x,y,xc,yc,L,d,A,delta,&P,Rh,Vf);
gapInfo(1,0,2,x,y,xc,yc,L,d,A,delta,&P,Rh,Vf);
gapInfo(2,0,1,x,y,xc,yc,L,d,A,delta,&P,Rh,Vf);


/* Calculate and sum first moments of component primitives */
Qx = Qy = sumA = 0.0;
```

```
    for(i=0;i<7;i++){
        Qx += xc[i]*A[i];
        Qy += yc[i]*A[i];
        sumA += A[i];
    }

    Az = sumA;        /* Pore element axial cross-section area */

    Rh[3] = Az/P;         /* Local hydraulic radius for axial flow */

    /* Calculate coordinates of pore centroid */
    X[0] = Qx/sumA;
    X[1] = Qy/sumA;


/* Define control volumes centered around pore and gap centroids */
    /* Pore CV #'s correspond to first cellnum CV'c in each layer.
        Gap CV #'s in each layer added to end of CV list for each layer
        and specified by endpoints of corresponding L[i]. */

    if(k) u = tempb->cell + k*(*nXsect);
    else u = tempb->cell;

    addNodeData(node,u);
    uptr = getNode(u,node);

    for(i=0;i<3;i++) uptr->x[i] = X[i];

    uptr->V = dz*Az;
    uptr->Rh = Rh[3];
    uptr->Vf = Vf[3];
    *avgVf += Vf[3];    /* Use pore-based values of Vf for stats */
    Vf2 += Vf[3]*Vf[3];
    sumVf += Vf[3];

    nbru = uptr->neighbor;


    if(!checkGap(gap,vertex[1],vertex[2],k,*nXsect)){
        addGap(gap,gapcount,vertex[1],vertex[2]);

        addNodeData(node,gapcount);
        vptr = getNode(gapcount,node);
        vptr->x[0] = xc[4];
        vptr->x[1] = yc[4];
        vptr->x[2] = X[2];
        vptr->V = -dz*A[4];    /* "-" because only -0.5*A[gap] recorded */
        vptr->Rh = Rh[0];
        vptr->Vf = Vf[0];

        nbrv = vptr->neighbor;

        v = gapcount;
        addFace(face,F,u,v);
        fptr = getFace(F,face);
        fptr->L = dist(X,vptr->x);

        fptr->H = dz*TransverseH(1,2,delta[0],d);
```

```
            fptr->Rcap = delta[0];
            fptr->A = dz*delta[0];
/*
            vptr->Rh = 0.5*delta[0];

            fptr->H = Cgap*fptr->A*vptr->Rh*vptr->Rh/fptr->L;
*/
            *Kr += vptr->V*fptr->L*fptr->H/fptr->A;
            sumVr += vptr->V;

            F++;
            gapcount++;

            numgap++;
            *avgGap += delta[0];
            gap2 += delta[0]*delta[0];
        }
        else{
            tempg = getGap(gap,vertex[1],vertex[2],k,*nXsect);
            v = tempg->i;

            vptr = getNode(v,node);
            vptr->V -= dz*A[4];        /* see note above */
            nbrv = vptr->neighbor;

            addFace(face,F,u,v);
            fptr = getFace(F,face);
            fptr->L = dist(X,vptr->x);

            fptr->H = dz*TransverseH(1,2,delta[0],d);

            fptr->Rcap = delta[0];
            fptr->A = dz*delta[0];
/*
            fptr->H = Cgap*fptr->A*vptr->Rh*vptr->Rh/fptr->L;
*/
            *Kr += 0.5*vptr->V*fptr->L*fptr->H/fptr->A;
            sumVr += 0.5*vptr->V;

            F++;
        }

        addNeighbor(nbru,F-1,v);
        addNeighbor(nbrv,F-1,u);


        if(!checkGap(gap,vertex[2],vertex[0],k,*nXsect)){
            addGap(gap,gapcount,vertex[2],vertex[0]);

            addNodeData(node,gapcount);
            vptr = getNode(gapcount,node);
            vptr->x[0] = xc[5];
            vptr->x[1] = yc[5];
            vptr->x[2] = X[2];
            vptr->V = -dz*A[5];    /* "-" because only -0.5*A[gap] recorded */
            vptr->Rh = Rh[1];
            vptr->Vf = Vf[1];
```

```
            nbrv = vptr->neighbor;

            v = gapcount;
            addFace(face,F,u,v);
            fptr = getFace(F,face);
            fptr->L = dist(X,vptr->x);

            fptr->H = dz*TransverseH(2,0,delta[1],d);

            fptr->Rcap = delta[1];
            fptr->A = dz*delta[1];
/*
            vptr->Rh = 0.5*delta[1];

            fptr->H = Cgap*fptr->A*vptr->Rh*vptr->Rh/fptr->L;
*/
            *Kr += vptr->V*fptr->L*fptr->H/fptr->A;
            sumVr += vptr->V;

            F++;
            gapcount++;

                numgap++;
               *avgGap += delta[1];
               gap2 += delta[1]*delta[1];
        }
        else{
            tempg = getGap(gap,vertex[2],vertex[0],k,*nXsect);
            v = tempg->i;

            vptr = getNode(v,node);
            vptr->V -= dz*A[5];     /* See note above */
            nbrv = vptr->neighbor;

            addFace(face,F,u,v);
            fptr = getFace(F,face);
            fptr->L = dist(X,vptr->x);

            fptr->H = dz*TransverseH(2,0,delta[1],d);

            fptr->Rcap = delta[1];
            fptr->A = dz*delta[1];
/*
            fptr->H = Cgap*fptr->A*vptr->Rh*vptr->Rh/fptr->L;
*/
            *Kr += 0.5*vptr->V*fptr->L*fptr->H/fptr->A;
            sumVr += 0.5*vptr->V;

            F++;
        }

        addNeighbor(nbru,F-1,v);
        addNeighbor(nbrv,F-1,u);

        if(!checkGap(gap,vertex[0],vertex[1],k,*nXsect)){
            addGap(gap,gapcount,vertex[0],vertex[1]);
```

```
            addNodeData(node,gapcount);
            vptr = getNode(gapcount,node);
            vptr->x[0] = xc[6];
            vptr->x[1] = yc[6];
            vptr->x[2] = X[2];
            vptr->V = -dz*A[6];    /* "-" twice because only -0.5*A[gap] recorded */
            vptr->Rh = Rh[2];
            vptr->Vf = Vf[2];

            nbrv = vptr->neighbor;

            v = gapcount;
            addFace(face,F,u,v);
            fptr = getFace(F,face);
            fptr->L = dist(X,vptr->x);

            fptr->H = dz*TransverseH(0,1,delta[2],d);

            fptr->Rcap = delta[2];
            fptr->A = dz*delta[2];
/*
            vptr->Rh = 0.5*delta[2];

            fptr->H = Cgap*fptr->A*vptr->Rh*vptr->Rh/fptr->L;
*/
            *Kr += vptr->V*fptr->L*fptr->H/fptr->A;
            sumVr += vptr->V;

            F++;
            gapcount++;

                numgap++;
               *avgGap += delta[2];
               gap2 += delta[2]*delta[2];
        }
        else{
            tempg = getGap(gap,vertex[0],vertex[1],k,*nXsect);
            v = tempg->i;

            vptr = getNode(v,node);
            vptr->V -= dz*A[6];
            nbrv = vptr->neighbor;

            addFace(face,F,u,v);
            fptr = getFace(F,face);
            fptr->L = dist(X,vptr->x);

            fptr->H = dz*TransverseH(0,1,delta[2],d);

            fptr->Rcap = delta[2];
            fptr->A = dz*delta[2];

            fptr->H = Cgap*fptr->A*vptr->Rh*vptr->Rh/fptr->L;

            *Kr += 0.5*vptr->V*fptr->L*fptr->H/fptr->A;
            sumVr += 0.5*vptr->V;

            F++;
```

```
        }

        addNeighbor(nbru,F-1,v);
        addNeighbor(nbrv,F-1,u);


    } /* end "cell" loop */

    if(!k){
        *nXsect = gapcount;
        *fXsect = F + (*nXsect);
        *inNode = nz*(*nXsect);
        *refnode = nz*(*nXsect) + 1;

        /* Add "inNode" */
        u = *inNode;


        addNodeData(node,u);
        nptr = getNode(u,node);

        nptr->V = 1;
        nptr->Rh = 1;
        nptr->x[2] = -dz;
        nptr->sat0 = nptr->sat = 1;
        nptr->flag = 1;

        for(v=0;v<*nXsect;v++){
            addFace(face,F,u,v);

            vptr = getNode(v,node);

            fptr = getFace(F,face);
            fptr->L = dz;

            avgRh = vptr->Rh;
            fptr->Rcap = avgRh;
            fptr->A = vptr->V/dz;

            if(v<*cellnum) C = Cpore;
            else C = Cgap;

            fptr->H = C*fptr->A*avgRh*avgRh/fptr->L;

            *Kz += vptr->V*C*avgRh*avgRh;
            sumVz += vptr->V;

            nbru = nptr->neighbor;
            nbrv = vptr->neighbor;

            F++;

            addNeighbor(nbru,F-1,v);
            addNeighbor(nbrv,F-1,u);

        }

        /* Add "reference" node */
```

```
    addNodeData(node,*refnode);
    nptr = getNode(*refnode,node);

    nptr->V = 1;
    nptr->Rh = 1;
    nptr->x[2] = nz*dz;

}


/* Add axial faces to control volume elements behind current layer */
if(k){
    if(k<nz-1){
        for(i=0;i<*nXsect;i++){
            v = i + k*(*nXsect);
            u = v - (*nXsect);
            addFace(face,F,u,v);

            uptr = getNode(u,node);
            vptr = getNode(v,node);

            fptr = getFace(F,face);
            fptr->L = dist(uptr->x,vptr->x);


            /* Use volume-weighted averages */
            avgRh = (uptr->V*uptr->Rh + vptr->V*vptr->Rh)/(uptr->V + vptr->V);
            fptr->Rcap = avgRh;
            fptr->A = (uptr->V*uptr->V+vptr->V*vptr->V)/(dz*(uptr->V+vptr->V));

            if(i<*cellnum) C = Cpore;
            else C = Cgap;

            fptr->H = C*fptr->A*avgRh*avgRh/fptr->L;

            *Kz += vptr->V*C*avgRh*avgRh;
            sumVz += vptr->V;

            nbru = uptr->neighbor;
            nbrv = vptr->neighbor;

            F++;

            addNeighbor(nbru,F-1,v);
            addNeighbor(nbrv,F-1,u);

        }
    }
    if(k==nz-1){
        for(i=0;i<*nXsect;i++){
            hx = 0;
            v = i + k*(*nXsect);
            u = v - (*nXsect);
            addFace(face,F,u,v);

            uptr = getNode(u,node);
            vptr = getNode(v,node);
```

```
        fptr = getFace(F,face);
        fptr->L = dist(uptr->x,vptr->x);


        /* Use volume-weighted averages */
              avgRh = (uptr->V*uptr->Rh + vptr->V*vptr->Rh)/(uptr->V + vptr->V);

              fptr->Rcap = avgRh;
             fptr->A = (uptr->V*uptr->V+vptr->V*vptr->V)/(dz*(uptr->V+vptr->V));


              if(i<*cellnum) C = Cpore;
              else C = Cgap;

              fptr->H = C*fptr->A*avgRh*avgRh/fptr->L;

        *Kz += vptr->V*C*avgRh*avgRh;
        sumVz += vptr->V;

        hx = fptr->H;

        nbru = uptr->neighbor;
        nbrv = vptr->neighbor;

        F++;

        addNeighbor(nbru,F-1,v);
        addNeighbor(nbrv,F-1,u);

        /* Add faces to reference node for last layer */
        u = v;
        v = *refnode;
        addFace(face,F,u,v);

        uptr = getNode(u,node);
        vptr = getNode(v,node);

        fptr = getFace(F,face);
        fptr->L = dz;

        avgRh = uptr->Rh;
        fptr->Rcap = avgRh;
        fptr->A = uptr->V/dz;
        fptr->H = hx;

        *Kz += uptr->V*C*avgRh*avgRh;
        sumVz += uptr->V;

        nbru = uptr->neighbor;
        nbrv = vptr->neighbor;

        F++;

        addNeighbor(nbru,F-1,v);
        addNeighbor(nbrv,F-1,u);


}
```

```
            }
        }

        *avgVf /= *cellnum;
        printf("X-sect avgVf = %.3f\n",*avgVf);
        addsource(XsectVf,k,*avgVf);
        towVf += *avgVf;

    }  /* end k-loop */

        *stdevVf = (Vf2 - sumVf*sumVf/(k*(*cellnum)))/(k*(*cellnum));
        if(*stdevVf>fTol*fTol) *stdevVf = sqrt(*stdevVf);
        else *stdevVf = 0;

        *avgVf = towVf/k;
        printf("Tow average Vf = %.3f\tTow std dev in Vf = %.3e (%.1f%% of avgVf)\n",
            *avgVf,*stdevVf,100*(*stdevVf)/(*avgVf));


        *stdevGap = (gap2 - (*avgGap)*(*avgGap)/numgap)/numgap;
        if(*stdevGap>fTol*fTol) *stdevGap = sqrt(*stdevGap);
        else *stdevGap = 0;

        *avgGap /= numgap;

        *Kz /= sumVz;
        *Kr /= sumVr;

        *Apore /= (*cellnum*nz);

        printf("minGap = %e  maxGap = %e Kz = %e Kr = %e\n",minGap,maxGap,*Kz,*Kr);

        Scaling(node,face,Dnom,Apore);

        printf("Apore = %e\n",*Apore);

    /*  fptr = face->next;

        for(;fptr!=NULL;fptr=fptr->next){
            printf("%d %.2e\t",fptr->F,fptr->H);
        }

        printf("\n\n");

        for(;fptr!=NULL;fptr=fptr->next) printf("%d (%d,%d)\t",fptr->F,fptr->u,
            fptr->v);

        printf("\n\n");*/


    /*  nptr = node->next;
        for(;nptr!=NULL;nptr=nptr->next){
                if(nptr->V>0) printf("%d ",nptr->node);
        }
        printf("\n\n");

    */
    }
```

```
float  myrandom()
{
     float      number,largeNum;

     number = 0;
     largeNum = 32767;
     number = (float) rand()/largeNum;
/*     number = (float) random()/ largeNum;*/  /* If not Sun */

     return number;

}

int evenOdd()
{
     int        test;

     test = 0;
     if(myrandom()>0.5) test = 1;

     return test;
}

float dist(float *P1,float *P2)
/* Calculate distance between points P1 and P2 */
{
     int        i;
     float      L;

     L = 0.0;
     for(i=0;i<3;i++) L += (P2[i]-P1[i])*(P2[i]-P1[i]);
     L = sqrt(L);

     return L;
}

void sectorCentroid(int v0,int v1,int v2,float *x,float *y,float *d,float *angle,
                         float *xc, float *yc)

/* Add directed line segments from vertex[v0] to vertices j&k and normalize to
     calculate unit vector in direction of angle[v0] bisector. The sector
     centroid lies on the bisector at a distance of
     2d[v0]*sin(0.5*angle[v0])/(3*angle[v0]) from vertex[v0].  */

{
     float      R,Rx,Ry,dcosX,dcosY;     /* dcosI: direction cosine in I-direction */

/* Initialize local variables */
     R = Rx = Ry = dcosX = dcosY = 0.0;


/* Add vectors from v0 to v1 and v2 */
     Rx = x[v1] + x[v2] - 2*x[v0];
     Ry = y[v1] + y[v2] - 2*y[v0];

/* Calculate magnitude of resultant */
     R = sqrt(Rx*Rx + Ry*Ry);
```

```
/* Calculate direction cosines of resultant */
    dcosX = Rx/R;
    dcosY = Ry/R;

/* Calculate sector v0's centroid coordinates (xc[v0],yc[v0]) */
    xc[v0] =  x[v0] + dcosX*2*d[v0]*sin(0.5*angle[v0])/(3*angle[v0]);
    yc[v0] = y[v0] + dcosY*2*d[v0]*sin(0.5*angle[v0])/(3*angle[v0]);
}


void gapInfo(int v0,int v1,int v2,float *x,float *y,float *xc,float *yc,
        float *L,float *d,float *A,float *delta,float *P,float *Rh,float *Vf)
/* Generate info on gap element's

Centroid Location in x-y plane:
Locate centroid coordinates of "gap" element between fibers v1 and v2
on side[0],opposite v0. Determine direction of directed line segment from v1
to v2. By symmetry, it can be shown that the gap element centroid lies on this
line segement a distance 0.5*(d[v1]+gap[v0]) from v1 (i.e., in the center of the
gap along the line segment from v1 to v2).
*/

{
    float     R,Rx,Ry,dcosX,dcosY,    /* dcosI: direction cosine in I-direction */
          width,h1,h2,dmin,dist,theta1,theta2,Asect,Abox,Pgap;

/* Initialize local variables */
    width = h1 = h2 = dmin = dist = theta1 = theta2 = Asect = Abox = 0.0;
    R = Rx = Ry = dcosX = dcosY = Pgap = 0.0;

/* Calculate distance of gap centroid from v1 */
    dist = 0.5*(d[v1]+delta[v0]);

/* Calculate components of vector from v1 to v2 */
    Rx = x[v2] - x[v1];
    Ry = y[v2] - y[v1];

/* Calculate magnitude of resultant */
    R = sqrt(Rx*Rx + Ry*Ry);

/* Calculate direction cosines of resultant */
    dcosX = Rx/R;
    dcosY = Ry/R;

/* Locate centroid of gap element in the x-y plane */
    xc[v0+4] = x[v1] + dcosX*dist;
    yc[v0+4] = y[v1] + dcosY*dist;


/* Calculate gap element area */

    /* Determine width of gap element */
    if(d[v1]<d[v2]) dmin = d[v1];
    else dmin = d[v2];

    /* Special case:  "very" low local Vf */
    if(delta[v0]/dmin>0.3) width = 0.1*delta[v0]/dmin;
```

```
    else width = 0.99*delta[v0]/dmin;

    /* Determine angles included by gap width limits */
    /* Used to be 0.5*width/d[] */
    theta1 = 2*asin(width/d[v1]);
    theta2 = 2*asin(width/d[v2]);

    /* Calculate total of sector areas on v1 and v2 */
    Asect = 0.125*(pow(d[v1],2)*theta1 + pow(d[v2],2)*theta2);

    /* Calculate sum of other areas overlapping fibers v1 and v2, Ares */
    h1 = 0.5*d[v1]*cos(0.5*theta1);
    h2 = 0.5*d[v2]*cos(0.5*theta2);

    Abox = 0.5*width*(h1+h2);

    /* Calculate gap element area (<0 because it will be subtracted later) */
    /* Multiply by 0.5 since only 1/2 of gap area overlaps with triangular
    primitive */
    A[v0+4] = 0.5*(Abox + Asect - width*L[v0]);

    /* Calculate length of fiber surfaces bounding gap */
    Pgap = 0.5*(theta1*d[v1] + theta2*d[v2]);

    /* Calculate hydraulic radius of gap in axial direction */
    Rh[v0] = -2*A[v0+4]/Pgap;          /* Factor of "-2", note on A[v0+4] */

    /* Calculate gap element fiber volume fraction */

    if(-2*A[v0+4]>=0) Vf[v0] = Asect/(Asect-2*A[v0+4]);
                                    /* Factor of "-2", note on A[v0+4] */
    else{
     printf("Error in fiber perturbation:  fiber overlap.  Exit program.\n");
     exit(1);
    }

    *P -= 0.5*Pgap;

}

void Scaling(nodeptr node,faceptr face,float Dnom,float *Apore)
{
     float          D2,D3;
     nodeptr            nptr;
    faceptr         fptr;

    nptr = node->next;              /* Skip header */
    fptr = face->next;              /* Skip header */


    D2 = pow(Dnom,2);
    D3 = pow(Dnom,3);

    for(;nptr!=NULL;nptr=nptr->next){
        nptr->V *= D3;
        nptr->Rh *= Dnom;
    }
```

```
    for(;fptr!=NULL;fptr=fptr->next){
        fptr->L *= Dnom;
        fptr->A *= D2;
        fptr->Rcap *= Dnom;
        fptr->H *= D3;
    }

    *Apore *= D2;
}


#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

float TransverseH(int f1,int f2,float minGap,float *d)
{

/* Calculate transverse H using lubrication approximation.  Evaluate integral using
   Simpson's rule evaluated from x=0 to x=minR. */

  int     i;
  float   Integral,xLimit,dx,H,gap[100],minR,maxR,L,x;

  for(i=0;i<101;i++) gap[i] = 0;

  H = dx = minR = maxR = L = x = 0;
  if(minGap){

   if(d[f1]<d[f2]){
     minR =0.5*d[f1];
     maxR = 0.5*d[f2];
   }
   else{
     minR = 0.5*d[f2];
     maxR = 0.5*d[f1];
   }

   L = minGap + minR + maxR;

   /* Evaluate gap at "small" uniform intervals.  Here use 100 intervals. Then raise gap to -3 */

   dx = 0.01*minR;
   for(i=0;i<101;i++){
     x = (i)*dx;
     gap[i] = L - sqrt(minR*minR-x*x) - sqrt(maxR*maxR-x*x);
     gap[i] = 1/(gap[i]*gap[i]*gap[i]);
   }

   Integral = gap[0] + gap[100];
```

```
   for(i=1;i<100;i++) Integral += gap[i]*(3+pow(-1,i+1));

   Integral = Integral*dx/3;

   if(Integral) H = 1/(12*Integral);
   else{
     H = 0.0;
     printf("Integral = 0\n");
    }
  }

 return H;

}
```

C.4 Analysis for Current Time Step

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void Step(int nXsect,float dz,sourceptr Source,frontptr Front,frontptr Next2Front,
      int inNode,int refnode,double *t,double *dt,double *Dt,nodeptr node,faceptr face,
      float surfT,char *voidfile,float *vol,float Pref,groupinfo voidgroup,int *numvoid,
      char *solndat,float Pscale,float Pwave)
{
    int          i,k,j,errorNode,a,idQmax,iRmax,c;
    double       tmin,err,maxerror,maxP;
    float        sum,sumH,Qout,Pc,Qnet,dQ,Qin,dp,dp0,dQmax,Rmax;
    frontptr     AtFront,AtBehind,Behind,FrontEl,temp,frntptr;
    groupinfo    voidElem;
    nodeptr      tempnode,nptr,nptr2;
    faceptr      fptr;
    neighborptr  nbr;

    FrontEl = newFrontNode(-1);
    voidElem = newgroup(0);

    nptr = NULL;
    fptr = NULL;
    frntptr = NULL;
    nbr = NULL;

    Qout = 0;

    Qin = GetSource(inNode,Source);
    dp = 0;
```

```
/* Identify nodes and faces at flow front*/

/*Initialize lists. */
AtFront = newFrontNode(-1);          /* Nodes At front */
AtBehind = newFrontNode(-1);         /* Nodes At or AtBehind front */
Behind = newFrontNode(-2);     /* Initialize list of nodes Behind flow front */
FrontCheck(Source,node,face,AtFront,AtBehind,Behind,Next2Front,refnode);

 VoidNode(nXsect,inNode,dz,Next2Front,AtBehind,Behind,FrontEl,Front,voidElem,
        node,face,Next2Front,*t,voidfile,vol,voidgroup,numvoid);


/* Solve "compact" system */
Solver(node,face,Source,Behind,AtBehind,Next2Front,AtFront,voidgroup,inNode,
      refnode,Qin,surfT,Pref,*t,solndat,Pscale,Pwave);

printf("Full nodes:\n");

 frntptr = AtBehind->next;      /* Skip header */
 for(;frntptr!= NULL; frntptr = frntptr->next){
  nptr = getNode(frntptr->node,node);
  if(nptr->sat==1) printf("%d,",nptr->node);

    nptr->flag = 1;
  }

printf("\n\nNodes at front:\n");
  frntptr = AtFront->next;      /* Skip header */
 for(;frntptr!= NULL; frntptr = frntptr->next){
   printf("%d,",frntptr->node);
  }
printf("\n");


/* Calculate new time step, dt.*/
  tmin = 1.0e6;           /* Set initial value */
 NewTime(Source,*t,&tmin,node,face,AtFront,surfT,Pscale,refnode,voidgroup);

/* Set new time step. */
 if(tmin==1e6){
 if(*dt) *Dt = *dt;
 *dt = 0;
}
else *Dt = *dt = tmin;

printf("dt = %e  Dt = %e\n",*dt,*Dt);

/* Update front node saturations. Advance flow front by updating node
   saturations based on Qij calculated using P. */

 NewSat(Source,*dt,node,face,AtFront,surfT,Pscale,refnode,voidgroup);

 /* Update control volume saturations and pressures used in this time step */
 frntptr = AtBehind->next;      /* Skip header */
 for(;frntptr!= NULL; frntptr = frntptr->next){
  nptr = getNode(frntptr->node,node);
    nptr->sat0 = nptr->sat;
```

```
      nptr->P0 = nptr->P;
      }

      frntptr = AtFront->next;      /* Skip header */
      for(;frntptr!= NULL; frntptr = frntptr->next){
       nptr = getNode(frntptr->node,node);
        nptr->sat0 = nptr->sat;
       nptr->P0 = nptr->P;
       }

      /* Adjust H values if there is flow out of bundle into refnode. */
      nptr = getNode(refnode,node);
      if(nptr->flag) newH(node,face,Source,surfT,Pscale,refnode,*Dt);


}
```

C.5 Determination of Node Status

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>


void FrontCheck(sourceptr Source,nodeptr node,faceptr face,frontptr AtFront,frontptr AtBehind,
        frontptr Behind,frontptr Next2Front,int refnode)
{

     int       i,n,front,x;  /* x = a counter to keep track of neighbors */
     nodeptr        tempnode,nptr;
     neighborptr   tempN;
     float          Qout;

     Qout = 0;

     renewFrontNode(Next2Front);

/* Scan through all nodes */
     tempnode = node->next;              /* Skip header */
     for(;tempnode != NULL ;tempnode = tempnode->next){
      i = tempnode->node;

      /* Initialize by assuming that node i is NOT in flow domain. */
      front = x = 0;


       /* Check if node satisfies conditions for front status */
      /* Is CV i only partially full? */
```

```
    if((tempnode->sat) && (tempnode->sat<1.0)) front = 1;
    else if (!tempnode->sat){
      /* If CV i is empty, is at least 1 neighbor CV full? */
      /* Scan neighboring CV's */
      tempN = tempnode->neighbor->next;          /* Skip header */
      for(;tempN != NULL; tempN = tempN->next){
        x++;
        n = tempN->n;
        nptr = getNode(n,node);              /* Get pointer to node n */

        if(nptr->sat==1.0) (front)++;
      }

      /* Is node i a Source node? */

      if(GetSource(i,Source)){
        if (front<x) front = 1;
      }
      else if ((front<x) && (front)) front = 1;
      else if(i==refnode) front = 1;  /* Is i the reference node? */

    }
    else if (tempnode->sat==1.0){
      /* Check neighbhors to see if CV i is "Next2Front" */
      tempN = tempnode->neighbor->next;          /* Skip header */
      for(;tempN != NULL; tempN = tempN->next){
        x++;
        n = tempN->n;
        nptr = getNode(n,node);              /* Get pointer to node n */

        if(nptr->sat==1.0) (front)++;
      }
      if(front<x) front = 2;                /* CV i is Next2Front */
      else front = 3;                      /* CV i is Behind the front */
    }

    if(front==1){
      addFrontNode(AtFront,i);
      tempnode->flag = 0;      /* This is to accommodate newly "drained" CV's */
    }
    else if(front==2){
      addFrontNode(Next2Front,i);
      addFrontNode(AtBehind,i);
      if(!tempnode->flag) printf("New front node = %d\n",i);
    }
    else if(front==3){
      addFrontNode(Behind,i);
      addFrontNode(AtBehind,i);
    }
  }

  /* In order to take into account flow out to refnode at end of simulation */
  if(AtFront->next == NULL) addFrontNode(AtFront,refnode);

}
```

C.6 Void Identification

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void VoidNode(int nXsect,int inNode,float dz,frontptr AtFront,frontptr AtBehind,
              frontptr Behind,frontptr FrontEl,frontptr Front,groupinfo voidElem,
              nodeptr node,faceptr face,frontptr Next2Front,double t,char *voidfile,
              float *vol,groupinfo voidgroup,int *numvoid)
{
    int       i,j,k,kmax,kvent,El[NC],NextNode[NC],MaxFront,ventpath,pore,n,e,
              Kfront,flag;
    float     a[NC],vol2,satvol,avg,stdev,voidFraction,Volume;
    frontptr  temp,vent;
    groupinfo group,adj,tempgroup;
    vnodeptr  dry,templist;
    sourceptr voidVol,tempVol;
    nodeptr   tempnode,nptr;
    neighborptr tempN;
    faceptr   fptr;
    FILE      *fp;


    nptr = NULL;
    fptr = NULL;
    tempN = NULL;
    fp = NULL;


    vent = newFrontNode(-1); /* Initialize list of vent nodes */
                             /* Each ith entry will correspond to the ith pore */


    satvol = *vol = vol2 = avg = stdev = voidFraction = 0;
    MaxFront = *numvoid = kvent = 0;


    dry = newVnode(-2);      /* Initialize list of dry nodes behind vent plane */
    flag = 0;
    voidVol = newsource(0,0.0);

    temp = AtBehind->next;
    for( ;temp != NULL;temp = temp->next){
      i = temp->node;
      nptr = getNode(i,node);   /* Calculate saturated volume */
      if(i<inNode) satvol += nptr->sat*nptr->V;
      if((i>MaxFront)&&(i!=inNode)) MaxFront = i;
    }
```

```
    kvent = MaxFront/nXsect;

    /* Generate list of nodes in vent plane */
    n = 0;
    for(pore=0;pore<nXsect;pore++){
     n = pore + kvent*nXsect;
     addFrontNode(vent,n);
     /* Generate list of global front nodes */
     kmax = pore;
     Kfront = 0;
     for(k=0;k<=kvent;k++){
       Kfront = pore + k*nXsect;
       if(CheckList(Kfront,AtFront)){
         if(Kfront>kmax) kmax = Kfront;
        }
      }
     addFrontNode(Front,kmax);
    }



/* Scan for dry nodes behind vent plane (i.e., possible void nodes) */
    for(i=0;(i/nXsect)<kvent+1;i++){
     nptr = getNode(i,node);
     if(nptr->sat!=1) addVnode(dry,i);
    }

    if(dry->next!=NULL) flag = 1;
/*flag = 0;  */
if(flag){
    /* Form adjacency list for nodes in dry */
     adj = newgroup(-2);                    /* Skip header */
    templist = dry->next;                    /* Skip header */

    for(;templist != NULL;templist = templist->next){
     tempnode = getNode(templist->node,node);        /* Get pointer to node */
     addgroup(adj,templist->node);
     group = Scan2EndGroup(adj);

     /* Scan neighboring nodes */
     tempN = tempnode->neighbor->next;          /* Skip header */
     for(;tempN!=NULL;tempN=tempN->next){
       n = tempN->n;
       if(CheckVnodeList(n,dry)||(CheckList(n,vent))) addVnode(group->list,n);
      }
    }

    /* Add vent nodes to adjacency list */
    temp = vent->next;                      /* Skip header */
    for(;temp!=NULL;temp=temp->next){
     tempnode = getNode(temp->node,node);     /* Get pointer to node */
     addgroup(adj,temp->node);
     group = Scan2EndGroup(adj);

     /* Scan neighboring nodes */
     tempN = tempnode->neighbor->next;          /* Skip header */
     for(;tempN!=NULL;tempN=tempN->next){
       n = tempN->n;
```

```
    if(CheckVnodeList(n,dry)||(CheckList(n,vent))) addVnode(group->list,n);
   }
  }


 /* Use breadth first search and traversal to link nodes into voidgroups */
  LinkGroups(adj,voidgroup);
  printf("\nLinked groups of dry nodes behind front (possible voids):\n");
  printGroupList(voidgroup);

  /* Eliminate groups from voidgroup that are connected to vent nodes */
  group = voidgroup->next;      /* Skip header */
  for(;group != NULL; group = group->next){
    templist = group->list;
    ventpath = 0;
    do{
      if(CheckList(templist->node,vent)) ventpath++;
       templist = templist->next;

    }while((templist != NULL) && (!ventpath));

    if(ventpath) rmvGroup(group->label,group);
  }

  GroupRenumber(voidgroup);

  printf("\nVoids and void nodes:\n");
  printGroupList(voidgroup);

  /* Calculate void volume, vol. */
  templist = NULL;
  printf("\nCalculating void volume.\n");
  group = voidgroup->next;      /* Skip header */
  for(;group != NULL; group = group->next){
    (*numvoid)++;
    templist = group->list->next;
    for(;templist!=NULL;templist = templist->next){
      nptr = getNode(templist->node,node);

      /* Shut off flow to from/to neighboring nodes */
      tempN = nptr->neighbor->next;  /* Skip header */
      for(;tempN!=NULL;tempN=tempN->next){
       fptr = getFace(tempN->f,face);
       fptr->H = 0;
       nptr->sat = nptr->sat0;
       }

      *vol += (1-nptr->sat)*nptr->V;
      vol2 += pow((1-nptr->sat)*nptr->V,2);
    }
  }
  if(satvol + *vol){
    voidFraction = *vol/(satvol + *vol);
    printf("Saturated volume = %.2e\tVoid volume = %.2e\tVoid fraction = %.2e\n",
        satvol,*vol,voidFraction);
  }

  if(*numvoid){
```

```
        avg = *vol/(*numvoid);
        stdev = (vol2 - pow(*vol,2)/(*numvoid))/(*numvoid);
        stdev = sqrt(stdev);
        if(avg){
         printf("Average void volume = %.2e\t std dev = %.2e (%.1f%% of avg)\n",
             avg,stdev,100*stdev/avg);

         /* Open file for input */
         if ((fp = fopen(voidfile, "a")) ==NULL) {
          printf("cannot open file");
          exit(1);
          }

         fprintf(fp,"%f\t%d\t%e\t%e\t%e\t%e\t%e\t%e\n",t,*numvoid,satvol,*vol,
             voidFraction,avg,stdev,stdev/avg);

         fclose(fp);
        }
       }
      }

}
```

C.7 Solution of Pressure Distribution

```
/*NOTE: NEED (ROW) DIAGONAL DOMINANT COEFFICIENT MATRIX FOR CONVERGENCE */

#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void Solver(nodeptr node,faceptr face,sourceptr Source,frontptr Behind,frontptr AtBehind,
        frontptr Next2Front,frontptr AtFront,groupinfo voidgroup,int inNode,int refnode,
        float Qin,float surfT,float Pref,double t,char *solndat,float Pscale,float Pwave)

{
    int         i,k,j,errorNode,a,idQmax,newnode,maxPnode,fullcount;
    double      err,maxerror,maxP,dQold;
    float       W,M,Tol,sum,sumH,Pc,newP,dQ,dQmax,QTol;
    frontptr    frntptr;
    nodeptr     nptr,nptr2,newptr;
    faceptr     fptr;
    neighborptr nbr;
    FILE        *fp;


    W = 1;      /* Overrelaxation coefficient, typically 1.2<W<1.6
                   Note: W=1 yields Gauss-Seidel method */
```

```
M = 5e4;       /* Number of solution iterations */

if(Pwave) Tol = 1e-6;
else Tol = 1e-10;
/* Solution tolerance (relative to Pscale)
  if fibers are have waviness, then set relative to Pwave */

QTol = 1e-6;  /* Max nodal residual (relative to Qin) */
newnode = inNode;

nptr = nptr2 = newptr = NULL;
fptr = NULL;
frntptr = NULL;
nbr = NULL;

fullcount = 0;

 /* Transform pressures of nodes involved in pressure solution relative to Pref */
 frntptr = AtBehind->next;      /* Skip header */
 for(;frntptr!= NULL; frntptr = frntptr->next){
  fullcount++;
  nptr = getNode(frntptr->node,node);
   nptr->P0 -= Pref;
   nptr->P -= Pref;
   nptr->Ptemp = nptr->P;
  }

 frntptr = AtFront->next;      /* Skip header */
 for(;frntptr!= NULL; frntptr = frntptr->next){
  nptr = getNode(frntptr->node,node);
   nptr->P0 -= Pref;
   nptr->P -= Pref;
   nptr->Ptemp = nptr->P;
  }

/* Solve "compact" system */
k = 0;
dQold = 0;
do {

 errorNode = inNode;
 maxerror = err = 0;

 dQmax = 0;
 idQmax = inNode;
 frntptr = AtBehind->next;
 for( ; frntptr!= NULL; frntptr= frntptr->next){

   nbr = NULL;
   nptr = NULL;
   fptr = NULL;

   i = frntptr->node;

   sum = sumH = dQ = 0;
   dQold = dQmax;
   /* Get pointer to node i */
   nptr = getNode(i,node);
```

```
 sum += GetSource(i,Source);

 nbr = nptr->neighbor->next;    /* Skip header */
 for(;nbr!=NULL;nbr=nbr->next){
   a = 1;
   nptr2 = getNode(nbr->n,node);
   fptr = getFace(nbr->f,face);

   Pc = getPc(nptr2,fptr,node,face,surfT,Pscale,refnode);

   if(nptr2->sat<1){
    if(nptr->node<i){
      if(nptr->P - nptr2->P + Pc < 0) a = 0;
    }
    else{
      if(nptr->P - nptr2->Ptemp + Pc < 0) a = 0;
    }
    }

   sumH += a*fptr->H;
   if(nptr2->node<i) sum += a*fptr->H*(nptr2->P - Pc);
   else sum += a*fptr->H*(nptr2->Ptemp - Pc);
 }

 if(!sumH){
   nptr->P = nptr->Ptemp = nptr->P0; /* "Shut-off" void */
   dQ = 0;
 }
 else{
   nptr->P = (1-W)*nptr->Ptemp + W*sum/sumH;
   dQ = nptr->P*sumH - sum;
 }

 nptr->Q = dQ;

 err = fabs(nptr->P - nptr->Ptemp);
 if(err>maxerror){
   maxerror = err;
   errorNode = i;
 }

 if(fabs(dQ)>fabs(dQmax)){
   dQmax = dQ;
   idQmax = i;
 }

}

k++;

/* Set reference values */
nptr = getNode(inNode,node);
maxP = nptr->P;
maxPnode = inNode;

frntptr = AtBehind->next;      /* Skip header */
for(;frntptr!= NULL;frntptr=frntptr->next){
```

```
        nptr = getNode(frntptr->node,node);
        nptr->Ptemp = nptr->P;
        if(nptr->P>maxP){
          maxP = nptr->P;
          maxPnode = nptr->node;
        }
      }

      if(Pwave) maxerror /= Pwave;

      /* Stopping criteria */
/*    } while ((k<M)&&((maxerror>Tol)||((fabs(dQmax)>QTol)&&(fabs(dQmax-dQold)>QTol)))); */
      } while ((k<M)&&(maxerror>Tol));


      printf("\nk = %d fullcount = %d #Op's = %.1e  maxerror = %.2e at node: %d\n",k,fullcount,
          (float)k*fullcount,maxerror,errorNode);
      printf("maxP = %e at node: %d\n",maxP,maxPnode);
      printf("Max residual/Qin = %e  at node:  %d\n\n",dQmax,idQmax);

      /* Write data on solution to file. */
       fp = NULL;
       /* Open file for input */
      if ((fp = fopen(solndat, "a")) ==NULL) {
            printf("cannot open file");
            exit(1);
       }

      fprintf(fp,"%e\t%d\t%d\t%.1e\t%e\t%d\t%e\t%d\t%e\t%d\n",t,k,fullcount,(float)k*fullcount,
          maxerror,errorNode,maxP,maxPnode,dQmax,idQmax);

      fclose(fp);


      /* Transform pressures of nodes involved in pressure solution to abs values */
       frntptr = AtBehind->next;      /* Skip header */
      for(;frntptr!= NULL; frntptr = frntptr->next){
        nptr = getNode(frntptr->node,node);
        nptr->P0 += Pref;
        nptr->P += Pref;
        }

       frntptr = AtFront->next;      /* Skip header */
      for(;frntptr!= NULL; frntptr = frntptr->next){
        nptr = getNode(frntptr->node,node);
        nptr->P0 += Pref;
        nptr->P += Pref;
        }

}




float getPc(nodeptr nptr,faceptr fptr,nodeptr node,faceptr face,float surfT,float Pscale,
        int refnode)
{
```

```
 float  Pc;

 if(nptr->sat<1){
  if((Pscale)&&(fptr->Rcap)) Pc = 2*surfT/(Pscale*fptr->Rcap);
  else{
   if(!fptr->Rcap){
    printf("Error:  Division by zero.  Rcap = 0.\n");
    exit(1);
    }
   else if(!Pscale){
    printf("Error:  Division by zero.  Pscale = 0.\n");
    exit(1);
    }
  }
 }
 else Pc = 0;

 return Pc;
}


float getAvgPc(nodeptr nptr,nodeptr node,faceptr face,float surfT,float Pscale,int refnode)
{

 float        Pc,sumH;
 nodeptr      nptr2;
 faceptr      fptr;
 neighborptr  nbr;


 nptr2 = NULL;
 fptr = NULL;
 nbr = NULL;

 Pc = sumH = 0;
 nbr = nptr->neighbor->next;
 for(;nbr!=NULL;nbr=nbr->next){

  nptr2 = getNode(nbr->n,node);
  fptr = getFace(nbr->f,face);

  Pc += fptr->H*getPc(nptr2,fptr,node,face,surfT,Pscale,refnode);

  sumH += fptr->H;
 }

 Pc /= sumH;

 return Pc;

}
```

C.8 Calculation of New Time Step

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void NewTime(sourceptr Source,double t,double *tmin,nodeptr node,faceptr face,
        frontptr AtFront,float surfT,float Pscale,int refnode,groupinfo voidgroup)
{
      int           i,inNode,drain,count;
    float           Qnet,Qin,Qout;
    double          tfill;
    frontptr  temp;
    nodeptr         nptr;

    printf("\n");

    drain = count = 0;
    inNode = refnode - 1;
    Qin = GetSource(inNode,Source);
    Qout = 0;

     /* Check to see if there are any front nodes elligible for AtBehind list. */
     temp = AtFront->next;                  /* Skip header */
     for( ;temp != NULL;temp = temp->next){
     if(!CheckGroupList(temp->node,voidgroup)&&(temp->node!=refnode)) count++;
    }

    if(count){

     /* Consider only nodes at front */
     temp = AtFront->next;                  /* Skip header */
     for( ;temp != NULL;temp = temp->next){
       i = temp->node;
       nptr = getNode(i,node);              /* Get pointer to control volume i */

       /*Initialize local variables.*/
       Qnet = 0.0;
       tfill = 1.0e6;

       /* Get flows into control volume i */
       FrontFlow(nptr,Source,&Qnet,node,face,surfT,voidgroup,Qin,Pscale,refnode);

       if(Qnet){
        if(i!=refnode){
         /* Calculate filling time for CV i */
         if(Qnet > 0) tfill = (1-nptr->sat)*nptr->V/Qnet;
         /* Calculate emptying time for CV i */
         else if((Qnet<0)&&(nptr->sat)){
           tfill = 1.0e6;
           /*       tfill = -nptr->sat*nptr->V/Qnet;   */
         }
```

```
        }

        if(Qnet>0) Qout += Qnet;

       if((Qnet>0)||(i==refnode))
        printf("t[%d] = %e sat = %.2f  Q = %.2f%%\n",i,tfill,nptr->sat,100*Qnet/Qin);

        /* Compare filling time for CV i with tmin. */
       if((tfill)&&(tfill<(*tmin))){
        *tmin = tfill;
        if(Qnet<0) drain = 1;
        else drain = 0;
        }

      }

     }
/*
     if(drain) printf("dt = %e  DRAIN\n",*tmin);
     else printf("dt = %e\n",*tmin);
*/
     printf("Qout = %.2f%%\n",100*Qout/Qin);
    }

    if(!count){
     printf("\nNo elligible nodes remaining at t = %e.  Exit program.\n",t);
     exit(1);
    }
    else{
     if(*tmin >= 1.0e6){
      printf("\nTime step greater than maximum at t = %e.  Exit program. \n",t);
/*      exit(1); */
      *tmin = 1e6;
     }
     if(*tmin < 0){
      printf("Negative time step at t = %e.  Exit program.\n",t);
      exit(1);
     }
    }

}
```

C.9 Calculate Net Flow Into Control Volume

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
 #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>
```

```
void FrontFlow(nodeptr iptr,sourceptr Source,float *Qnet,nodeptr node,faceptr face,
        float surfT,groupinfo voidgroup,float Qs,float Pscale,int refnode)
{
    int       i,a;
    float     P,Pc,Pc2,Qc,Hout,sumH,sum,Qin,Qout;
    faceptr   fptr,fptr2;
    nodeptr   nptr,nptr2;
    neighborptr tempN,tempN2;


    fptr = fptr2 = NULL;
    nptr = nptr2 = NULL;
    tempN = tempN2 = NULL;

    i = iptr->node;

    *Qnet = 0;      /* Initialize net flow into control volume i */

    /* Incorporate flows into model.  Voids at inlets block source flows. */
    if(!CheckGroupList(i,voidgroup)){ /* Ignore void nodes */
     *Qnet += GetSource(i,Source);

     /* Scan faces neighboring iptr->node */
     tempN = iptr->neighbor->next;                /* Skip header */
     for(; tempN != NULL; tempN = tempN->next){
      nptr = getNode(tempN->n,node);             /* Get pointer to node n */

      /* Flow between control volumes iff at least 1 control volume is full. */
      if(nptr->sat0==1.0){
       fptr = getFace(tempN->f,face); /* Get pointer to face f */

       Pc = getPc(iptr,fptr,node,face,surfT,Pscale,refnode);

       if(nptr->P - iptr->P + Pc>0) *Qnet += fptr->H*(nptr->P - iptr->P + Pc);

      }
     }
    }
}
```

C.10 Update of Control Volume Saturations

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void NewSat(sourceptr Source,double Dt,nodeptr node,faceptr face,frontptr AtFront,float surfT,
```

```
      float Pscale,int refnode,groupinfo voidgroup)
{
    int       i,inNode;
    float          Qnet,delta,Qin,Qout;
    frontptr  temp;
    nodeptr        nptr;



    inNode = refnode - 1;
    Qin = GetSource(inNode,Source);
    Qout = 0;

    /* Consider only nodes at front */
    temp = AtFront->next;                /* Skip header */
    for( ;temp != NULL;temp = temp->next){
     i = temp->node;
     nptr = getNode(i,node);         /* Get pointer to control volume i */

     /*Initialize net flow into control volume i.*/
     Qnet = 0;
     delta = 0;

     /* Get flows into control volume i */
     FrontFlow(nptr,Source,&Qnet,node,face,surfT,voidgroup,Qin,Pscale,refnode);

     if(Qnet>0) Qout += Qnet;

     if(i!=refnode){
       /* Update saturation of control volume i. */
       nptr->sat += Dt*Qnet/nptr->V;
       delta = fabs(1 - nptr->sat);
       if(nptr->sat<0) nptr->sat = 0;
       if(nptr->sat - 1 > fTol){
         printf("\nWarning: sat[%d] = %e\n",i,nptr->sat);
          nptr->sat = 1;
        }
       if(delta<fTol) nptr->sat = 1;
      }
      else{
       nptr->sat = nptr->sat0 = 0;
       if(Qnet) nptr->flag = 1;     /* Set flag to adjust H before next time step. */
      }

    }
    printf("Qout = %.2f%%\n",100*Qout/Qin);
}
```

C.11 Adjustment of Conductance Matrix for Flow Out of Model

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
  #include <stdlib.h>
#endif
```

```
#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void newH(nodeptr node,faceptr face,sourceptr Source,float surfT,float Pscale,int refnode,float dt)
{
      int           i;
      float         Pc,Q,L;
      faceptr       fptr;
      nodeptr       iptr,nptr;
      neighborptr   nbr;


      fptr = NULL;
      nptr = NULL;
      nbr = NULL;

    iptr = getNode(refnode,node);
    i = iptr->node;

     Q = L = 0;

     /* Incorporate flows into model. */
    Q += GetSource(i,Source);

    /* Scan faces neighboring iptr->node */
    nbr = iptr->neighbor->next;              /* Skip header */
    for(; nbr != NULL; nbr = nbr->next){
     nptr = getNode(nbr->n,node);            /* Get pointer to node n */

     /* Flow between control volumes iff at least 1 control volume is full. */
     if(nptr->sat0==1.0){
       fptr = getFace(nbr->f,face); /* Get pointer to face f */

       Pc = getPc(iptr,fptr,node,face,surfT,Pscale,refnode);

       if(nptr->P - iptr->P + Pc > 0) Q = fptr->H*(nptr->P - iptr->P + Pc);

       if(fptr->A) L = dt*Q/fptr->A;
       else{
         printf("Error:  Division by zero in newH.  A = 0.\n");
         exit(1);
        }

       if(L){
         fptr->H *= fptr->L/L;
         fptr->L = L;
        }
       /* else don't change fptr->H or fptr->L */

     }
    }

}
```

C.12 Data Structure Utilities

```
#include "netdefs.h"
#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
 #include <stdlib.h>
#endif

#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

/***************************** FiberInfo Op's ************************/
fiberptr newFiber(int i,float *x,float d)
{
    int         j;
    fiberptr  new;

    new = NULL;
    if((new = (fiberptr) malloc(sizeof(struct FiberInfo))) != NULL) {
     new->i = i;

     for(j=0;j<3;j++) new->x[j] = x[j];

     for(j=0;j<2;j++) new->phi[j] = 0;

     new->d = d;
     new->next = NULL;

     return new;
    }
    else{
     printf("Error allocating memory for new fiber.  Program aborted.\n");
     exit(1);
    }
}

void addFiber(fiberptr fiber, int i,float *x,float d)
{
    int       j; /* control variable to stop for-loop after 1 iteration */
    fiberptr  temp;

    temp = fiber;
    j = 0;
    for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (j==0)){
       temp->next = newFiber(i,x,d);
        j++;
      }
    }
}


fiberptr getFiber(int i, float z,fiberptr fiber)
{
    fiberptr  temp,test;
```

```
    test = NULL;
    temp = fiber->next;          /* Skip header */

    for( ;temp != NULL;temp = temp->next){
     if(z==temp->x[2]){
       if(i==temp->i) test = temp;
     }
    }

    if(test!=NULL) return test;
    else{
     printf("Error getting pointer to fiber %d.  Program aborted.\n",i);
       exit(1);
     }
}

fiberptr Scan2EndFiber(fiberptr fiber)
{
    fiberptr  temp;

    temp = fiber;
    for( ;temp != NULL;temp = temp->next){
     if(temp->next == NULL) return temp;
    }

}


void printFiber(fiberptr fiber)
{

    int       i;
    fiberptr  temp;

    temp = fiber->next;              /* Skip header */
    for(;temp !=NULL;temp = temp->next){
     printf("Fiber: %d\t",temp->i);
     for(i=0;i<3;i++) printf("%.3f\t",temp->x[i]);
     printf("%.3f\n",temp->d);
    }
}



void ReadFiberData(fiberptr fiber,char *filename,int *numfibers)
/* Read fiber data at z = 0 from input text file */
{
    FILE    *data;
    int  i,j;
    float    x[3],d;

    i = 0;
    d = 0.0;
    for(j=0;j<3;j++) x[j] = 0.0;
    *numfibers = 0;

    /* Open text file (with line breaks) for output */
```

```
    if ((data = fopen(filename, "r")) ==NULL) {
      printf("cannot open file\n");
      exit(1);
    }

    while(!feof(data)){
      fscanf(data,"%d %f %f %f\n",&i,&x[0],&x[1],&d);
      addFiber(fiber,i,x,d);
      (*numfibers)++;
    }

    fclose(data);
}

void perturbFibers(int nz,float dz,int numfibers,fiberptr fiber,float pert,
    float deltaD,float *avgD,float *stdevD,float Lscale,float avgGap)
{
    int       i,j,k,count,Lcount;
    float          x[3],y[3],z,D,d,L,sumD,D2,dx[2],pert2,delta,x0,x1;
    fiberptr  temp1,temp2,fbrptr;

    pert2 = pow(pert,2);

    D = d = z = L = sumD = D2 = *avgD = *stdevD = delta = 0;
    j = 0;

    for(i=0;i<3;i++) x[i] = y[i] = 0;

    temp1 = temp2 = fbrptr = NULL;


    /* Get stats on fiber diameters at tow entrance */
    temp1 = fiber->next;                        /* Skip header */
    for(;(temp1 != NULL)&&(j<numfibers);temp1 = temp1->next){
      temp1->phi[0] = 0.5*PI*myrandom();
      temp1->phi[1] = PI*myrandom();
      D2 += (temp1->d)*(temp1->d);
      sumD += temp1->d;
    }

    temp1 = NULL;
    for(k=1;k<nz;k++){
      j = 0;
      z = k*dz;
      temp1 = fiber->next;                      /* Skip header */
      for(;(temp1 != NULL)&&(j<numfibers);temp1 = temp1->next){
        D = d = L = delta = 0;
        Lcount = 1;

        do{
          x0 = x1 = 0;
          for(i=0;i<2;i++) dx[i] = 0;

          delta = pert*0.5*avgGap*(1-cos(2*PI*z/Lscale + temp1->phi[1]));

          if(Lcount>1) delta *= myrandom();  /* If fiber overlap on first iteration,
                                                impose random perturbation on subsequent
                                                iterations */
```

```
        if(pert){
         /* Find x0 and x1 components of delta */
         dx[0] = delta*cos(temp1->phi[0]);
         dx[1] = delta*sin(temp1->phi[0]);
         }
        else dx[0] = dx[1] = 0;

        for(i=0;i<2;i++) x[i] = temp1->x[i] + dx[i];

/*      dx[0] = pert*myrandom();
        dx[1] = sqrt(pert2 - dx[0]*dx[0]);

        for(i=0;i<2;i++) x[i] = temp1->x[i] + pow(-1,evenOdd())*dx[i];
*/



/*
        for(i=0;i<2;i++) x[i] = (temp1->x[i]) + pow(-1,evenOdd())*pert*myrandom();

*/
        x[2] = z;
        D = (temp1->d) + pow(-1,evenOdd())*deltaD*myrandom();


        /* Check for overlap */
        if(j){
         count = 1;
         L = 0;
         temp2 = getFiber(1,z,fiber);     /* Get 1st fiber in layer */
         for(;(temp2 != NULL)&&(count<=j)&&(L>=0);temp2 = temp2->next){
           for(i=0;i<3;i++) y[i] = temp2->x[i];
           d = temp2->d;

           L = dist(x,y);
           L -= 0.5*(D+d);

           count++;
         }

         }
         Lcount++;
       } while (L<0);

       D2 += D*D;
       sumD += D;

      addFiber(fiber,temp1->i,x,D);
      fbrptr = getFiber(temp1->i,z,fiber);
      fbrptr->phi[0] = temp1->phi[0];
       j++;
     }
   }

   printf("Fibers perturbed OK\n");
```

```
    *avgD = sumD/(k*numfibers);
    *stdevD = (D2 - sumD*sumD/(k*numfibers))/(k*numfibers);
    *stdevD = sqrt(*stdevD);
    printf("Average D = %.3f\t std dev in D = %.3f (%.1f%% of Dnom)\n",*avgD,
        *stdevD,100*(*stdevD));

}


/***************************** PackingInfo Op's ************************/
packptr newCell(int cell)
{
    int       i;
    packptr new;

    if((new = (packptr) malloc(sizeof(struct PackingInfo))) != NULL) {
     new->cell = cell;
     new->xc = -1;
     new->yc = -1;

     for(i=0;i<3;i++){
      new->vertex[3] = -1;
      new->Az = new->L[3] = new->angle[3] = new->x[3] = new->y[3] = 0.0;
     }

     new->next = NULL;
    }
    return new;
}

void addCell(packptr basepack, int cell,int *vertex)
{
    int         i,j; /* control variable to stop for-loop after 1 iteration */
    packptr       temp;

    temp = basepack;
    j = 0;
    for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (j==0)){
       temp->next = newCell(cell);
       for(i=0;i<3;i++) temp->next->vertex[i] = vertex[i];
        j++;
     }
    }
}

packptr getCell(int cell, packptr basepack)
{
    packptr  temp;

    temp = basepack;

    for( ;temp != NULL;temp = temp->next){
     if(cell==temp->cell) return temp;
    }
```

```
}

packptr Scan2EndCell(packptr basepack)
{
     packptr   temp;

     temp = basepack;
     for( ;temp != NULL;temp = temp->next){
      if(temp->next == NULL) return temp;
     }

}

void printVertex(packptr basepack)
{
     int  i;
     packptr temp;

     temp = basepack->next;                  /* Skip header */
     for(;temp !=NULL;temp = temp->next){
      printf("Cell: %d Vertices: ",temp->cell);
      for(i=0;i<3;i++){
       printf("%d\t",temp->vertex[i]);
      }
      printf("\n");
     }
}



void ReadVertexData(packptr basepack,int *cellnum,char *filename)
/* Read data on vertex fiber numbers for each cell from input text file */
{
     FILE     *data;
     int      i,cell,vertex[3];

     *cellnum = 0;
     cell = -1;
     for(i=0;i<3;i++) vertex[i] = -1;

     /* Open text file (with line breaks) for output */
     if ((data = fopen(filename, "r")) ==NULL) {
      printf("cannot open file\n");
      exit(1);
     }

     while(!feof(data)){
      (*cellnum)++;
      fscanf(data,"%d ",&cell);
      for(i=0;i<3;i++) fscanf(data,"%d ",&vertex[i]);
      addCell(basepack,cell,vertex);
     }

     fclose(data);
}

/***************************** GapInfo Op's ************************/
gapptr newGap(int i,int v1,int v2)
```

```
{
    gapptr    new;

    if((new = (gapptr) malloc(sizeof(struct GapInfo))) != NULL) {
     new->i = i;
     new->v1 = v1;
     new->v2 = v2;
     new->next = NULL;
    }
    return new;
}

void addGap(gapptr gap,int i,int v1,int v2)
{
    int       j; /* control variable to stop for-loop after 1 iteration */
    gapptr          temp;

    temp = gap;
    j = 0;
    for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (j==0)){
       temp->next = newGap(i,v1,v2);
        j++;
     }
    }
}


int checkGap(gapptr gap,int u,int v,int k,int nXsect)
{
    int       test,slice;
    gapptr    temp;

    test = slice = 0;
    temp = gap;

    if(k){
     for(; temp!=NULL; temp=temp->next){
       slice = temp->i/nXsect;
      if(slice==k){
        if(temp->v1==u){
         if(temp->v2==v) test++;
         }
        else if(temp->v1==v){
         if(temp->v2==u) test++;
         }
       }
     }
    }
    else{
     for(; temp!=NULL; temp=temp->next){
      if(temp->v1==u){
        if(temp->v2==v) test++;
       }
      else if(temp->v1==v){
        if(temp->v2==u) test++;
       }
     }
```

```
    }

    return test;
}


gapptr    getGap(gapptr gap,int u,int v,int k,int nXsect)
{
    int       slice;
    gapptr    temp;

    slice = 0;
    temp = gap;


    if(k){
     for(; temp!=NULL; temp=temp->next){
       slice = temp->i/nXsect;
      if(slice==k){
        if(temp->v1==u){
         if(temp->v2==v) return temp;
         }
        else if(temp->v1==v){
         if(temp->v2==u) return temp;
         }
       }
     }
    }
    else{
     for(; temp!=NULL; temp=temp->next){
      if(temp->v1==u){
        if(temp->v2==v) return temp;
       }
      else if(temp->v1==v){
        if(temp->v2==u) return temp;
       }
     }
    }
}



/****************************** FrontInfo Op's ***********************/
frontptr newFrontNode(int i)
{
    frontptr new;

    if((new = (frontptr) malloc(sizeof(struct FrontInfo))) != NULL) {
     new->node = i;
     new->next = NULL;

     return new;
    }
    else{
     printf("Error allocating frontptr memory.  Program aborted.\n");
     exit(1);
    }
```

```
}

void addFrontNode(frontptr list, int i)
/* Add front nodes in ascending numerical order. */
{
    int       j; /* control variable to stop for-loop after 1 iteration */
    frontptr  temp,tempnext;

    temp = tempnext = NULL;

    temp = list;
    j = 0;
    do{
     if(temp->node<i){
      if(temp->next==NULL){
       temp->next = newFrontNode(i);
        j++;
      }
      else if(temp->next->node>i){
       tempnext = temp->next;
       temp->next = newFrontNode(i);
       temp->next->next = tempnext;
        j++;
      }
     }
     temp = temp->next;
    } while ((temp != NULL) && (!j));

/*  for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (j==0)){
      temp->next = newFrontNode(i);
       j++;
     }
    }
*/
}

void rmvFrontNode(int i, frontptr list)
{
    frontptr  temp,temp2,temp3;

    temp = temp2 = temp3 = NULL;


    for( ;temp != NULL;temp = temp->next){
     if(temp->node==i){
       temp2 = list;
      for(;temp2!=NULL;temp2=temp2->next){
       if(temp2->next==temp){
        temp2->next = temp->next;
        temp3 = temp;
        temp = temp->next;
        free(temp3);
        return;
       }
      }
     }
    }
```

```
}


void renewFrontNode(frontptr list)
{

    list->next = NULL;


}

void printList(frontptr list)
{
    frontptr temp;

    temp = list->next; /* Disregard the first item in list */

    for( ;temp != NULL;temp = temp->next) printf("%d\t",temp->node);
    printf("\n\n");

}

int CheckList(int i, frontptr list)
{
    int test;
    frontptr temp;

    test = 0;
    temp = list->next; /* Disregard the first item in list */

    for( ;temp != NULL;temp = temp->next){
     if(i==temp->node) test = 1;
    }

    return test;

}

/* ========== Source Op's ========== */

sourceptr newsource(int i, float Q)
{
    sourceptr new;

    if((new = (sourceptr) malloc(sizeof(struct SourceInfo))) != NULL) {
     new->i = i;
     new->Q = Q;
     new->next = NULL;
    }
    return new;
}

void addsource(sourceptr list, int i, float Q)
{
    int j; /* control variable to stop for-loop after 1 iteration */
    sourceptr temp;
```

```
    temp = list;
    j = 0;
    for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (j==0)){
       temp->next = newsource(i,Q);
        j++;
     }
    }
}

void ReadSourceData(sourceptr Source,char *filename)
/* Read Source data from input file (text with line breaks) */
{
    FILE    *data;
    int  i;
    float    Q;


    /* Open text file (with line breaks) for output */
    if ((data = fopen(filename, "r")) ==NULL) {
     printf("cannot open file\n");
     exit(1);
    }

    while(!feof(data)){
     fscanf(data,"%d %f\n",&i,&Q);
     addsource(Source,i,Q);
    }

    fclose(data);
}



void ChangeSource(sourceptr list, int i, float Q)
{
    sourceptr temp;

    temp = list->next;      /* Disregard the first item in list */

    for( ;temp != NULL;temp = temp->next){
     if(i==temp->i) temp->Q = Q;
    }
}

float GetSource(int i, sourceptr list)
{
    sourceptr temp;
    float Q;

    temp = list->next;      /* Disregard the first item in list */
    Q = 0.0;                /* Initialize Q */

    for( ;temp != NULL;temp = temp->next){
     if(i==temp->i) Q = temp->Q;
    }

    return Q;
```

```
}

int CheckSource(int i, sourceptr list)
{
     sourceptr temp;
     float test;

     temp = list->next;      /* Disregard the first item in list */
     test = 0;               /* Initialize test */

     for( ;temp != NULL;temp = temp->next){
      if(i==temp->i) test = 1;
     }

     return test;
}


void printSourceList(sourceptr list)
{
     sourceptr temp;

     temp = list->next; /* Disregard the first item in list */

     for( ;temp != NULL;temp = temp->next)
      printf("%d\t%.1e\n",temp->i,temp->Q);

     printf("\n\n");

}

void fprintList(frontptr list)
{
     frontptr temp;
     FILE *fp;


     /* open file for input */
     if ((fp = fopen("Solndata", "a")) ==NULL) {
      printf("cannot open file");
      exit(1);
     }

     temp = list->next; /* Disregard the first item in list */

     for( ;temp != NULL;temp = temp->next) fprintf(fp,"%d\t",temp->node);
     fprintf(fp,"\n");

     fclose(fp);

}

/* ========== Save this guy! ========== */
void printvoidList(groupinfo label)
{
     groupinfo temp;

     temp = label;
```

```
    for( ;temp != NULL;temp = temp->next){
     printf("%d:\t",temp->label);
     printVnodeList(temp->list);
    }
}
/*=============================================================*/

vnodeptr newVnode(int i)
{
    vnodeptr new;

    if((new = (vnodeptr) malloc(sizeof(struct voidnodetype))) != NULL) {
     new->node = i;
     new->prev = NULL;
     new->next = NULL;

     return new;
    }
    else{
     printf("Error allocating newVnode. Program aborted.\n");
     exit(1);
    }
}

void addVnode(vnodeptr list, int i)
{
    int         k; /* control variable to stop for-loop after 1 iteration */
    vnodeptr    temp;

    temp = list;
    k = 0;
    for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (k==0)){
       temp->next = newVnode(i);
       temp->next->prev = temp;
        k++;
     }
    }
}

vnodeptr Scan2EndVnode(vnodeptr list)
{
    vnodeptr    temp;

    temp = list;
    for( ;temp != NULL;temp = temp->next){
     if(temp->next == NULL) return temp;
    }

}



void printVnodeList(vnodeptr list)
{
    vnodeptr temp;
```

```
    temp = list->next;              /* Disregard first element in list */
    for( ;temp != NULL;temp = temp->next){
      printf("%d\t",temp->node);
    }

    printf("\n\n");

}

int CheckVnodeList(int i, vnodeptr list)
{
    int             test;
    vnodeptr        temp;

    test = 0;
    temp = list;

    for( ;temp != NULL;temp = temp->next){
      if(i==temp->node) test = 1;
    }

    return test;

}


vnodeptr GetVnode(int i, vnodeptr list)
{
    vnodeptr        temp;

    temp = list;

    for( ;temp != NULL;temp = temp->next){
      if(i==temp->node) return temp;
    }

}

void rmvnode(int i, vnodeptr list)
{
    vnodeptr        temp,temp2;

    temp = temp2 = NULL;
    temp = list;
    for( ;temp != NULL;temp = temp->next){
      if(i==temp->node){
        if(temp->next!=NULL){
          temp->next->prev = temp->prev;
          temp->prev->next = temp->next;
        }
        else temp->prev->next = temp->next;

        temp2 = temp;
    /*    free(temp2); */
        return;
      }
    }
}
```

```
/* ########################################################################## */
groupinfo newgroup(int i)
{
    groupinfo new;

    if((new = (groupinfo) malloc(sizeof(struct GroupInfoType))) != NULL) {
     new->label = i;
     new->prev = NULL;
     new->next = NULL;
     new->list = newVnode(-2);

     return new;
    }
    else{
     printf("Error allocating newgroup.  Program aborted.\n");
     exit(1);
    }
}

void addgroup(groupinfo list, int i)
{
    int         k; /* control variable to stop for-loop after 1 iteration */
    groupinfo   temp;

    temp = list;
    k = 0;
    for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (k==0)){
       temp->next = newgroup(i);
       temp->next->prev = temp;
        k++;
     }
    }
}

groupinfo Scan2EndGroup(groupinfo group)
{
    groupinfo   temp;

    temp = group;
    for( ;temp != NULL;temp = temp->next){
     if(temp->next == NULL) return temp;
    }

}


void printGroupLabel(groupinfo group)
{
    groupinfo   temp;

    temp = group;

    for( ;temp != NULL;temp = temp->next) printf("%d\t",temp->label);

    printf("\n\n");
```

```
}

void printGroupList(groupinfo group)
{
    groupinfo    temp;

    temp = group;
    for(;temp != NULL;temp = temp->next){
      printf("%d:\t",temp->label);
      printVnodeList(temp->list);
    }

    printf("\n\n");

}

int CheckGroupLabel(int i, groupinfo group)
{
    int          test;
    groupinfo    temp;

    test = 0;
    temp = group;

    for( ;temp != NULL;temp = temp->next){
      if(i==temp->label) test = 1;
    }

    return test;

}

int  CheckGroupList(int i,groupinfo group)
{
    int          test;
    groupinfo    temp;
    vnodeptr     list;

    test = 0;
    temp = group->next;        /* Skip header */

    for(;temp != NULL;temp = temp->next){
      list = temp->list;
      for(;list != NULL;list = list->next){
        if(i==list->node) test = 1;
      }
    }

    return test;

}

groupinfo GetGroup(int i, groupinfo group)
{
    groupinfo    temp;

    temp = group;
    for( ;temp != NULL;temp = temp->next){
```

```
      if(i==temp->label) return temp;
    }

}


void rmvGroup(int i, groupinfo list)
{
    groupinfo    temp,temp2;

    temp = temp2 = NULL;
    temp = list;

    for( ;temp != NULL;temp = temp->next){
     if(i==temp->label){
      if(temp->next!=NULL){
        temp->next->prev = temp->prev;
        temp->prev->next = temp->next;
       }
      else temp->prev->next = temp->next;

      temp2 = temp;
      free(temp2);
      return;
     }
    }
}

void LinkGroups(groupinfo group,groupinfo voidgroup)
/* Use Breadth First Search and Traversal of undirected graph,group */
{
    groupinfo    tempgroup;
    vnodeptr     templist,visit,newvisit;
    int        count,first;


    tempgroup = NULL;
    templist = visit = newvisit = NULL;

    visit = newVnode(-1);                /* Skip header */
    tempgroup = group;

/* Breadth first traversal of undirected graph, group, to get voidgroups. */
    count = 0;
    first = group->next->label;      /* Initial group for search. Skip header. */

    do{
     newvisit = newVnode(0);
     Search(first,group,visit,newvisit);

      count++;
     addgroup(voidgroup,count);
     templist = newvisit->next;          /* Skip header, arbitrary 0. */
     tempgroup = Scan2EndGroup(voidgroup);
     for(;templist != NULL;templist = templist->next){
      addVnode(tempgroup->list,templist->node);
     }
```

```
      first = GetNext2Visit(group,visit);

    } while(first);
}

void Search(int first,groupinfo group,vnodeptr visit,vnodeptr newvisit)
/* Use breadth first search and traversal to determine connected void groups. */

{
    vnodeptr    queue;
    int         next;

    next = 0;
    queue = NULL;
    queue = newVnode(0);

    Explore(first,group,visit,newvisit,queue);

    do{
     if(queue->next != NULL){
       next = queue->next->node;
       rmvnode(next,queue);
       Explore(next,group,visit,newvisit,queue);
      }
    } while (queue->next!=NULL);
}

void Explore(int i,groupinfo group,vnodeptr visit,vnodeptr newvisit,
             vnodeptr queue)
/* Breadth First Search of undirected graph,group */
{
    groupinfo   tempgroup;
    vnodeptr    templist;
    int         j;


    tempgroup = NULL;
    templist = NULL;

    tempgroup = GetGroup(i,group);

    if(tempgroup!=NULL){    /* For safety...*/
     if(!CheckVnodeList(i,visit)){
       addVnode(visit,i);
       addVnode(newvisit,i);
       templist = tempgroup->list->next;   /* Skip header */
       for(;templist!=NULL;templist=templist->next){
         j=templist->node;
         if(!CheckVnodeList(j,visit)) addVnode(queue,j);
        }
      }
    }
}

int GetNext2Visit(groupinfo group,vnodeptr visit)
{
    groupinfo   tempgroup;
    vnodeptr    templist;
```

```
    int             i,next;

    tempgroup = group->next;      /* Skip header */
    next = 0;

    do{
     i = tempgroup->label;
     if(!CheckVnodeList(i,visit)) next = i;
     tempgroup = tempgroup->next;
    } while ((tempgroup != NULL) && (!next));

    return next;
}

int CheckVector(int goal,int *vector,int size)
{
    int  count,test;

    test = count = 0;

    do{
     if(vector[count]==goal) test++;
     count++;
    } while ((count<size) && (!test));

    return test;
}

void GroupRenumber(groupinfo group)
{
    int             count;
    groupinfo    temp;

    count = 1;
    temp = group->next;       /* Skip header */

    for( ;temp != NULL;temp = temp->next){
     temp->label = count;
     count++;
    }
}
/* ************************* FaceInfo Op's ************************************/


faceptr newFace(int F,int u,int v)
{
    faceptr new;

    if((new = (faceptr) malloc(sizeof(struct FaceInfo))) != NULL) {
     new->F = F;
     new->u = u;
     new->v = v;
     new->A = 0;
     new->L = 0;
     new->H = 0;
     new->Rcap = 0;
     new->next = NULL;
    }
```

```
    return new;
}

void addFace(faceptr face,int F,int u,int v)
{
    int             k; /* control variable to stop for-loop after 1 iteration */
    faceptr   temp;

    temp = face;
    k = 0;
    for( ;temp != NULL;temp = temp->next){
     if((temp->next == NULL) && (k==0)){
       temp->next = newFace(F,u,v);
        k++;
      }
    }
}

faceptr Scan2EndFace(faceptr face)
{
    faceptr   temp;

    temp = face;
    for( ;temp != NULL;temp = temp->next){
     if(temp->next == NULL) return temp;
    }

}

faceptr getFace(int F,faceptr face)
{
    faceptr   temp;

    temp = face;
    for( ;temp != NULL;temp = temp->next){
     if(temp->F == F) return temp;
    }

}

faceptr   findFace(faceptr face,int u,int v)
{
    faceptr   temp;

    temp = face;


    for(; temp!=NULL; temp=temp->next){
     if(temp->u==u){
       if(temp->v==v) return temp;
      }
     else if(temp->u==v){
       if(temp->v==u) return temp;
      }
    }
}
```

```
void printFace(faceptr face)
{

    faceptr temp;

    temp = face->next;                  /* Skip header */
    for(;temp !=NULL;temp = temp->next){
      printf("F: %d\tu: %d\tv: %d\n",temp->F,temp->u,temp->v);
    }
}



void ReadFaceData(faceptr face,char *filename,int *nXsect,
    int *fXsect,int *numSlice)
/* Read face data from input text file */
{
    FILE    *data;
    int     F,u,v,FperSlice;

    F = u = v = -1;

    /* Open text file (with line breaks) for output */
    if ((data = fopen(filename, "r")) ==NULL) {
      printf("cannot open file\n");
      exit(1);
    }

    /* Read #nodes and #faces in X-sect */
    fscanf(data,"%d\n",nXsect);
    fscanf(data,"%d\n",fXsect);

    /* Read number of "slices" of tow in axial direction to be modeled */
    fscanf(data,"%d\n",numSlice);

    while(!feof(data)){
      fscanf(data,"%d %d %d\n",&F,&u,&v);
      addFace(face,F,u,v);
    }

    fclose(data);
}

faceptr addSlices(faceptr base,int nXsect,int fXsect,int numSlice)
{
    int     i,j,k,FperSlice,F,u,v;
    faceptr   face,temp;


    face = newFace(-1,-1,-1);     /* Skip header */
    FperSlice = fXsect + nXsect;

    /* Add faces of base and additional slices into face */
    for(k=0;k<numSlice;k++){

      /* Add faces in next X-section */
```

```
      for(i=0;i<fXsect;i++){
        temp = getFace(i,base);
        F = temp->F + k*FperSlice;
        u = temp->u + k*nXsect;
        v = temp->v + k*nXsect;
        addFace(face,F,u,v);
      }

      /* Add linking faces */

      for(i=0;i<nXsect;i++){
        F++;
        u = i + k*nXsect;
        v = u + nXsect;
        addFace(face,F,u,v);
      }
    }

    return face;
}


/* ************************* NodeInfo Op's ***************************** */
nodeptr newNodeData(int n)
{
    int i;
    nodeptr new;

    if((new = (nodeptr) malloc(sizeof(struct NodeInfo))) != NULL) {
      new->node = n;
      for(i=0;i<3;i++) new->x[i] = 0.0;
      new->V = 0;
      new->Vf = 0;
      new->Rh = 0;
      new->sat = 0;
      new->sat0 = 0;
      new->Q = 0;
      new->P = 0;
      new->P0 = 0;
        new->Ptemp = 0;
      new->flag = 0;        /* Flag = 1 if AtBehind */
      new->neighbor = newNeighbor(-1,-1);
      new->next = NULL;
    }
    return new;
}

void addNodeData(nodeptr node,int n)
/* Insert nodes into list in ascending numerical order */
{
    int       k;    /* control variable to stop for-loop after 1 iteration */
    nodeptr   temp,tempnext;

    temp = node;
    k = 0;
    do{
      if(temp->node<n){
```

```
        if(temp->next==NULL){
          temp->next = newNodeData(n);
           k++;
         }
         else if(temp->next->node>n){
           tempnext = temp->next;
          temp->next = newNodeData(n);
           temp->next->next = tempnext;
           k++;
         }
        }
        temp = temp->next;
      } while ((temp != NULL) && (!k));
}


nodeptr Scan2EndNodeData(nodeptr node)
{
      nodeptr   temp;

      temp = node;
      for( ;temp != NULL;temp = temp->next){
        if(temp->next == NULL) return temp;
      }

}

nodeptr getNode(int n,nodeptr node)
{
      nodeptr   temp;

      temp = node;
      for( ;temp != NULL;temp = temp->next){
        if(temp->node == n) return temp;
      }
      temp = NULL;
      return temp;

}


int   CheckNode(int goal,nodeptr node)
{
      int         test;
      nodeptr   temp;

      test = 0;
      temp = node;

      for(;temp != NULL;temp = temp->next){
        if(temp->node==goal) test = 1;
      }

      return test;

}
```

```
void printNodeData(nodeptr node)
{

    nodeptr temp;

    temp = node->next;    /* Skip header */
    for(;temp !=NULL;temp = temp->next){
     printf("node: %d\n",temp->node);
    }
}

void printNodeNeighbors(nodeptr node)
{

    nodeptr temp;

    temp = node->next;    /* Skip header */
    for(;temp !=NULL;temp = temp->next){
     printf("node: %d\n",temp->node);
     printNeighbor(temp->neighbor);
    }
}


nodeptr MakeNodeList(faceptr face)
{
    int       F,x,y,count;
    faceptr       tempf;
    nodeptr       node,tempnode;

    node = newNodeData(-1);       /* Skip header */

    tempf = face;                 /* Skip header */
    for(;tempf != NULL; tempf = tempf->next){
     F = tempf->F;
     x = tempf->u;
     y = tempf->v;
     count = 0;
     do{
      if(!CheckNode(x,node)){
       addNodeData(node,x);
       tempnode = getNode(x,node);
       addNeighbor(tempnode->neighbor,F,y);
      }
      else{
       tempnode = getNode(x,node);
       addNeighbor(tempnode->neighbor,F,y);
      }
      x = y;
      y = tempf->u;
      count++;
     } while (count<2);    /* 2 Since each face has only 2 endpts */
    }
    return node;

}
```

```
/* ********************** NeighborInfo Op's ************************** */
neighborptr newNeighbor(int f,int n)
{
    neighborptr new;

    if((new = (neighborptr) malloc(sizeof(struct NeighborInfo))) != NULL) {
      new->f = f;
      new->n = n;
      new->prev = NULL;
      new->next = NULL;
    }
    return new;
}

void addNeighbor(neighborptr neighbor,int f,int n)
{
    int             k; /* control variable to stop for-loop after 1 iteration */
    neighborptr   temp;

    temp = neighbor;
    k = 0;
    for( ;temp != NULL;temp = temp->next){
      if((temp->next == NULL) && (k==0)){
        temp->next = newNeighbor(f,n);
         k++;
      }
    }
}

neighborptr Scan2EndNeighbor(neighborptr neighbor)
{
    neighborptr   temp;

    temp = neighbor;
    for( ;temp != NULL;temp = temp->next){
      if(temp->next == NULL) return temp;
    }

}

void printNeighbor(neighborptr neighbor)
{

    neighborptr   temp;

    temp = neighbor->next;              /* Skip header */
    for(;temp !=NULL;temp = temp->next){
      printf("f: %d\tn: %d\n",temp->f,temp->n);
    }
}
```

C.13 File Utilities

```
#include "netdefs.h"
#include <stdio.h>
```

```
#if defined(__STDC__) && !defined(__HIGHC__)
 #include <stdlib.h>
#endif


#include <string.h>
#include <stddef.h>
#include <math.h>
#include <time.h>

void fPrintFront(frontptr Front,float dz,nodeptr node,char *filename)
{
    int       i,layer,count;
    float         stdev,avgZ,Z2,sumZ;
    frontptr  fptr;
    nodeptr       nptr;
    FILE          *fp;

    fp = NULL;
    nptr = NULL;
    fptr = NULL;

    /* Open file for input */
    if ((fp = fopen(filename, "w")) ==NULL) {
        printf("cannot open file");
        exit(1);
    }

    layer = count = 0;
    stdev = Z2 = sumZ = avgZ = 0;
    fptr = Front->next;          /* Skip header */
    for(;fptr!=NULL;fptr=fptr->next){
        nptr = getNode(fptr->node,node);    /* Get pointer to node at front */

        for(i=0;i<3;i++) fprintf(fp,"%f\t",nptr->x[i]);
        fprintf(fp,"%.3f\n",nptr->sat);
        sumZ += nptr->x[2];
        Z2 += (nptr->x[2])*(nptr->x[2]);
        count++;
    }

    layer = sumZ/count;
    avgZ = layer*dz;
    stdev = (Z2 - sumZ*sumZ/count)/count;
    stdev = sqrt(stdev);
    printf("stdev = %.3f\n",stdev);

    if(avgZ) fprintf(fp,"Average Front Z = %.3f\tstd dev in Z = %.3f (%.1f%% of avgZ)\n",
        avgZ,stdev,100*stdev/avgZ);
    else fprintf(fp,"Average Front Z = %.3f\tstd dev in Z = %.3f\n",
        avgZ,stdev);


    fclose(fp);
}

void fFrontGrowth(double t,float satVol,frontptr Front,frontptr Full,float dz,int nz,
          nodeptr node,char *filename,float *minZ,float *avgZ,float *maxZ,
```

```
        int numfibers,float visc,sourceptr XsectVf,float avgHz,float Dnom,int inNode)

/* Write data on front progression and avg Kz/Dnom^2 to filename */
{
    int       count,z;
    float          stdev,Z2,Q,constant,exitVf,L,dp,Vexit,Kz;
    frontptr  fptr;
    nodeptr        nptr;
    FILE           *fp;

    nptr = NULL;
    fptr = NULL;
    fp = NULL;

    /* Open file for input */
     if ((fp = fopen(filename, "a")) ==NULL) {
      printf("cannot open file");
      exit(1);
      }


    /* Scan front nodes to determine minimum sat layer (i.e., ref layer for stats) */
    *maxZ = *avgZ = Q = constant = exitVf = L = dp = Vexit = Kz = 0;
    *minZ = nz*dz;
    count = z = 0;
    stdev = Z2 = 0;
    fptr = Front->next;          /* Skip header */
    for(;fptr!=NULL;fptr=fptr->next){
        nptr = getNode(fptr->node,node);    /* Get pointer to node at front */

        if(nptr->x[2] < *minZ) *minZ = nptr->x[2];
        if(nptr->x[2] > *maxZ) *maxZ = nptr->x[2];
        *avgZ += nptr->x[2];
        count++;
    }
    *avgZ /= count;

    count = 0;
    /* Scan front nodes to calculate std dev of front profile rel. to min sat layer */
    fptr = Front->next;          /* Skip header */
    for(;fptr!=NULL;fptr=fptr->next){
        nptr = getNode(fptr->node,node);    /* Get pointer to node at front */

        Z2 += pow((nptr->x[2] - *minZ),2);
        count++;
    }

    stdev = sqrt(Z2/count);

    /* Calculate average Kz/Dnom^2 */


    if(*minZ){
      z = (int)(*minZ);
      constant = 4*visc*avgHz/(numfibers*PI*Dnom*Dnom*Dnom);
      exitVf = GetSource(z,XsectVf);
      printf("exitVf = %e\n",exitVf);
      if(exitVf) Vexit = dz*(0.25*numfibers*PI*Dnom*Dnom)*(1-exitVf)/exitVf;
```

```
        else{
          printf("Error:  division by zero.  exitVf = 0\n");
          exit(1);
        }

        L = *minZ + 1;  /* "+1" because inNode located at z = -1 */


        fptr = Full->next;
        for(;fptr!=NULL;fptr=fptr->next){
          nptr = getNode(fptr->node,node);

          if((nptr->x[2]==(*minZ))&&(*minZ)){
            dp += nptr->V*nptr->P;
          }

        }

        dp /= Vexit;

        nptr = getNode(inNode,node);

        dp = fabs(nptr->P - dp);

      /*  Kz = constant*L*exitVf/dp;*/
        Kz = 0.5*visc*avgHz*L*L/(dp*t);
        printf("Saturated pressure drop = %e  Kz = %e\n",dp,Kz);
      }


      if(t) Q = satVol/t;

      printf("minZ = %e  avgZ = %e wetting range = %e stdev = %e\n",*minZ,*avgZ,*maxZ - *minZ,
            stdev);
      printf("avgQ = %e\n",Q);
      fprintf(fp,"%e\t%e\t%e\t%e\t%e\t%e\t%e\n",t,*minZ,*avgZ,*maxZ - *minZ,stdev,Q,Kz);

      fclose(fp);

}


void fPrintXsectData(float time,int step,int nz,float dz,nodeptr node,char *filename,
            char option)
{
      int  i,j;
      FILE     *fp;
      nodeptr  temp;
      char     *in1,*in2,*output;

      fp = NULL;
      temp = NULL;
      in1 = NULL;
      in2 = NULL;

      in1 = "temp2";
      in2 = "tempname";
```

```
    /* open file for input */
    if ((fp = fopen(in1, "w")) ==NULL) {
        printf("cannot open file");
        exit(1);
    }

    fprintf(fp,"%s\n",filename);
    fclose(fp);

    for(j=0;j<nz;j++){
        output = NULL;

        output = fReadName(in1);
        printf("%s\t",output);
        fPrintName(output,".",j);

        output = NULL;
        output = fReadName(in2);
        printf("%s\n",output);

        fp = NULL;
        /* open file for input */
        if ((fp = fopen(output, "w")) ==NULL) {
            printf("cannot open file");
            exit(1);
        }

        fprintf(fp,"%.3f\t%d\n",time,step);


        temp = node->next;              /* Skip header */
        for(;temp!=NULL;temp=temp->next){
            if(temp->x[2]/dz==j){
                for(i=0;i<2;i++) fprintf(fp,"%.3f\t%",temp->x[i]);
                if(option=='s') fprintf(fp,"%.3f\n",temp->sat);
                else if(option=='P') fprintf(fp,"%.3f\n",temp->P);
            }
        }

        fprintf(fp,"\n");
        fclose(fp);
    }
}

void fPrintFaceData(float time,int step,faceptr face,char *filename)
{
    FILE        *fp;
    faceptr     temp;

    /* open file for input */
    if ((fp = fopen(filename, "w")) ==NULL) {
        printf("cannot open file");
        exit(1);
    }

    fprintf(fp,"%.3f\t%d\n",time,step);

    temp = face->next;              /* Skip header */
```

```
    for(;temp!=NULL;temp=temp->next){
        fprintf(fp,"%d\t%d\t%d\n",temp->F,temp->u,temp->v);
    }

    fprintf(fp,"\n");
    fclose(fp);
}


void FileRead(void *Var,int size,char *filename)
{
    FILE *data;

        /* Open binary file for output */
        if ((data = fopen(filename, "rb")) ==NULL) {
            printf("cannot open file\n");
            exit(1);
        }
        /* Read data from binary file into Var */
        if(fread(Var, size, 1, data)!=1){
            printf("read error");
            exit(1);
        }
        fclose(data);
}

void FileWrite(void *Var,int size,char *filename)
{
    FILE    *data;

        /* Open binary file for intput */
        if ((data = fopen(filename, "wb")) ==NULL) {
            printf("cannot open file\n");
            exit(1);
        }

        /* Write Var into binary file */
        if(fwrite(Var, size, 1, data)!=1){
            printf("write error");
            exit(1);
        }

        fclose(data);

}

void fPrintName(char *filename,char* option,int step)
{
    FILE    *fp;

    /* open file for input */
    if ((fp = fopen("tempname", "w")) ==NULL) {
        printf("cannot open file");
        exit(1);
    }

    fprintf(fp,"%s%s%d\n",filename,option,step);
```

```
    fclose(fp);
}

char *fReadName(char *input)
{
    FILE    *fp;
    char    *filename;

    fp = NULL;
    filename = NULL;

    /* open file for input */
    if ((fp = fopen(input, "r")) ==NULL) {
        printf("cannot open file");
        exit(1);
    }

    fscanf(fp,"%s\n",filename);

    fclose(fp);
    return filename;
}

void saveModel(int nXsect,int fXsect,int numfibers,int inNode,int refnode,float avgD,
        float stdevD,float avgVf,float stdevVf,float Va,float baseGap,
        float Lscale,float avgGap,float stdevGap,nodeptr node,faceptr face,
        char *paramfile,char *nodefile,char *nbrfile,char *facefile,
        char *porefile,sourceptr XsectVf,int cellnum,float Kz,float Kr,float Apore)
{
    int         i;
    float       z,maxV;
    nodeptr     nptr;
    neighborptr Nbr;
    faceptr     fptr;
    sourceptr   sptr;
    FILE        *fp[5];
    char        model[5][40];

    strcpy(model[0],paramfile);
    strcpy(model[1],nodefile);
    strcpy(model[2],nbrfile);
    strcpy(model[3],facefile);
    strcpy(model[4],porefile);

    for(i=0;i<5;i++) fp[i] = NULL;

    nptr = NULL;
    Nbr = NULL;
    fptr = NULL;
    sptr = NULL;

    maxV = 0;

    /* Open text files for input */
    for(i=0;i<5;i++){
     if((fp[i] = fopen(model[i], "w")) == NULL) {
       printf("cannot open file\n");
       exit(1);
```

```
    }
  }

  /* Write info on numfibers,nXsect,fXsect,inNode,refNode,avgD,stdevD,XsectVf,
     avgGap and stdevGap into ".param" file */
  fprintf(fp[0],"%d\t%d\t%d\t%d\t%d\n",numfibers,nXsect,fXsect,inNode,refnode);
  fprintf(fp[0],"%e\t%e\t%e\t%e\t%e\t%e\n",avgD,stdevD,avgVf,stdevVf,avgGap,stdevGap);

  fprintf(fp[0],"%e\t%e\t%e\t%d\n",Va,baseGap,Lscale,cellnum);
  fprintf(fp[0],"%e\t%e\t%e\n",Kz,Kr,Apore);
  sptr = XsectVf->next;      /* Skip header */
  for(;sptr!=NULL;sptr=sptr->next)
   fprintf(fp[0],"%d\t%e\n",sptr->i,sptr->Q);

  /* Write node data into ".node" file */
  nptr = node->next;                          /* Skip header */
  for(;nptr!=NULL;nptr=nptr->next){
   fprintf(fp[1],"%d\t",nptr->node);
   for(i=0;i<3;i++) fprintf(fp[1],"%e\t",nptr->x[i]);
   fprintf(fp[1],"%e\t%e\t%e\n",nptr->V,nptr->Vf,nptr->Rh);
   if(nptr->V>maxV) maxV = nptr->V;

   /* Write neighbor data into ".nbr" file */
   Nbr = nptr->neighbor->next;            /* Skip header */
   fprintf(fp[2],"%d\t%d\n",nptr->node,-1);
   for(;Nbr!=NULL;Nbr=Nbr->next) fprintf(fp[2],"%d\t%d\n",Nbr->f,Nbr->n);


  }

  /* Write face data into ".face" file */
  fptr = face->next;                          /* Skip header */
  for(;fptr!=NULL;fptr=fptr->next){
   fprintf(fp[3],"%d\t%d\t%d\t%e\t%e\t%e\t%e\n",fptr->F,fptr->u,fptr->v,
        fptr->A,fptr->L,fptr->H,fptr->Rcap);
  }


  /* Write data on pore space volumes into "model.pore" file */
   z = 0;
/*  fprintf(fp[4],"%e\t",z);*/
   nptr = node->next;                                 /* Skip header */
  for(;nptr->node!=inNode;nptr=nptr->next){
   if(nptr->x[2]==z) fprintf(fp[4],"%e\t",nptr->V/maxV);
   else{
     z += 1;  /* This assumes dz = 1 */
  /*  fprintf(fp[4],"\n%e\t%e\t",z,nptr->V/maxV);*/
    fprintf(fp[4],"\n%e\t",nptr->V/maxV);
   }
  }

  for(i=0;i<5;i++) fclose(fp[i]);

}

void readModel(int *nXsect,int *fXsect,int *numfibers,int *inNode,int *refnode,float *avgD,
        float *stdevD,float *avgVf,float *stdevVf,float *Va,float *baseGap,
        float *Lscale,float *avgGap,float *stdevGap,nodeptr node,faceptr face,
```

```
        char *paramfile,char *nodefile,char *nbrfile,char *facefile,sourceptr XsectVf,
        int *cellnum,float *Kz,float *Kr,float *Apore)
{
    int       i,F,u,v,f,n;
    float     x[3],A,L,H,Rcap,Vf;
    nodeptr   nptr;
    neighborptr Nbr;
    faceptr   fptr;
    sourceptr  sptr;
    FILE      *fp[4];
    char      model[4][40];

    strcpy(model[0],paramfile);
    strcpy(model[1],nodefile);
    strcpy(model[2],nbrfile);
    strcpy(model[3],facefile);

    for(i=0;i<4;i++) fp[i] = NULL;

    nptr = NULL;
    Nbr = NULL;
    fptr = NULL;
    sptr = NULL;

    F = u = v = f = n = 0;
    for(i=0;i<3;i++) x[i] = 0;
    A = L = H = Rcap = 0;

    /* Open text files for output */
    for(i=0;i<4;i++){
     if ((fp[i] = fopen(model[i], "r")) ==NULL) {
       printf("cannot open file\n");
       exit(1);
     }
    }


    /* Read data on numfibers,nXsect,fXsect,inNode,refNode,XsectVf from ".param" file */
    fscanf(fp[0],"%d\t%d\t%d\t%d\t%d\n",numfibers,nXsect,fXsect,inNode,refnode);
    fscanf(fp[0],"%e\t%e\t%e\t%e\t%e\n",avgD,stdevD,avgVf,stdevVf,avgGap,stdevGap);
    fscanf(fp[0],"%e\t%e\t%e\t%d\n",Va,baseGap,Lscale,cellnum);
    fscanf(fp[0],"%e\t%e\t%e\n",Kz,Kr,Apore);
    while(!feof(fp[0])){
     fscanf(fp[0],"%d\t%e\n",&i,&Vf);
     addsource(XsectVf,i,Vf);
    }

    /* Read node data from ".node" file */
    while(!feof(fp[1])){
     fscanf(fp[1],"%d\t",&n);
     addNodeData(node,n);
     nptr = getNode(n,node);
     for(i=0;i<3;i++) fscanf(fp[1],"%e\t",&nptr->x[i]);
     fscanf(fp[1],"%e\t%e\t%e\n",&nptr->V,&nptr->Vf,&nptr->Rh);
    }


    /* Read neighbor data from ".nbr" file */
```

```
    while(!feof(fp[2])){
     fscanf(fp[2],"%d\t%d\n",&f,&n);
     if(n<0){
       nptr = getNode(f,node);
       Nbr = nptr->neighbor;
     }
     else addNeighbor(Nbr,f,n);
    }

    /* Read face data from ".face" file */
    while(!feof(fp[3])){
     fscanf(fp[3],"%d\t%d\t%d\t%e\t%e\t%e\t%e\n",&F,&u,&v,&A,&L,&H,&Rcap);
     addFace(face,F,u,v);
     fptr = getFace(F,face);
     fptr->A = A;
     fptr->L = L;
     fptr->H = H;
     fptr->Rcap = Rcap;
    }


    for(i=0;i<4;i++) fclose(fp[i]);

}


void mystrcat(char *str1,char *str2,char *newstr)
{
    char *p1,*p2;

    p1 = p2 = NULL;
    p1 = str1;
    while(p1 <= str1 + strlen(str1) - 1)
        *newstr++ = *p1++;
    p2 = str2;
    while(p2 <= str2 + strlen(str2))
        *newstr++ = *p2++;

}
```

APPENDIX D Program for Image-Based Fingering Analysis

The procedure for analyzing the data on fiber-scale fingering is outlined. The algorithm used for quantitative analysis of images acquired from impregnation experiments on the model tows is described in Appendix D.1. An ANSI standard C program incorporating the algorithm is listed in Appendix D.2.

D.1 Description of Algorithm

Images of fiber-scale fingering were acquired from videotape using a frame grabber (Perceptics PixelGrabber™) and stored on a Macintosh IIfx computer. The images were analyzed using the public domain software package *Image*, available from the National Institutes of Health. Figure D.1 shows the nomenclature used for analysis of the acquired images.

As indicated in Figure D.1, the datum for measurement of finger length, z_f, is the line defined by the points P_1 and P_2, the points of apparent intersection of the fully saturated region behind the flow front and the walls of the test tube (i.e., mold wall). The general equation for the line $\overline{P_1P_2}$ is

$$(y_2 - y_1)x + (x_1 - x_2)y + (x_2 - x_1)(y_1 - mx_1) = 0 \qquad \text{(D.1)}$$

where

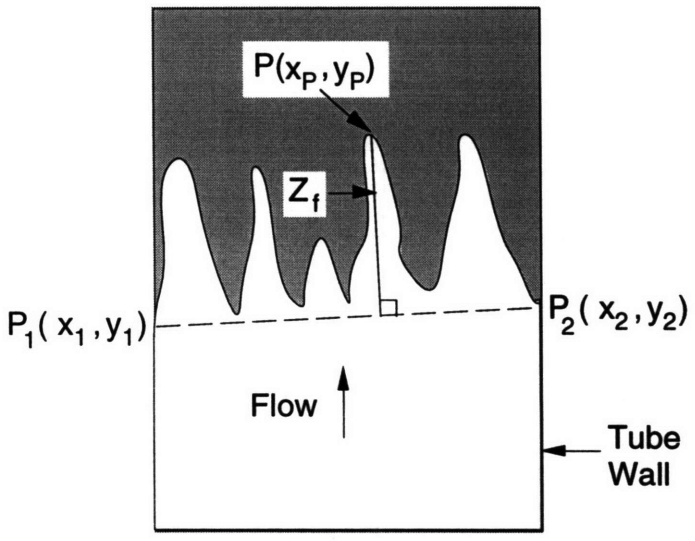


Figure D.1 Nomenclature for analysis of fiber-scale fingering.

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{D.2}$$

is the slope of line $\overline{P_1P_2}$. The finger length, z_f, corresponds to the perpendicular distance from the finger tip, point $P(x_P, y_P)$, to the line $\overline{P_1P_2}$ and is given by

$$z_f = \frac{Ax_P + By_P + C}{\left|\sqrt{A^2 + B^2}\right|} \tag{D.3}$$

where

$$\begin{aligned} A &= y_2 - y_1 \\ B &= x_1 - x_2 \\ C &= (x_2 - x_1)(y_1 - mx_1) \end{aligned} \tag{D.4}$$

and m is defined as in Eq. D.2.

As shown in Eqs D.1-D.4, the input data required for analysis of the fiber-scale fingers consist of the (x, y) coordinates of points P_1 and P_2 and the coordinates of P for each finger observed in the image. Note that the line $\overline{P_1P_2}$ truly represents the bulk wetting front since any dry spots in the interior of the sample are clearly visible.

D.2 Program Listing

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "stddef.h"
#include "math.h"

typedef struct FingerInfo *fngrptr;

struct FingerInfo{
    float    x[3];
    fngrptr  next;
};

fngrptr newFinger(float *x);
void addFinger(fngrptr finger,float *x);
void printFinger(fngrptr finger);

void main()
{
    int      i,numfingers;
    float    x[3],x1[2],x2[2],A,B,C,D,avg,stdev,sum,f2,diam;
    FILE     *fp;
```

```
char      *ans,*input,*output;
fngrptr   finger;

diam = 0.4572; /* 0.018 in = 0.4572 mm */;
fp =  NULL;
input = output = ans = NULL;
ans = "y";
A = B = C = D = 0;

i = 0;
do{
     numfingers = 0;
     A = B = C = D = 0;
     printf("Enter name of input file: ");
     scanf("%s",input);
     printf("%s\n",input);

     /* Open text file (with line breaks) to read data */
     if ((fp = fopen(input, "r")) ==NULL) {
          printf("cannot open file\n");
          exit(1);
     }

     for(i=0;i<2;i++) x1[i] = x2[i] = 0;
     for(i=0;i<3;i++) x[i] = 0;

     fscanf(fp,"%f\t%f\n",&x1[0],&x1[1]);
     fscanf(fp,"%f\t%f\n",&x2[0],&x2[1]);

     /* Normalize measurements to nominal fiber diameter */
     for(i=0;i<2;i++){
          x1[i] /= diam;
          x2[i] /= diam;
     }

     /* Calculate constants for gen'l eqn for base line, Ax+By+C=0 */
     A = x2[1]-x1[1];
     B = x1[0]-x2[0];
     C = -(A*x1[0]+B*x1[1]);

     /* Calculate denominator, D, in eqn for dist of a point to a line */
     D = sqrt(A*A + B*B);
     /* Check for division by zero. If ok, D = 1/D. A*(1/B) faster than A/B */
     if(D) D = 1/D;
     else{
          printf("Division by zero\n");
          exit(1);
     }

     printf("A = %f\tB = %f\tC = %f\tD = %f\n",A,B,C,D);

     finger = newFinger(x);


     sum = avg = stdev = f2 = 0;
     /* Read data from Image measurement file */
     while(!feof(fp)){
          for(i=0;i<3;i++) x[i] = 0;
```

```
            fscanf(fp,"%f\t%f\n",&x[0],&x[1]);

            /* Normalize measurements to nominal fiber diameter */
            for(i=0;i<2;i++) x[i] /= diam;

            x[2] = fabs((A*x[0]+B*x[1]+C)*D);
            sum += x[2];
            f2 += x[2]*x[2];
            addFinger(finger,x);
            numfingers++;
        }
        fclose(fp);

        printFinger(finger);

        avg = sum/numfingers;
        stdev = (f2 - sum*sum/(numfingers))/(numfingers);
        stdev = sqrt(stdev);
        printf("Average Finger Length = %.3f\t std dev = %.3f\n",avg,stdev);



        printf("Would you like to analyze another file (y or n)? ");
        scanf("%s",ans);

    } while (*ans=='y');

}


fngrptr newFinger(float *x)
{
    int     i;
    fngrptr new;

    if((new = (fngrptr) malloc(sizeof(struct FingerInfo))) != NULL) {
        for(i=0;i<3;i++) new->x[i] = x[i];
        new->next = NULL;
    }
    return new;
}

void addFinger(fngrptr finger,float *x)
{
    int         k; /* control variable to stop for-loop after 1 iteration */
    fngrptr temp;

    temp = finger;
    k = 0;
    for( ;temp != NULL;temp = temp->next){
        if((temp->next == NULL) && (k==0)){
            temp->next = newFinger(x);
            k++;
        }
    }
}

void printFinger(fngrptr finger)
```

```
{

    int       i;
    fngrptr   temp;

    temp = finger->next;        /* Skip header */
    for(;temp !=NULL;temp = temp->next){
        for(i=0;i<3;i++) printf("%.2f,",temp->x[i]);
        printf("\n");
    }
}
```