12-2000

Creating Virtual Wood Particulate Composites

Huaijun Wang

CREATING VIRTUAL WOOD PARTICULATE COMPOSITES

BY

Huaijun Wang

B.S. Nanjing Forestry University, China 1986

M.S. University of Maine, 1996

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

(in Forest Resources)

The Graduate School

The University of Maine

December, 2000

Advisory Committee:

Stephen Shaler, Professor of Wood Science and Technology, Advisor
James Fastook, Associate Professor of Computer Science
Douglas Bousfield, Professor of Chemical Engineering and Pulp & Paper
Vincent Caccese, Associate Professor of Mechanical Engineering
Thomas Brann, Professor of Forest Resources and Forest Engineering

CREATING VIRTUAL WOOD PARTICULATE COMPOSITES

By Huaijun Wang

Thesis Advisor: Dr. Stephen M. Shaler

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
(in Forest Resources)
December, 2000

Wood particulate composites are inhomogeneous, short fiber materials. The behavior of such materials depends on the properties of the constituent elements (wood particles, fibers, and adhesives) and the manner in which they are organized (the microstructure). Research on the microstructure of these composites has been carried out for several decades. However, due to the complexity of the engineered structures, most existing works are either over-simplified or lacking in general representations. While experimental approaches to improve material performance are necessary, this thesis presents another approach for better depictions of such complex systems. In this research project, digital surrogates of the composite structures are created and tested in a virtual laboratory - the computer. Each *structure* is constructed by packing digital particulate elements. The electronic versions of wood fibers, strands or flakes are generated based on the known properties of their real-world counterparts, including stochastic descriptions of their geometries, and some physical and mechanical properties. More importantly, a certain level of intelligence built-in to the *elements* enables them to adjust their geometries on the basis of the interaction with other elements. As it is based on the very fundamental rules, this simula-

tion methodology guarantees much more realistic descriptions of the microstructures compared with other models. This method allows controlled and repeatable experiments with virtually zero experimental and material costs. Since the *structures* of the virtual composites are digitized, it is fairly easy to obtain the information which may be impossible or hard to obtain from the real composites, such as the morphology of particle-to-particle bonds and the quantities of inter/intra-particle voids. It lays a more reliable foundation for further research on performance prediction and manufacturing optimization of the wood particulate composites.

ACKNOWLEDGMENTS

I gratefully appreciate my graduate advisor, Professor Stephen Shaler, for letting me work on this fascinating project, for his inspiration and guidance throughout this work, and for his trust and understanding over the past 70 months.

I am very grateful to the other members of my advisory committee: Dr. Douglas Bousfield, Dr. Vincent Caccese, Dr. Thomas Brann, and particularly, Dr. James Fastook who ushered me into the wonderful world of computer simulation, for their constructive advice and valuable support.

I would also like to express my thanks to some of my friends and colleagues: George Hua, Dr. Yi Zhang, Dr. Beihong Liang, Dr. Yanruo Han, Gang Xu, Liren Chen, Dr. Laurence Mott, Dr. John Guerin, Dr. Jin Liu, Kim Adler, Christopher Paduan, Rizuan Ramli, Ethan Davis, Kamala Raman, Matthew Brown, Avril Egan, Eoin Battles, Chunyan Wang, Yuhui Qian, Katharina Prall and Jonathan Alexander, as well as Professor Fenghu Wang and his wife, Tiemei Chen, who either helped me starting a new life in the US or made my life easier in different ways.

I am greatly indebted to my parents, sister, brother and my parents in law. Their continuous support and encouragement made the completion of this research possible.

Finally, I would like to thank my wife and best friend, Ping, for her love, patience and sacrifice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1. Background

Wood based particulate composites, such as medium density fiber board (MDF), flake board and oriented strand board (OSB), are manufactured with wood components (e.g. flakes, particles or fibers) partly coated with resin. The wood components are dispersed randomly or partially oriented on a plane, packed into a mat, then compressed into a densified board. The particles are bonded together into a three dimensional web by the added adhesive. Paper products can also be viewed as a particulate composite, with the connection between fibers primarily consisting of hydrogen bonds with starches or other materials, at times used as a binder. Particulate composites cannot be analyzed using the rules developed for continuous or discontinuous particulate composites in which the particles or fibers are distributed in a continuous-phase matrix for the purpose of mechanical reinforcement (Chou, 1992). Voids and bonds are two features of high importance for wood particulate composites. The performance of wood composite products is determined by many factors which can be classified into three types:

- *material factors,* including properties of wood and adhesive,
- *structural factors,* including the geometry of the composite elements and their organization in the composite, and

- *manufacturingfactors* such as press time, temperature and pressure applied during the pressing. All these factors interact to determine the final properties of the composites.

Bonds, either adhesive or hydrogen, serve to transfer load between the constituent elements. Stress can be caused in a composite by moisture, as well as applied loads. Due to the hygroscopic nature of wood, changes in environmental temperature and humidity affect the equilibrium moisture content of the wood elements. The water absorbed or desorbed in response to the changes causes swelling or shrinkage of the wood elements, which in turn puts stresses on the bonds. The quality and quantity of the bonds are believed to strongly depend on the properties and organization of the wood elements (Alexander and Mat-ton, 1968a, 1968b, Leopold and Thorpe, 1968, Kallmes and Corte, 1960, Laufenburg, 1984).

Voids serve as substance transportation channels in the composites. The morphological characteristics of the void structure, which also depend on the geometry and organization of the wood elements, are important for understanding some physical properties of the composites, such as the permeability and optical properties.

Wood is an anisotropic material with a large variance in its mechanical and physical properties. Wood particles and fibers for making wood composites inherit these special characters, in conjunction with the irregular geometry due to the natural and manufacturing conditions. The microstructure formed by randomly organizing these particles or fibers is definitely inhomogeneous and analytically indescribable. Several models have been established theoretically or experimentally to describe these kinds of systems (Kallmes, etc.

196 1, Dai and Steiner, 1994a, 1994b, Shaler, et al. 1996). The preponderance of models in the literature treat the microstructure as two-dimensional (2D). Such models typically describe the materials as layer-packed assemblies. In other words, the elements in panels or paper are assumed parallel to the horizontal plane and the thickness or diameter of the elements is zero. Some three-dimensional (3D) models have been proposed (Niskanen and Alava, 1994, Wang and Shaler, 1998, Stahl and Cramer, 1998, Nilsen, etc. 1998). These models are either imbued with unrealistic features or are limited to the structures with relatively small vertical dimensions or fairly low density structures. In a mat of wood composite, the voids and vertical (a planar perpendicular to the predominant plane of the products) orientation angle of the elements may be reduced by the densification and the deformation or damage of the elements during the consolidation, but some will always remain. In addition, there is some spring-back after the consolidation because of the viscoelasticity of wood materials. Obviously, the geometric complexity of the structure is far beyond the reach of two-dimensional models. The 3D orientation of each element affects 3D panel performance due to the inherent anisotropy properties of wood such as mechanical response and hygroscopic expansion and contraction. A comprehensive and effective prediction of mechanical and physical behaviors of the wood composites can only be accomplished through a 3D microstructural model.

The limitations of existing models was historically due in part to insufficient computational power to deal with such complex systems. The development of high-speed processors and large-capacity storage has increased the complexity of solvable problems.

Consequently, it makes it possible to build a more sophisticated microstructural model of the wood particulate composites.

1.2. Objectives

The specific objectives of this research were to:

- Construct 3D microstructural models which can reasonably represent the real wood particulate composites,
- Characterize some important microstructural phenomena based on properties of composite elements and their organizations.

Original computer software was developed in the current research. The simulation procedure was based on the methodology by which a surrogate world is created to explain a high-level phenomenon by appealing to interactions among lower-level agents (Casti, 1997). The program simulated the emergent, the microstructure of wood particulate composites by describing the behavior of each agent, the wood fiber or particle, and the interactions among them in the packing and consolidation processes. With the software, virtual composite structures can be produced according to stochastic characterization of elements's geometry, as well as some physical, mechanical properties and manufacturing conditions.

The simulation process may not only supply reasonable references or partial substitute to experiment procedures for the research of composite properties, but also help researchers to obtain certain information that is difficult or impossible to obtain through analytical or experimental methods, such as the bonding area and voids.

2 LITERATURE REVIEW

2.1. Microstructure of Wood Particulate Composites

2.1.1. Properties of Elements and Effects

Wood fiber based composites consist predominantly of ligno-cellulose fibers, with the addition of up to 10% adhesive by weight. Fiber characteristics are variable due to biological factors including species, geographic conditions and location within a tree as well as processing procedures such as chemical pulping, thermal mechanical pulping, etc. Other wood particulate composites are manufactured from different types of wood particles. The particle size and shape can range from large wood chips or flakes down to fines such as sander dust. Particle geometry is one of the basic factors governing the properties and characteristics of the composite panels along with wood species, type and amount of binder and other additives, and manufacturing conditions, e.g. mat forming and pressing conditions (Malony, 1977). Malony (1977) presented a detailed summary after reviewing the research on the effects of particle geometry. All the studies (Talbott and Maloney, 1957, Maloney, 1958, Post, 1958; Heebink and Hann, 1959, Klauditz and Buro, 1962, Gatchell, Heebink and Hefty, 1966, and Mottet, 1967) reviewed related the particle geometries to the mechanical and physical performances of the wood composite panels instead of the microstructure.

2.1.1.1. Length

Wood fibers are essentially cylindrical with lengths from a few millimeters down to fibrous debris or tines and the diameters ranging from several to 80 microns (Panshin and Dezeeuw, 1970, Deng and Dodson, 1994). Fiber length distributions typically skew towards longer fiber lengths and can be modeled by lognormal distributions (Dodson, 1992, Deng and Dodson, 1994). In addition, the variance of the length distribution decreases as the mean fiber length is reduced by refining. Dodson (1992) derived a paper sheet model which indicated that the coefficient of variation of mass density of a paper sheet made from fibers with a lognormal distributed length was a few percent lower than that of a sheet made from fibers with uniform length.

Seth (1995) tested wet webs made of 30% solids from softwood kraft pulps and found that the tensile strength was proportional to the fiber length. Karenlampi et al. (1996) proposed that the in-plane tearing work, which was modeled as the total energy dissipated during the test, was a linear function of fiber length and it increased while fiber length increased.

2.1.1.2. Cross-Sectional Geometries

Descriptions of the cross-sectional geometries depend on the types of the particulates. For flake type composites, the cross section of the elements can be easily described as rectangles with certain widths and thicknesses. For fibrous composites, the cross section of fibers can be viewed as circles with certain diameters and cell wall thicknesses. As may be seen later, usually widths and thicknesses are used instead to describe cross-sectional deformed or flatterned fibers.

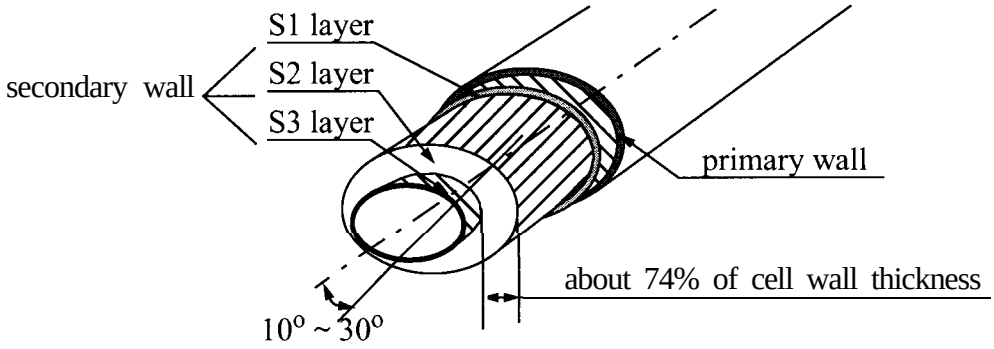


Figure 2- 1. Schematic of structure of woody cell walls.

The physical and mechanical properties of wood fibers are dominated by the properties and organization of the microfibrils in the S2 layer of the secondary-wall. This is due to the fact that the S2 layer constitutes about 74% of the total volume of the cell wall and the microfibrils are nearly parallelly aligned, with about 10 to 30 degree orientation angle from the cell axis (Panshin and Dezeeuw, 1970). The diameters of fibers in wood have distributions that are approximately normal, with the standard deviations in the region of 10% of the mean (Deng and Dodson, 1994).

Several articles have been written about the impact of paper forming process on the cell wall structure (Alexander and Mar-ton 1968 a, 1968 b, Leopold and Thorpe, 1968 and Kim, et al., 1975). Drying stress may strengthen fibers both by reducing the fibril angle and by aligning the fibrils in the dislocated regions which are formed during defibering, refining and other mechanical treatments. However, this may only happen if the matrix between the fibrils can flow in shear under the applied stress. Thus, fibers with high hemicellulose content, the matrix of which flows readily in the water swollen state, may be strengthened during the drying. For predried kraft fibers or alkali-extracted holocellulose, strengthening may not occur since the matrix is well bonded and insufficiently swollen (Kim et al., 1975).

Although it is generally believed that the pycnometric value of cell-wall density of wood fiber is about 1.5 g/cm^3 (Panshin and deZeeuw, 1970), Yiannos' (1964) research showed that it was about unity (Table 2-l) because of the voids in the cell wall.

Table 2-l. Coarseness and cell-wall density of pulp fibers (Yiannos, 1964).

Pulp	Coarseness (mg/100m)	Cross-sectional area (μm^2)	Cell-wall density (g/cm^2)
Bleached southern softwood kraft	30.1	286	1.06
Bleached West Coast softwood sulfite	24.1	230	1.05
Bleached northeastern softwood sulfite	16.8	175	0.96
Unbleached special Parana pine pulp	46.3	425	1.08
Unbleached spruce kraft pulp	18.4	210	0.88
Unbleached special spruce bisulfite pulp	20.5	230	0.88

Fiber coarseness, which is defined as the weight per unit length, has been used to describe fiber properties. Since it is assumed that the density of cell wall substance is a constant, the coarseness must be dependent on the cell wall structure. Seth (1995) found that fiber coarseness for softwood was nearly proportional to fiber cell wall thickness when the fiber width did not change. It was also reported that the tensile strength of wet webs, made of 30% solids from softwood kraft pulps for fibers of a given length, decreased rapidly with an increase in fiber coarseness. The wet-web tensile strength at a given fiber length was almost inversely proportional to the square of fiber coarseness (Seth, 1995).

2.1.1.3. Flexibility

Since many properties of paper are functions of fiber-to-fiber bonded area and the bonded area is a function of the fiber flexibility (or conformability, deformability), flexibility is

generally accepted as one of the most important controlling factors of paper properties (Lossada, 1998).

According to Tam Doo and Kerekes (1982), the fiber flexibility, F, is defined as

$$F = \frac{1}{S} = \frac{1}{I\text{-}E} \tag{2-1}$$

where S, I and E are the stiffness, moment of inertia and Young's modulus in the axial direction of the fiber, respectively. The moment of inertia is a consequence of the fiber's cross-sectional dimensions. The Young's modulus is a strong function of the ultra-structural organization or the microfibrillar orientation of the fiber (Page et al., 1977 and Da Silva, 1983). Page et al. (1977) measured microfibril angle and found that the individual wood fibers were strongest when their microfibril angle in the S2 layer was closest to zero. The ultimate tensile stress and related moduli decreased with increasing microfibril angle. The elastic modulus versus fibril angle relation is shown in Figure 2-2.

Paavilainen (1993) further reiterated that the moment of inertia depended on cell wall thickness and width of the fiber. The cell wall elasticity was determined by the chemical composition, fibril angle and size of fibrils on a fiber's S2 layer. Paavilainen's research showed that fiber flexibility was affected by manufacturing procedures. Pulping, bleaching, beating and drying all influenced the flexibility by removing and modifying lignin as well as by changing the cell wall microstructure and cross-sectional dimensions. Increased levels of pulping, bleaching and beating would increase flexibility while drying process decreased fiber flexibility (Paavilainen, 1993).

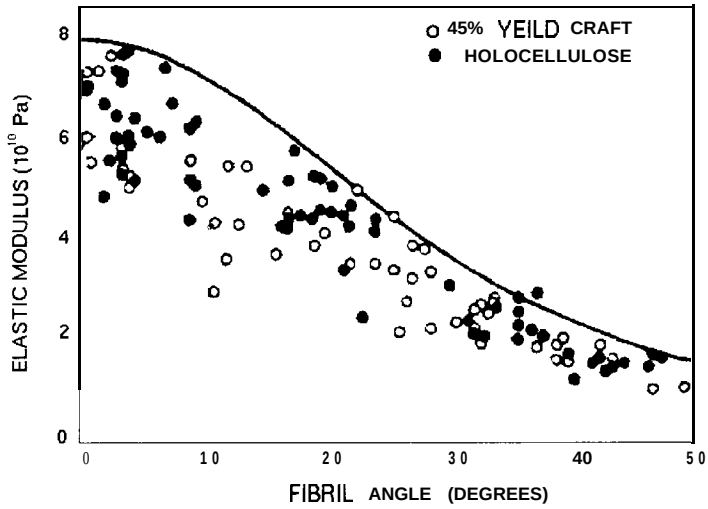


Figure 2-2. Variation in elastic moduli with fibril angle for spruce holocellulose fibers (Page, et al. 1977).

Paavilainen (1993) found that the tensile strength and apparent sheet density of paper made of softwood fibers was controlled by wet fiber flexibility since the paper sheet became more consolidated and thus the number and area of fiber-to-fiber bonds increased if fibers were more flexible. It was found that the tensile strength and apparent sheet density increased linearly as fiber flexibility increased.

There are two ways of measuring wet fiber flexibility. One was developed by Steadman and Young (1978). The other was developed by Tam Doo and Kerekes (1982). Both methods involved beam theories. Steadman's method uses a cantilever beam configuration while in Tam Doo and Kerekes' method two ends of the tested fiber are simply supported.

Tam Doo and Kerekes also found that chemical pulps were typically 20 to 30 times more flexible than mechanical pulps from the same wood species. Decreasing the yield of a

chemical pulp from 85 to 50% led to a fourfold increase in fiber flexibility. Different wood species pulped under common conditions differed greatly in flexibility. Additionally, they found a correlation between fiber stiffness and breaking length for some British Columbian pulps, (e.g., the breaking length decreased with increasing fiber stiffness.) (Tam Doo and Kerekes, 1980).

2.1.1.4. Collapsibility

Fiber collapsibility depends on the transverse elasticity of the cell wall (Page, 1967 and Hartler and Nyren, 1970) and the cross-sectional dimensional properties (Paavilainen, 1993). In Paavilainen's experiment, the degree of collapse was determined as a function of the aspect ratio which was the ratio of major to minor axes of the fiber cross-section. To a certain extent the flexibility and collapsibility of fibers are related to each other because both of them depend upon the elasticity of the cell wall and the cross-sectional properties (Paavilainen, 1993). Paavilainen found similar effects of the forming process on the fiber collapsibility as on flexibility. In addition, increased fiber collapse was correlated with improved paper sheet properties.

Mataki (1972) discussed the cross-sectional deformation of fibers in fiberboard and the relations to the fiber properties and load. The wood fibers were considered as tubular cylinders which were free from lateral restraint. The increase in diameter of the fiber, $2\Delta x$, due to the forming pressure in the direction perpendicular to the pressure and the major axis of the fiber (x-direction) was given by:

$$2\Delta x = \frac{0.137P}{2bE\sum_{n=1}^{\infty}\frac{1}{\partial n+1}\left(\frac{e}{r}\right)^{2n+1}} \qquad (2\text{-}2)$$

in which P is the compressive load, E is the modulus of elasticity of the fiber cell wall, b is the width of the fiber in the x direction, e is the fiber wall thickness, and r is the fiber radius to the center of fiber wall. The ratio of fiber cross-sectional dimensions, which was the dimension of the fiber in the x-direction divided by the dimension in the direction of pressure (z-direction), was used to describe the cross-sectional deformation of fibers (Mataki, 1972).

2.1.1.5. Curl

The curl of fiber was determined by Kallmes and Corte (1960) as the ratio between the measurement of the actual fiber length and the distance between the two fiber ends. Jang-malm and Ostlund (1995) investigated effect of fiber curl on the mechanical properties of paper. In their research, the curl factor was defined as the ratio of the distance between the fiber ends and the total length of the fiber which is the reciprocal of the value defined by Kallmes and Corte (1960). Experimental results of their research showed that the tensile stiffness index increased linearly with the increase of curl factor.

It was claimed that the wet interlaminar tensile strength of cement-silica sheet was improved by fiber curl, which was deliberately induced in unbleached P. radiata kraft pulp fibers by mechanical treatment. However it had little effect on the values modulus of rupture and fracture toughness of the sheet (Michell and Freischmidt, 1990).

2.1.1.6. Orientation

Harris (1977) set up a model to predict the mechanical properties of oriented particle boards. In the model, both the von Mises distribution function and a truncated gaussian-type distribution function were employed to describe the angular data of particle alignment. The appropriateness of these two functions relies on their finite range which fits the property of angular data.

The von Mises distribution was also used by Shaler (199 1) to characterize the orientation of flake layers on a plane. The probability density function of flake angle, θ $(0 \leq \theta \leq \pi)$, was expressed as:

$$f(\theta) = \frac{e^{k\cos 2(\theta-\mu)}}{2\pi I_0(k)} \qquad (2\text{-}3)$$

where μ is the mean orientation angle of flakes, k is the concentration parameter, and $I,(k)$ is the modified Bessel function of the first kind and order zero.

Dodson and Fekih (199 1) developed a model of paper structure which predicted that the variance of mass density of paper would linearly increase with the increase of 'eccentricity' of fiber axis orientation distribution. It was further indicated that the variance increased 30% as the orientation increased from 1: 1 to 2: 1. Correspondingly, the coefficient of variation of mass density increased about 14%. A one-parameter angular intensity distribution function was used to characterize the orientation of fiber axes

$$d\mu(\theta) = \mu\left(\frac{1}{\pi} + e\cos 2\theta\right)d\theta \qquad (2\text{-}4)$$

where *e* is the *eccentricity* of orientation ($0 \le e < 1/\pi$). When $e = 0$, fibers have no orientation preference. When $e \to 1/\pi$, fraction of fibers oriented in direction $\theta = \pm\pi/2$ tends to zero (Dodson and Fekih, 199 1).

Lu and Carlsson (1996), in their simulation of formation and microstructure of paper, used warped Cauchy distribution to define the orientation of fibers:

$$f(\theta) = \frac{1}{\pi}[(1-\lambda)/(1+\lambda^2 - 2\lambda\cos 2\theta)] \qquad (2\text{-}5)$$

The simulation result showed that, at a given density, the orientation and dimensions of fibers had little effect on the volume fractions of bonds and fiber segments.

2.1.1.7. Relative Bonded Area

Bonding area, a special feature of the microstructure rather than a property of the composite elements, is the most important factor affecting the performance of paper and other wood fiber and particle composites since bonds serve to transferring load between composite elements. More bonding area at certain density level could mean stronger structure and better mechanical performance.

Relative bonded area (RBA) is the fraction of the external fibrous surface area bonded (Kallmes and Eckert, 1964). RBA is very difficult to measure experimentally. Kallmes and

Eckert (1964) discussed two major techniques for indirect measurement of RBA, i.e. nitrogen adsorption and light scattering. The two techniques are based on two characteristics unique to the unbonded surface of fibers, its ability to adsorb gas and scatter light. Sophisticated instruments are needed to perform the measurement. Since both methods can not take into account the contribution of the uncollapsed part of lumens, generally low RBA values are measured.

2.1.2. Simulation and Modeling of Paper Structures

Significant research on the structure of wood composite materials started as early as the 1950's with the description of the structure of paper. Before that, some effort had been made into the calculation of the number of fiber-to-fiber contacts and the bonded area due to the structural importance of the bonds. Van Wyk, in 1946, derived a model to calculate the average distance between contact points in a random assembly of uniform cylindrical fibers (Van Wyk, 1946). His formula is:

$$b = \frac{2v}{\pi l d} \tag{2-6}$$

where b is the average distance between contact points, I is the total length of the fibers contained in a assembly with volume v, and d is the diameter of fibers.

During the more than five decades following this work, the structure of paper has been studied extensively. Kallmes, Bernier and Corte (Kallmes et al., 1960, Kallmes, Corte and Bernier, 1961, Kallmes and Bernier, 1963) were the first to describe the fiber geometric network statistically. In their work, the paper structure was defined to be the geometric

arrangement of the fibers and inter-fiber spaces or pores. Because of its complexity, they approached the problem by considering paper to be made of a pile of ultra-thin or 2D sheets of which the grammage was 2.5 g/m^2 and the geometric elements were plainly visible. This allowed the simplified ideal paper to be simulated with uniform straight line segments placed at a random location and a random planar angle. The structure of the 2D sheets was described in terms of the means and distributions of several geometric properties. These properties were thought to be closely related to the mechanical, optical, and porous properties of paper. The equations were derived from probability theory and can be evaluated in terms of the number of fibers in a sheet and their dimensions. Their overall modeling process considered a total of N_f fibers deposited randomly onto a plane of area, A. The Poisson distribution was used to characterize the number of fiber segments per unit area

$$p(r) = \frac{e^{-n_f} \cdot n_f^r}{r!} \tag{2-7}$$

in which r is the number of fiber segments per unit area, n_f is the mean number of fiber segments per unit area, which equals $\frac{N_f}{A}$ (Kallmes, and Corte, 1960).

The distribution of the free fiber length, l_f, was given by a negative exponential equation

$$p(l_f) = \mu_l e^{-\mu_l l_f} \tag{2-8}$$

in which μ_l is the mean value of the free fiber length.

Based on these two equations, several other distribution functions were derived, which include:

1. The distribution of the length of fibrous material per unit area of the sheet
2. The distribution of the number of crossings per unit area of the sheet
3. The distribution of the number of crossings per fiber
4. The distribution of the number of free fiber lengths per fiber
5. The distribution of the areas of polygons.

A model for a multi-planar sheet was also developed (Kallmes et al., 1961). In the model, many of the geometric properties of interest to the papermaker were derived from the total number of fiber-to-fiber contacts (crossings) in the sheet. The number of crossings in a multi-planar sheet was the sum of the number of crossings within the two dimensional (2D) layers and the number of crossings between the layers. It was claimed that although the organization of fibers in a multi-planar sheet differed from that in normally formed paper in several ways, i.e. no Z-direction orientation and no interweaving among the layers, such phenomena were of minor importance in paper and that there were only small differences between the properties of a multi-planar model and the equivalent sheets (Kallmes et al., 1961).

Page and Seth (1980) derived a comprehensive equation which related the in-plane elastic modulus of well-bonded paper sheet made of straight fibers to most of properties of the fibers

$$E_p = \frac{1}{3} E_f \left[\left(1 - \frac{w}{L \cdot RBA} \right) \sqrt{\frac{E_f}{2G_f}} \right] \tag{2-9}$$

where E_p is the in-plane elastic modulus of paper, E_f is the elastic modulus of fiber in axial direction, G_f is the shear modulus of fiber for deformation in the plane of fiber length and width, L is the length of fiber, w is the width of fiber, and RBA is the relative bonded area of fiber. It was implied from the equation that for long fibers of high values of RBA the elastic modulus of paper was one third of the elastic modulus of it component fibers.

Gorres et al. (1989) developed an interactive multi-planar model (IMPM) which extends the multi-planar model by Kallmes et al. (1961) to account for the interaction between layers due to the effect of fiber flexibility. The analytical expressions describing the number of crossings between two layers and fraction of bonded fibers between two layers were presented. Gorres and Luner (1992) later derived a model of paper density from the IMPM. The model used the mean values of fiber coarseness, width, thickness, length and flexibility as well as the tines content of pulps to predict sheet density. Prediction errors of up to 27% were reported. A combination model of IMPM and fiber orientation function was then developed and it predicted that apparent sheet density increased with an increase in fiber orientation (Amiri et al. 1994).

Komori and Makishima (1977) estimated the number of fiber-to-fiber contacts in fiber assemblies with arbitrary distributions of orientation and fiber length as:

$$\bar{n}_v = D L_v^2 I, \tag{2-10}$$

where ii, is the average number of contacts per unit volume of an assembly, D is the diameter of the cross section of the fiber, L_v is the total length of fiber per unit volume and I is calculated with

$$I = \int_0^{\pi} d\theta \int_0^{\pi} d\varphi \int_0^{\pi} d\theta' \int_0^{\pi} d\varphi' \sqrt{1 - [\cos\theta\cos\theta' + \sin\theta\sin\theta'\cos(\varphi - \varphi')]^2}\,\Omega(\theta,\varphi)\Omega(\theta',\varphi')\sin\theta\sin\theta'$$

in which Ω is the density function of orientation, θ, φ and θ', φ' are respective polar angles characterizing the orientation of two fibers being brought into contact with each other.

Van den Akker's method (Van den Akker, 1962) was for calculating total bonded area. All the methods were based on mathematical derivation according to the average dimension of fibers and the volume of fiber assemblies. Deng and Dodson presented detailed methodologies and reviews of paper structure research in their book (1994), most of which were related to 2D networks.

Dodson (199 1) described a 3D mathematical model characterizing the coefficient of variation of the local volume density. In his analytical derivation, paper was viewed as a 3D stochastic fibrous network. The network was subdivided into unit cells. The coefficient of variation of the local volume density was:

$$cv(\tilde{\gamma}) = \sqrt{(cv(\tilde{\beta}))^2 + \left(\frac{\delta\lambda}{x^2 t}\right)\cdot\frac{1}{\bar{\gamma}}} \qquad (2\text{-}1\,1)$$

where $\tilde{\gamma}$ is the local volume density, $\tilde{\beta}$ is the local basis weight, λ is the mean fiber length, δ is the mean fiber mass per unit length, x is the length and width of the unit cells, t is the thickness of the unit cells, and $\bar{\gamma}$ is the mean density of the network.

A Poisson distribution was used to describe the vertical fiber disposition. In addition, it was stated that a compound Poisson process could be used to model *vertical contagion,* a component of vertical interaction.

Lu and Carlsson (1996) performed a statistical analysis on paper structure which employed a Monte Carlo simulation method to generate the position and orientation of individual fibers in a 2D planar sheet. The position coordinates of the fiber centers were uniformly distributed and the orientation was defined by the warped Cauchy distribution. The result of the model was the prediction of volume fractions of bonds and fiber segments in relation with the apparent sheet density.

Niskanen and Alava (1994) presented a computer simulation study of planar random fiber networks. In the simulation, fibers with unit width and thickness were initially straight and parallel to the substrate. Each fiber was laid down independently on a 2D square lattice of linear size. The fiber would deform or bend after it contacted the underlying network. A bending flexibility factor, T_f, was defined as the largest allowed vertical deflection of the fibers from one lattice cell to the next (Figure 2-2). The relationship between T_f and the wet fiber flexibility was

$$T_f = (Cw_f F)^{\frac{1}{4}} \tag{2-12}$$

where C is the constant depending on experimental details, w_f is the width of fibers and F is the wet fiber flexibility.

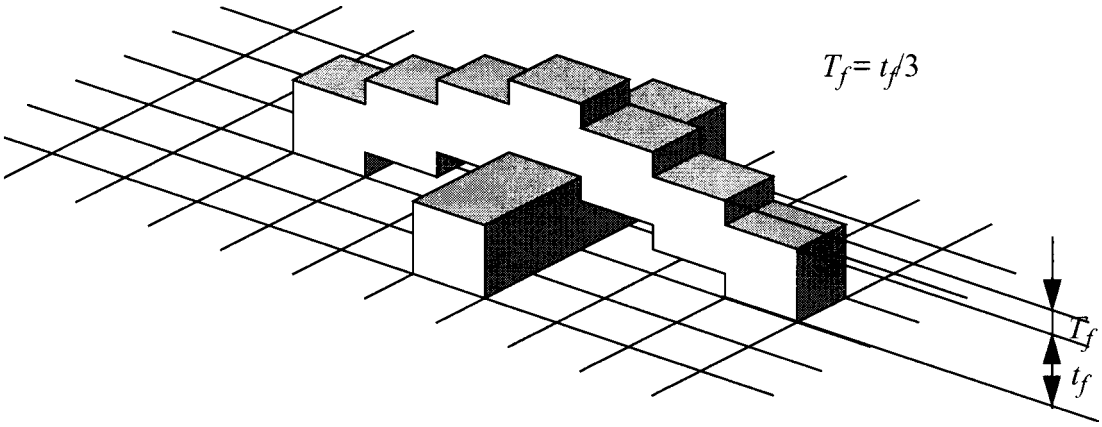


Figure 2-3. Fiber bending in Niskanen and Alava's simulation (1994).

It was found that roughness of the free surface decreased with the increasing of fiber flexibility. It was claimed that the model provides a connection between the properties of real planar fiber networks and ideal 2D networks. Later in 1998, Nilsen et al. later developed a simulation program, KCL-PAKKA, based on the same methodology. The input of the program included fiber dimensions (length, width and thickness), coarseness, flexibility and certain optical parameters, such as the refractive index. The paper-making pulps were modeled with three or four fiber fractions instead of continuous distributions of the fiber dimensions. For example, fibers had long, medium and short lengths (Nilsen et al., 1998).

Based on the finding that the mean number of sides per polygon is four and the distances between two adjacent crossings in a planar network of random lines are negative exponen-

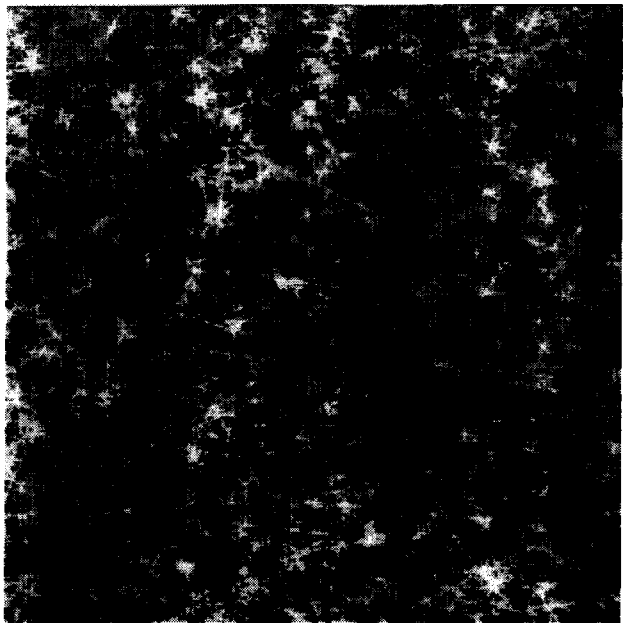


Figure 2-4. Surface contour of simulated paper structure (Niskanen and Alava, 1994)

tially distributed, Corte and Lloyd estimated the pore size distribution as the product of two negative exponential distributions (Corte and Lloyd, 1966). Since a negative exponential distribution is a special case of a gamma distribution (when the order of gamma distribution equals 1), Dodson and Sampson generalized free-fiber-length distribution with the gamma distribution and extended Corte and Lloyd's model into a product of two Gamma distributions (Dodson and Sampson, 1997). In their model, the free-fiber-length distribution is governed by

$$f(x) = \frac{b^k}{\Gamma(k)} x^{k-1} e^{-bx} \tag{2-13}$$

and the probability density function for equivalent pore radii is

$$p(r) = \frac{4b^{2k}\pi^{k}r^{2k-1}K_0(2br\sqrt{\pi})}{\Gamma(k)^2}. \tag{2-14}$$

where b is the parameter representing the state of flocculation, k is the order of Gamma distribution, and K,(x) is the 0 order modified Bessel function of the second kind.

The mean and variance of the pore size distribution are respectively given by

$$r = \frac{\Gamma\left(k+\frac{1}{2}\right)^2}{b\sqrt{\pi}\ T(k)'} \tag{2-15}$$

$$Var(r) = \bar{r}^2\left(\frac{k^2\cdot\Gamma(k)^4}{\Gamma\left(k+\frac{1}{2}\right)^4}-1\right) \tag{2-16}$$

It was found that the variables k and b were linearly dependent on each other, increased with grammage and decreased with flocculation so that the coefficients of variation of free-fiber-length and pore radius both increased with flocculation but decreased with increasing grammage (Dodson and Sampson, 1996).

Several other approaches have been reported for simulation of flocculated structures (Deng and Dodson, 1994, Farnood et al. 1995). In the method developed by Farnood, the distribution of local grammage in paper samples was modeled by the random deposition of low-grammage disks representing loose flocs (Figure 2-5). The diameter of disks was characterized with different distributions, uni-size, uniform and lognormal. Analytical expres-

sions for the variance of random structures of disks as a function of disk properties were derived. It was found that a lognormal distribution of disk diameters outperformed the other two diameter distributions in terms of predicting the variance of commercial paper samples.

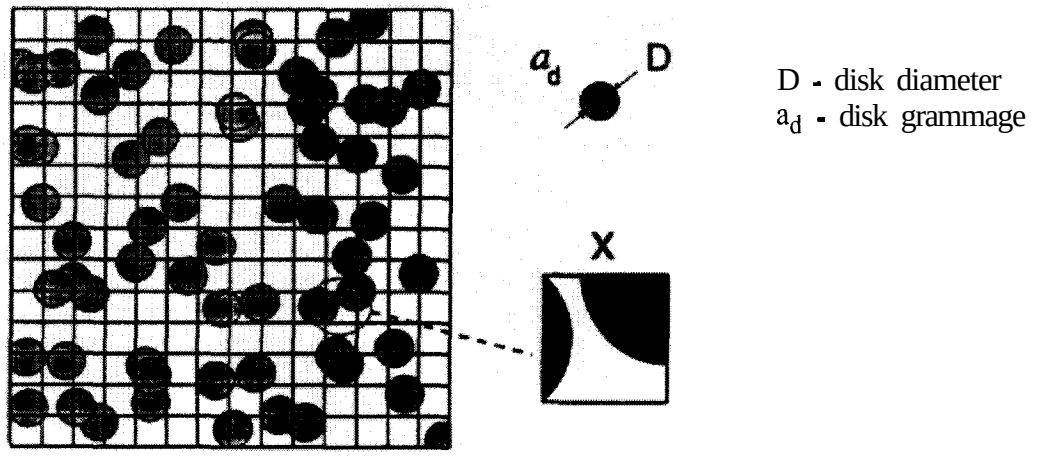


Figure 2-5. Partitioning a random structure of disks into square inspection zones of side x (Farnood et al., 1995)

Farnood et al. (1997) further developed this method with construction of images (Figure 2-6) of random disk structures by a computer simulation software. The input to the software included the mean and standard deviation of a lognormal distribution and disk grammage.

2.2. Models of Wood Particulate Composites Structure

In the area of forest products, some efforts have been made in modeling the mechanical properties of wood composite materials. Suchsland (1959) is generally acknowledged as the first to investigate the horizontal density variation of particleboard in the plane of the board: a Binomial distribution was used to describe the organization of the flake mat. Har-

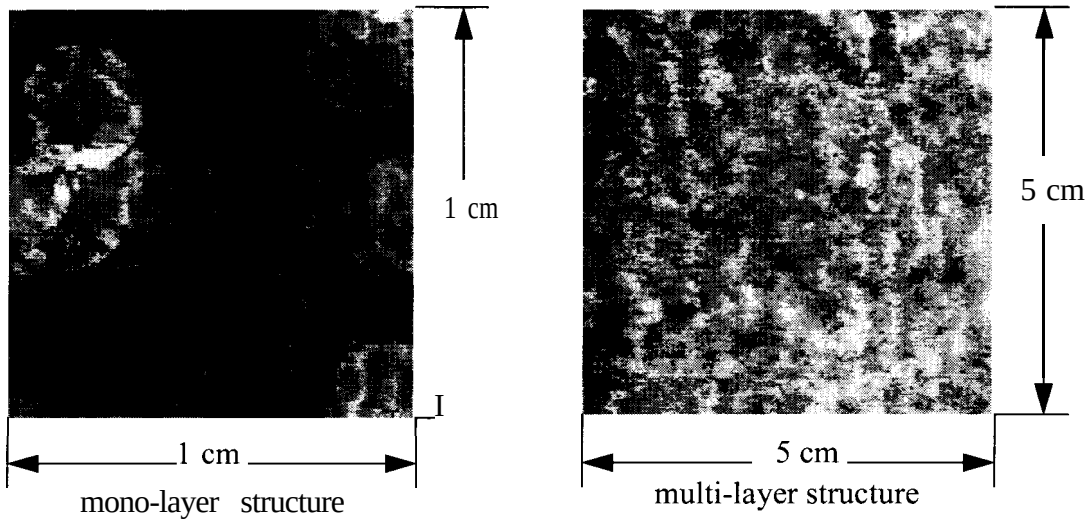


Figure 2-6. Gray level plots of the distribution of local grammage (Farnood et al., 1997).

less et al. (1987) introduced a model to predict the vertical density profiles of particleboard. The model simulated the physical and mechanical processes occurring in the press and mat system. Several researchers investigated the horizontal density distribution. Another structural model developed by Suchsland and Xu (Suchsland and Xu, 1989) consisted of narrow veneer strips arranged in mutually perpendicular layers. The number of veneer overlaps within the matrix represented the variation of the amount of wood material, and concomitantly the density variation. Xu and Steiner (1994) studied the variation of the horizontal density distribution of waferboard by relating the standard deviation of the horizontal density to the specimen size. They indicated that the variation decreased as the specimen size increased.

The most comprehensive microstructural model for wood composites was developed by Dai and Steiner (1994a, 1994b). Similar to the method used by Kallmes, Bernier and Corte (196 1), the model was based on the assumption that the thickness of wood elements (flake)

was zero so that all flakes lay parallel to the horizontal plane. The structural properties of the flake network were treated as random variables characterized by Poisson and exponential distributions. The monolayer model predicted distribution of: 1) flake centers; 2) flake area coverage; 3) free flake length and 4) void size over the flake layer network. A Monte Carlo simulation method was used to generate the uniform random variables for the simulation of flake deposition process. A descriptive model of multi-layered random flake mat was modeled as a summation of a series of two-dimensional randomly arranged flake layers (Dai and Steiner, 1994a, 1994b).

Another work for characterizing the spatial structure of wood strand mats was done by Lang and Wolcott (1996). A Monte Carlo simulation method was employed to generate random variables including the number of strands in the centroids of imaginary strand columns of finite size, the vertical distances between the adjacent strands (or void height) and the location of the column centroid relative to the constant length of each strand. They found that the Poisson distribution with a normally distributed mean was more suitable to describe the number of overlap strands. This was different from Dai and Steiner's assumption that the strand deposition in randomly formed mat follows the Poisson Distribution with a fitted mean. The parameters for the normal distribution were determined for manually formed strand mats. It was also found that both the void height and the location of the strand column centroid relative to the constant length of each strand were lognormally distributed.

Stahl et al. (Stahl et al. 1997, Stahl and Cramer, 1998) described a modeling procedure that generates and analyzes a 3D finite element model of the network microstructure in a small

volume of low density fibrous composite. In this research work, the composite elements were cement coated wood strands. The position of each strand was defined by a random point in the volume, which might be the position of the centroid, and three orientation angles of the central line. The orientation angle in the plane of the panel was either random or from certain nonuniform distributions if a preferred direction was required. The out of plane angles of the central lines were generated from normal distributions with zero mean and small variance. Bonds formed when two strands' volumes intersected or overlapped.

A model based on stochastic principles was developed to simulate the 3D geometric micro-structures of wood fiber composites (Wang and Shaler, 1998). This model assumed ideal rigid and cylindrical fibers. The input variables included the parameters for the distribution functions which were used to characterize the dimensions and spacial organizations of fibers in the model. Local void fraction, density, number of fiber segment and fiber bonding area could be obtained from the output structures of the model. Since rigid fibers were assumed, unrealistic fiber-to-fiber interferences existed in the simulated structures. Figure 2-7 shows a simulated 3D microstructure.

2.3. Experimental Measures of 3D Microstructures

Image processing techniques has been increasingly applied in the study of wood composite structure. Hasuike, Kawasaki and Murakami (1992) developed a technique of evaluating the 3D geometric arrangement of fibers in a paper sheet based on the measurement of the locations of fibrous segments in the serial cross-sections of the sheet. It is of interest to note

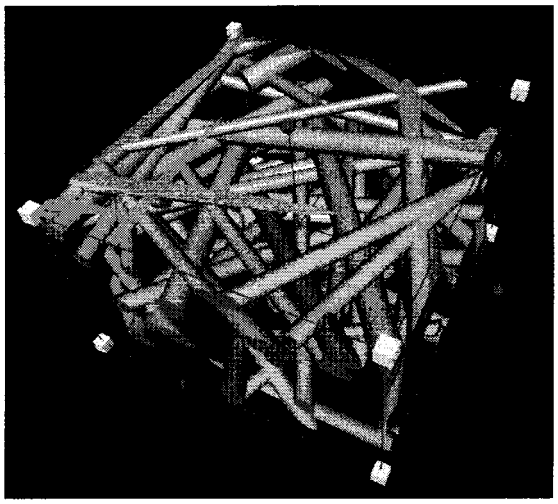


Figure 2-7. Simulated 3D microstructure of a wood fiber composite.

that the technique offered a means to visualize and analyze directly the extent of layering in papers of different types. Their method could be generally stated as

- prepare a series of vertical cross-sections;
- recording the relative coordinate positions of fiber peripheries in the sections by a digitizing system attached to a computer; analyzing the 3D geometric arrangement of fibrous segments with the aid of a computer graphics technique;
- reveal the entanglement and interconnection of fibers in 3D image and estimating the orientation of the segments in the Z-direction.

Jang et al. (199 1) reported a method for rapidly producing cross-sectional images of wood pulp fibers. In the report, an unbleached softwood kraft pulp fiber was measured with 177 μm^2 of cross-sectional area, 103 μm of the center-line perimeter and average 1.72 μm of cell wall thickness.

Recent work at University of Maine determined the ratio of interfiber void within MDF (medium density fiberboard) panels with image analysis techniques. The void space was determined using microtome sections and a light microscope interfaced with an image analysis system. Three dimensional microstructural models of panel were constructed based on sequential 2D images (Ramli, 1995). More recently, microtomography has been used to obtain and "stack" slices of MDF to create a reconstruction of the 3D structure. The 3D void fraction of the microstructure can be analyzed by manipulating the 3D voxel image with the aid of image analysis software (Shaler et al., 1998). These methods supplied a effective tool for the verification of 3D microstructural simulations.

2.4. Microstructure of Other Composites

The materials discussed in this section are different from wood fiber or particle composites in that most of them are composed of particles or fibers and solid matrix materials, and some are porous solids. The physical and mechanical properties of these materials also depend on the arrangement of features in 3D microstructural space. For some unidirectional composite materials, 2D models on the cross sectional properties would suffice. Some models use the homogenization method for reflecting the influence of material microstructure on the macroscopic behavior while the others try to characterize microscopic heterogneities with certain stochastic simulations. For example, Pyrz (1993) carried out a statistical analysis of fiber distribution by introducing a second-order intensity function and the Dirichlet tessellation to create a micro-mechanical model illustrating the influence of distribution statistics on the calculated stress field. In a 2D automatic mesh generator developed by Ghosh and Mukhopadayay (199 1) for finite element analysis of

random composites, random distribution of unidirectional fiber axes were assumed and the position of each axis was decided with two coordinates on the cross section generated by a random number generator.

Simulation of the random close packing of spheres has been used widely. A distribution of pore polyhedra in the random close packing of equal-sized spheres was obtained using computer simulation by Nolan and Kavanagh (1995). The method was also used in simulation of hard pigment particles and deformable latex particles in acrylic latex paint systems. The program input the particle size distribution of the pigment and the latex, the deformation or softness of the latex, and output the packing density (Nolan and Kavanagh, 1997).

Statistical simulation of multi-phase random media has been extensively studied. This class of composite is usually composed of two or more separately homogeneous materials. For example, in a two-phase composite, one of the phases may be in the form of discrete inclusions or particles which are distributed throughout the other phase, a continuously connected matrix (maybe fluid or void) according to some probability density functions. Different forms of the n-point probability function, which gives the probability of finding a certain subset of n points in the matrix phase and the remainder in the particle phase, were employed to characterize the microstructure of two-phase random media and impenetrable spheres were used to simulate the particles (Torquato and Stell, 1982, 1984). A model proposed by Quintanilla and Toquato (1996) for statistically inhomogeneous two-phase random media, including functionally graded materials, consists of fully penetrable Poisson distributed spheres.

A computer simulation model was reported by Louis and Gokhale (1995) to represent the 3D microstructure of a two-phase particulate composite, in which particles may be in contact with one another but do not overlap significantly. The program first generated freely overlapped and uniformly distributed spherical particles one by one and then analyzed the overlap and pulled the overlapping particles from one another according to a repulsion law. The model was used to quantify the connectedness of the particulate phase of a polymer matrix composite containing hollow carbon particles in a dielectric polymer resin matrix (Louis and Gokhale, 1995).

Roberts et al. (Roberts and Knackstedt, 1996, 1997, Roberts, 1997, Roberts and Garboczi, 1999) introduced several microstructural models which represent the complex morphology observed in a range of two-phase disordered material. Most of these models are based on the level-cut of Gaussian random field. It seems that the models are limited to isotropic materials. Figure 2-8 shows a gallery of simulated microstructures.

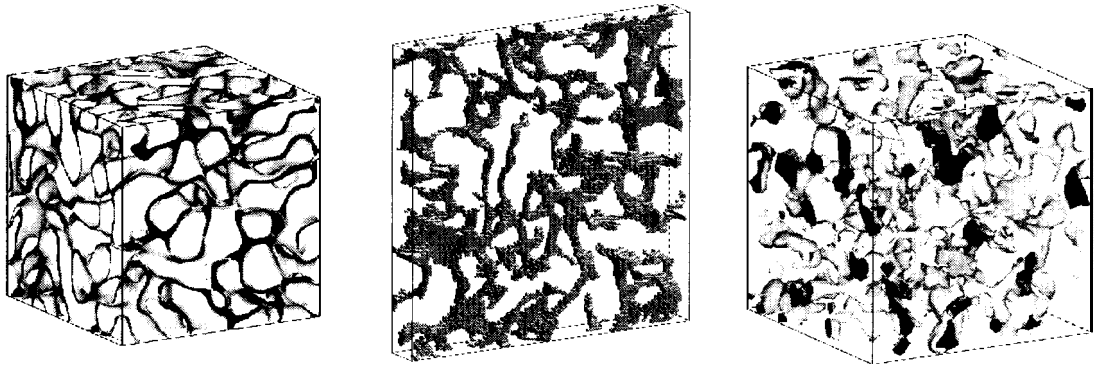


Figure 2-8. Three-dimensional realization of model microstructures.

(a) 2-level cut GRF model of composite media (Roberts and Knackstedt, 1996). (b) polymeric aerogel model (Roberts, 1997). (c) simulated silver phase of tungsten-silver composite (Roberts and Garboczi, 1999)

A particle-level simulation method was employed by Ross and Klingenberg (1997) to study the dynamic of flowing suspensions of rigid and flexible fibers. In the method fibers are modeled as chains of prolate spheroids connected through ball and socket joints (Figure 2-9). Flexibility or rigidity of fibers is modeled by varying the resistance in the joints. The motion of a fiber is determined by solving the translational and rotational equations of motion for each constituent spheroid.

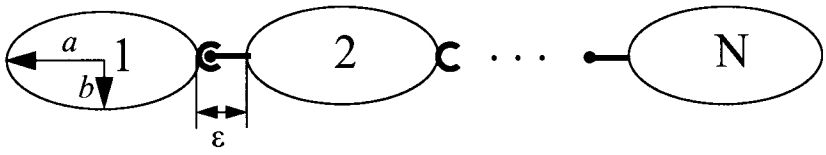


Figure 2-9. Linked rigid body model for a flexible fiber (Ross and Klingenberg, 1997). a and b are the spheroid major and minor axis lengths, respectively; ε is the separation between spheroids.

3 COMPUTER SIMULATION

To make composites in a laboratory, one needs to obtain the proper materials and the necessary equipment. Similarly, a computer also needs these necessities to build virtual wood composites. The 'materials' are virtual composite elements, either simulated fibers, flakes or strands, which are computerized data sets. The 'equipment' for making the composites is a software or a group of programs developed to generate and manipulate the data. This chapter describes how these two main components were designed and implemented.

3.1. Theories and Concepts

The construction of virtual composites is based on a stochastic simulation of composite elements within the composites. Since the research questions concerning the virtual composites in this research are mainly about the microstructural characterization, the simulation of the elements focuses on the description of their geometric character to serve as a basis for derived physical, mechanical and optical properties.

3.1.1. Random Numbers

Random number generation is essential for all stochastic simulation. Since truly random numbers are impossible to obtain in computer programming, algorithms must be employed to produce sequences of numbers that are statistically independent and uniformly distrib-

uted or so called *pseudorandom* numbers. The simplest and most frequently introduced random number generator (RNG) is probably the one shown below, which is a linear congruential generator (Ross, 1997 and Press, et al., 1990).

$$X_{n+1} = (aX_n + c) \ \ modulo\ m, n \geq 0 \tag{3-1}$$

By specifying positive integers a, c and m and an initial value X_0, the generator will recursively computes $X_1, X_2 \ldots$ and X_n. Each value is in the range of the series 0, 1 ,..., m - 1. The quantity of X_n/m is then taken as an approximation to a uniform (0, 1) random variable. The maximum period of this RGN is the value of m. In most computer systems m is the largest representable positive value of type *int*. The UNIX system on which the simulation software was developed, supplied two RNG's, **rand** () and **random** (). Both functions are parts of the standard C library. Function **rand** () uses a multiplicative congruent random-number generator with period 2^{32}, which returns successive positive integers between 0 and the largest positive short integer, 32767. Function **random** (), according to the manual pages, is a non-linear, additive feedback generator with a very large period, approximately 16 x ($2^{32} - 1$) . It returns successive pseudo-random numbers in the range (0, $2^{31} - 1$). Although the period of **rand** () seems large enough for this particular simulation as will be shown later, the weakness of the linear congruential method (Press, et al., 1990) prevents it from being reliably used without pre-modification. Therefore, **random** () was chosen to carry out the random number generation for the stochastic simulation. To obtain desired random variables in the range (0, 1), one may just take the values of **random** ()/RAND_MAX. In the C library, **RAND-MAX** is defined as 32767, which is

suitable for `rand ( ) .` If `random ()` is used instead, the maximum random variable has to be redefined as $2^{31} - 1$ for random variables in the range (0, 1] or 2^{31} for the range (0, 1).

3.1.2. Simulation of Probabilistic Deviates

3.1.2.1. Uniform Deviates

To simulate uniform deviates between a certain range a and *b*, the algorithm is designed as:

Step 1: Generate random variables U, `U = random ( )` /RAND-MAX.

Step 2: Set $X = a + (a + b) \cdot U$.

The variables X generated are uniform deviates in the range (a, *b).*

3.1.2.2. Normal and Lognormal Deviates

The normal deviates are generated with two basic Monte Carlo Methods, the Inverse and Rejection methods (Ross, 1997). The algorithm is shown below.

Step I: Generate U_1, U_1 being uniform on (0, 1).

Step 2: Set $Y_1 = -log(U_1)$. *Yl* is an exponential random variable with rate 1 (Inverse method),

$$g(x) = e^{-x}. \tag{3-2}$$

Step 3: Generate U_2, U_2 being uniform on (0, 1).

Step 4: If $U_2 \leq exp[-(Y_1 - 1)^2/2]$, set Y_2 = Y, and go to Step 5. Otherwise go to Step 1. Y_2 is a random variable with probability density function

$$f(x) = \frac{2}{\sqrt{2\pi}} exp(-x^2/2). \tag{3-3}$$

Step 5: Generate U_3, U_3 being uniform on (0, 1). Set $Z = Y_2$ if $U_3 \leq 0.5$. Otherwise, set Z = $-Y_2$. Z is a standard normal variable.

Step 6: Set X = $\mu + \sigma Z$, in which μ and σ are given parameters for the simulation. The random variables X are normal deviates with probability density function

$$N(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{3-4}$$

According to the definition of lognormal distributions, if $log X$ (X $\geq$ 0) has a normal distribution, variable X has a lognormal distribution. Therefore, lognormal deviates can be obtained by adjusting the algorithm for generation of normal deviates. After Step 6 above, let X' = e^X . Then X' is a lognormal random variable with probability density function

$$L(x) = \frac{1}{\sqrt{2\pi}\sigma x} exp\left[-\frac{1}{2}\left(\frac{log\, x - \mu}{\sigma}\right)^2\right] \tag{3-5}$$

In this equation, the parameters μ and σ are just the mean and standard deviation respectively of the normal deviates $log X$. The mean, μ', and standard deviation, σ', of the lognormal deviates are functions of μ and σ.

$$\mu' = exp\left(\mu + \frac{1}{2}\sigma^2\right) \tag{3-6}$$

$$\sigma' = exp\left(\mu + \frac{\sigma^2}{2}\right)\sqrt{exp(\sigma^2) - 1} \tag{3-7}$$

3.1.2.3. von Mises Deviates

The von Mises distribution is employed to simulate the orientation of individual composite elements. The von Mises distribution is a distribution function designed for data with finite ranges. Angles are typical examples of this type of data. The construction of the von Mises function ensures that the angular variables lie on the range of $0 \sim 2\pi$. The original probability density function of the von Mises distribution is (Mardia, 1972):

$$f(\theta|\Theta,\ k) = \frac{1}{2\pi I_0(k)} exp[k\cos(\theta - O)],\ 0 < \theta \leq 2\pi \qquad (3\text{-}8)$$

where Θ is the primary orientation angle and $0 \leq \Theta < 2\pi$, k is the concentration parameter with $k > 0$, and $I_0(k)$ is the modified Bessel function of the first kind and order zero.

The concentration parameter k is analogous to the variance of other continuous distributions. For example, if k = 0, the angular variables, θ, are uniformly distributed over the range (0, 2π). While the k is very large ($k \rightarrow \infty$), all angles equal Θ. If simulated objects have no directional properties, in other words, if the orientation ϕ appears no different from orientation $\pi + \phi$, the distribution may be characterized with the axial form of von Mises distribution (Harris, 1977 and Shaler, 1991):

$$f(\theta|\Theta,\ k) = \frac{1}{2\pi I_0(k)} exp[k\cos 2(\theta - 0)],\ -\frac{\pi}{2} < \theta \leq \frac{\pi}{2} \qquad (3\text{-}9)$$

If the primary orientation of the elements in the composite is set as O-degree, the function can be further simplified as:

$$f(\theta) = \frac{1}{2\pi I_0(k)} exp[k\cos(2\theta)] \quad (3\text{-}10)$$

To generate variables with the above probability density function, the algorithm may be designed as:

Step 1: Generate U_1 being a uniform random variable in the range (0, $\frac{e^k}{2\pi I_0(k)}$).

Step 2: Generate U_2 being a uniform random variable in the range ($-\frac{\pi}{2}$, $\frac{\pi}{2}$).

Step3: If $U_1 \leq \frac{1}{2\pi I_0(k)} exp[k\cos(2U_2)]$, set **X** $= U_1$. Otherwise, go to Step 1.

The variables X are the desired von Mises deviates.

3.1.3. Programming

There are two important methodologies for designing solutions to complex problems: top-down design (or structured design) and object-oriented design (OOD). The processes of implementing these two design paradigms are structured programming and object-oriented programming (OOP), respectively. Structured computer code design divides a problem into modules, the solutions of which create a solution to the overall problem (Dale, et al., 1996). A top-down designed program is a collection of interacting functions. In structured design and structured programming, data are passive quantities to be acted upon by functions. In OOD, a problem is decomposed into objects. Each object represents an instance of an abstracted data type (ADT) which has an internal state (the current values of its private data) and a set of methods. An object's state can only be inspected or modified through

methods invoked by another object. Objects within a program communicate with one another by message passing. In OOD and OOP, data play leading roles and the primary contribution of algorithms is to implement the operations on objects.

Top-down design yields an inflexible structure. Any modification on the top-level algorithm may force many lower level algorithms to be changed as well. In addition, the code developed with the technique is not easily reused. These are two major limitations for building large software systems. In contrast, OOD leads to programs that are more flexible and conducive to code reuse.

C++ is one of several programming languages supporting OOD and OOP. It provides a built-in structured type known as a class which is specifically designed for implementing objects. Generally, a class consists of private data members and public member functions. Within a program, the relationship between classes can occur in three typical ways:

1. Independent of each other,
2. Inheritance, and
3. Composition.

In the first case, two classes are independent of each other and have nothing in common. Inheritance is the mechanism by which one class acquires the properties - the data and operations of another class. Composition (or containment) is the relationship in which the internal data of one class includes an object of another class (Dale, et al., 1996).

3.2. Simulation Methodology

3.2.1. Scenario to be Simulated

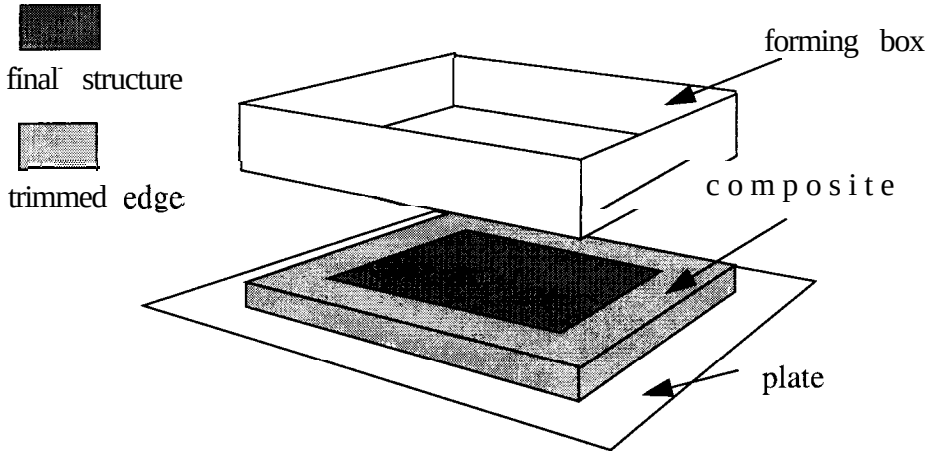


Figure 3-1. Forming Process of particulate composites

To make a wood composites, MDF for example, the wood which the fibers are made from has to be identified. This leads to the collection of the fiber properties. Further analysis may be carried out to obtain information about the fiber morphology. Besides the knowledge on fibers, the dimensions and target density of the composite must be set. Then, the total amount of fibers needed can be determined. Although resin plays a very important role in the mechanical and physical performance of most wood composites, the volumetric percentage is so small comparing that of the wood components that the geometric effect on the microstructure is negligible. Resin sprayed fibers are then dispersed on a plate that forms a horizontal plane. For pre-defining the edge of the composite mat, a rigid rectangular frame, named *Forming* Box in the current research, is usually placed on the plate

(Figure 3-l) and all fibers are deposited in the area surrounded by the *Forming Box*. In order to eliminate the possible edge effect on the composite sample, the four edges have to be trimmed off from the final panel. Therefore, the Forming *Box* is always larger than the desired composite panel. Finally, the mat is delivered into a press. The process in the scenario is not a standard procedure. Round forming frames may be used in making paper hand-sheet. In industrial practice, the forming of wood fiber or particle composites are usually continuous processes so that the edges are 'lose', which is another type of edge effect. The effect can be eliminated by trimming off certain width of edges.

If the fibers are assumed free of contact with each other during the journey into the Forming *Box,* and if the time interval is small enough, it will be clear that the forming process of the composite described in the previous paragraph is a sequential process in which fibers are dropped into *Forming Box* one by one. If an identification is applied to each fiber, e.g. *Fiber 1* for the first fiber dropped in, and *Fiber* 2 for the second and so on, the whole process will be

dropping *Fiber i* into a *Forming Box (i = 1, 2, ... n).*

The number *n* will be the total number of fibers needed to make a composite sample with required dimension and target density. Based on this scenario, if ith fiber's behavior in the composite can be described, the entire structure can be simulated by extending the description procedure to *Fiber i+1*, *Fiber i+2*,..., and *Fiber n.*

3.2.2. Simulation Environment

Based on the scenario described in the previous section, several entities should be defined in the simulation of virtual composite structures (Figure 3-2). A *Sample Space* is the space where a virtual structure locates. It is in Octant I of a right-hand Cartesian coordinate system, with three sides on x, y and z axes respectively. The x-y plane is parallel to the horizontal plane. The length of a virtual structure, X, is on the x-direction. The width, *Y*, is on the y-direction. The height or thickness, Z, is on the z-direction. A *Sample Area* is the area on the x-y plane covered by the structure. The four edges of a *Sample Area* form a *Sample* Edge. The area covered by the four trimmed edges is termed Edge *Zone*. The four trimmed edges are assumed to be the same width. Therefore, the size of an *Edge Zone* can be determined by a single parameter, *Width of Edge Zone, e,* in conjunction with the known structure dimensions. The combination of a *Sample Area* and an *Edge Zone* is the *Forming Area* which is the area surrounded by a *Forming Box*.

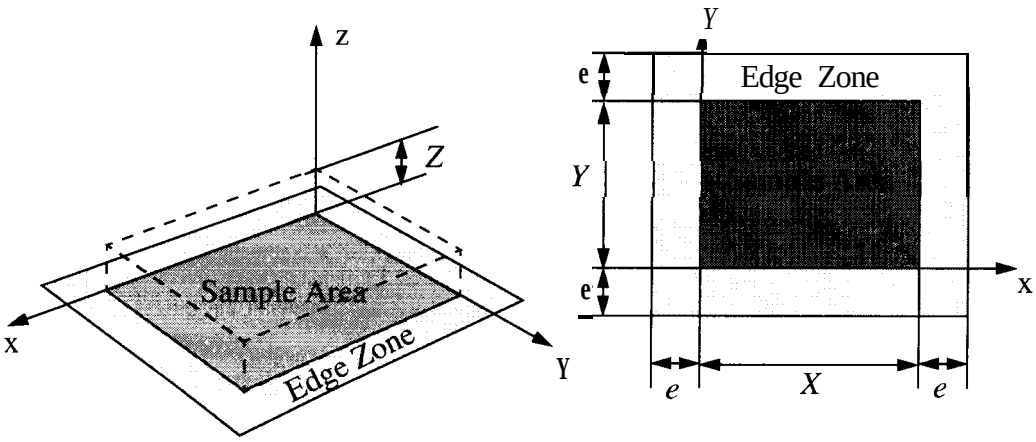


Figure 3-2. Environment for building a particulate composite structure.

The 3D *Sample Space* in the real world is a continuous system which consists of virtually an infinite number of points. In the digital computer, the system must be converted into a discrete system. The conversion is accomplished through the parameter *Resolution of Simulation* (ROS). All the parameters and variables related to the dimensional properties of elements and the virtual structure must be converted with the following equation,

$$\delta = (int)\left(\frac{\Delta}{\rho} + 0.5\right) \tag{3-11}$$

where ρ is the ROS, δ is the modified dimension, A is the real value of a dimension, and *(int)* is the integer type casting procedure in C++.

For example, if an `X` long and `Y` wide real-world composite sample is simulated, in the program, the length and width of *Sample Area* are defined as respectively,

```
xsize = (int)(X/ROS + 0.5);
ysize = (int)(Y/ROS + 0.5);.
```

After conversion, the area consists of `(xsize+1)*(ysize+1)` points. The distance between two orthogonally adjacent points equals the value of the resolution (ρ or `ROS` in source code) (Figure 3-3).

In the later sections, the dimensional variables, if not mentioned specifically, are all integral values modified with Equation 3-11.

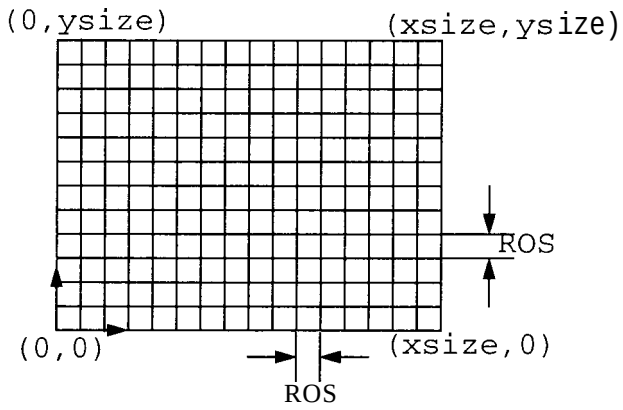


Figure *3-3.* Discretization of *Sample Area.*

3.2.3. Characterization of An Individual Element

3.2.3.1. Basic Geometry of an Element

Wood fibers free of deformation are generally tubular. The cross sections may be partly or totally collapsed due to the mechanical or chemical treatment in the pulping process. Wood flakes used in OSB or other wood composites are generally rectangular in shape. For simplification, it is reasonable to simulate a fiber before being packed into a composite as a hollow cylinder with uniform wall thickness and certain level of flatness on its cross section. An undeformed flake can be simulated as a cuboid.

As shown in Figure 3-4, a data structure which includes the following parameters is sufficient to represent a hollow cylindric element

- diameter (D),
- length (L),
- wall thickness to radius ratio (ω), and
- vertical deformation ratio (6).

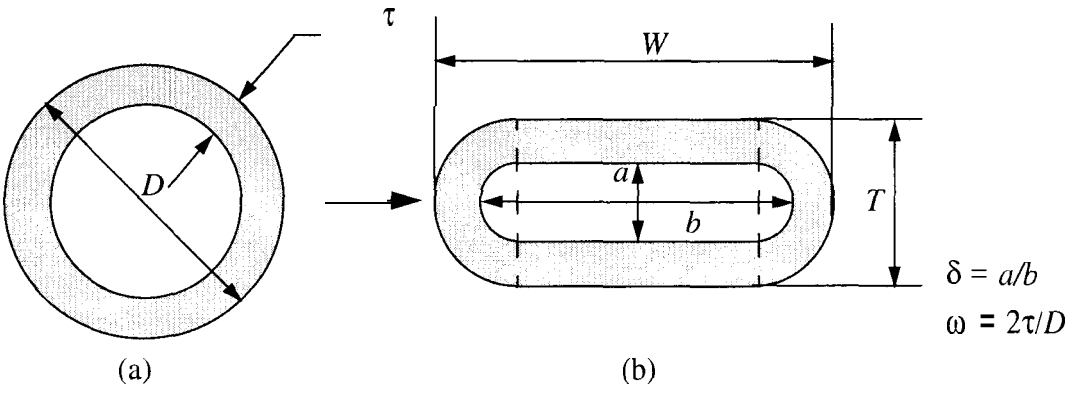


Figure 3-4. Cross sections of (a) a cylindrical element and (b) its conversion due to flattening.

The vertical deformation ratio represents the ratio of the vertical dimension to the horizontal dimension of the inner circle on the cross section. The ratio ranges from 0 to ∞. However, if load is only in vertical direction, the range of δ would be $0 \leq \delta \leq 1$. When $\delta = 1$, the cross section of the cylinder is a perfect circle. When $\delta = 0$, the cylinder is totally flattened. If width, W, and thickness, T, are used to describe a flattened cylinder,

$$W = \frac{\pi D - \pi D\omega}{\delta\pi + 2 - 2\delta} + D\omega \tag{3-12}$$

$$T = \frac{\delta\pi D - \delta\pi D\omega}{\delta\pi + 2 - 2\delta} + D\omega \tag{3-13}$$

Only the following parameters are needed to describe a cuboid

- width (W),
- length (L), and
- thickness (T).

It is assumed that the area of the cross section and the total volume of the element does not change during the deformation.

The dimensional values of elements in most composites are variables. Certain distribution functions may be employed to statistically characterize these variables. For example, normal distributions were used to simulate the diameter and lognormal distributions for the length of wood fibers (Deng and Dodson, 1994).

3.2.3.2. Morphology of an Element in Composites

Further deformation on the composite elements may occur during the forming and compression of the composite. Geometrical irregularities result due to the interaction among the elements. The parameters listed in the previous subsection are not enough to represent such a complex morphology. More sophisticated data structures have to be created.

To make the situation mathematically describable, the following assumptions are made:

1. The elements are initially straight,
2. There is no further horizontal deformation on elements,
3. If a load besides gravity is applied, it affects all the elements equally, and
4. No load is transferred between elements.

The first and second assumptions imply that the horizontal projection of any element is rectangular and does not change. The third and fourth assumptions eliminate the effect of any element to previously allocated elements, which also means that the morphology of an element depends on its own property, universal load, and the morphology of elements in

contact from underneath its body only. Therefore, to determine the morphology of *Element i*, the behavior of *Element I*, *Element 2*,..., and *Element i-l* has to be determined. Every element introduced in the *Forming Box* changes the status of the top surface of the assembly. The surface status will affect the behavior of the next coming element.

In the following part of this section, *Element i* will be used to elaborate the algorithm and methodology used for characterization of each individual element in the composite.

3.2.3.3. Definitions and Implementations

The objects identified for characterization of *Element i*, based on the physical entities involved in the real world, includes (Figure 3-5):

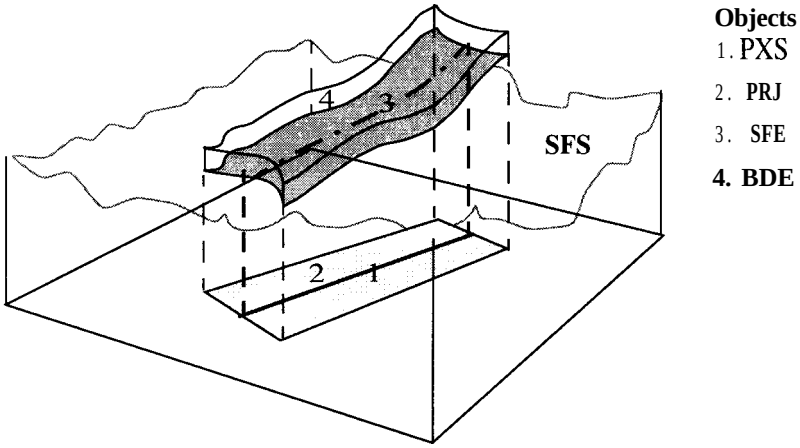


Figure 3-5. Objects contained in an element.

- horizontal projection of the element's major axis (PXS),
- horizontal projection of the element (PRJ),
- a specific surface in the element (SFE), and
- 3D body of the element (BDE).

The four objects listed above form a hierachy system (Figure 3-6). They are related by composition or containment, a mechanism in C++, by which the internal data of one class includes an object of another class.

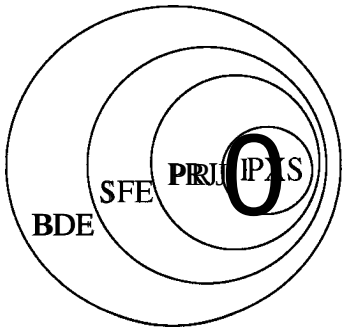


Figure 3-6. Structure of an element.

The top surface of the simulated composite structure (SFS) formed by *Element 1* to *Element i-1* does not belong to any element but relates to them.

Object 1. Surface *of Sample* (SFS)

SFS is a passive data structure. Information recorded in SFS includes the identification, `id`, of the elements at the top, the `height` of the virtual structure at any horizontal location within the *Sample Area,* and the local `coverage`, which defines the number of elements above the location. The status of SFS is updated by each element introduced into the composite structure. The surface is simulated as a two-dimensional dynamic array of nodes. Each node corresponds to a point in *Sample Area.* Node(i, j) has the same horizontal position with point(i, j) in *Sample Area.* Each node is a C++ record data type.

```
struct SFSNodeType
{
  int  id;
```

```
    int height;
    unsigned short coverage;
};
```

The two dimensions of the array are respectively xs `i z e` and ys `i z e`.

The advantage of using a dynamic array is that one dimension of the array can be specified dynamically or at execution time. In this case, the array of nodes is defined as:

```
SFSNodeType (*SFS)[MAX_SZ];
```

The `MAX-SZ` is the maximum array size in the program. It is determined based on the stack size limit of the compiler and the operating system. It represents the maximum possible width of SFS, as well as the simulated structure. The value has to be fixed at the time of compiling. For example, if the size of an array is xs `i z e` *ys `i z e` units, the total memory allocated for this array must be xs `i z e` *`MAX_SZ` units and ys `i z e` has to be less than or equal to `MAX_SZ`. Consider one unit as being defined as the memory size for one element of the array, e.g. for SFS, it would be 10 bytes. The wasted amount of memory could be xs `i ze` * (`MAX-SZ` –ys `i ze`) units. If a static array was used instead, the definition would be:

```
SFSNodeType SFS[MAX_SZ][MAX_SZ];
```

The memory allocated for it would be `MAX-SZ` *`MAX_SZ`. The wasted amount of memory could be `(MAX_SZ*MAX_SZ-xsize*ysize)` units. Here `xsize` hadtobelessthan or equal to `MAX-SZ,` too. Therefore, by using a dynamic array, potentially, `MAX-SZ*` `(MAX-SZ-xsize)` units of memory could be saved if `xsize` is less than or equal to `MAX-SZ` . For example, if `MAX-SZ` is set up as 4000, and let xs `i z e` and ys `i z e` be 2000 and 3000, respectively, when a dynamic array is chosen to implement SFS, the

wasted memory will be 2000 x (4000 – 3000) = 2000000 units. If a static array was used instead, the wasted memory would be 4000 x 4000 – 2000 x 3000 = 10000000 units.

Object 2. The *Horizontal Projection of Element's Axis* (PXS)

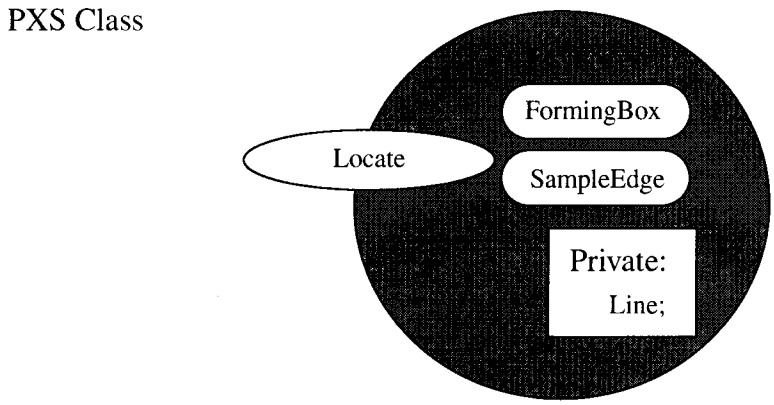


Figure 3-7. Design of Horizontal Projection of an Element's Axis (PXS).

The basic structure of PXS is shown in Figure 3-7. Since no horizontal deformation of an elements is assumed, the horizontal projection of the axis is a straight line on the x-y plane. As such, only two pairs of x-y coordinates are needed to build the object, specifically, the end-positions of the line. Two rules must be applied to these two pairs of coordinates.

1. They are related to each other by the length of the element who owns the axis.
2. Both of them should be within the area surrounded by *Forming Box*.

One of the member functions, Locate ( ) , determines the location of PXS. Locate ( ) is a five-step process diagramed in Figure 3-8.

Step 1. Generate the mid-point of PXS.

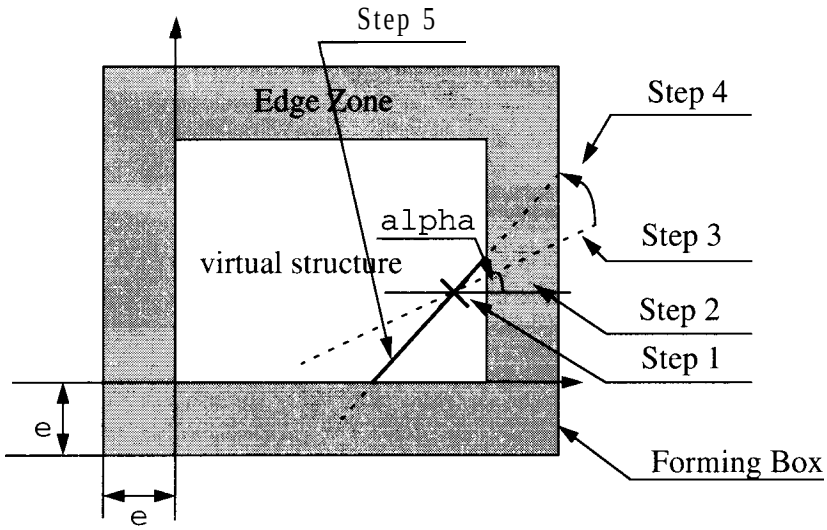


Figure 3-8. Effect of Forming Box on PXS.

Step 2. Generate the horizontal orientation of PXS.

Step 3. Calculate PXS' end coordinates.

Step 4. Adjust the end coordinates by *Forming Box.*

Step 5. Adjust the end coordinates by *Sample Edge.*

At Step 1, elements are distributed in either a *random* or *even* manner. In the case of *random,* the coordinates of a mid-point `(x0, y0 )` are uniform distribution deviates. The ranges of the two distributions are `(-e, xsize+e)` and `(-e, ysize+e)`, respectively, in which the parameter e represents the Equation 3-l1 modified width of the edges to be trimmed off. The ranges also define the location of simulated *Forming Box* which is represented by function `FormingBox ()`. If the *even* manner is desired, *Forming Area* will be evenly divided into *n* sub-areas, in which n is the total number of elements or predicted total number of elements based on the dimensional means of the elements and volume of the virtual structure, i.e.

$$n = \frac{4XYZ}{\pi\mu_l\mu_d^2\omega(2+\omega)} \text{ (for cylindrical elements)} \tag{3-14}$$

$$n = \frac{XYZ}{\mu_l\mu_w\mu_t} \text{ (for cuboid elements)} \tag{3-15}$$

where **X,** Y, Z are the dimensions of the simulated structure, and μ_l, μ_d, μ_w, μ_t are the mean values of length of elements, diameters of cylindrical elements, width and thickness of cuboid elements, respectively, ω is the mean wall thickness to radius ratio of the cylindrical elements.

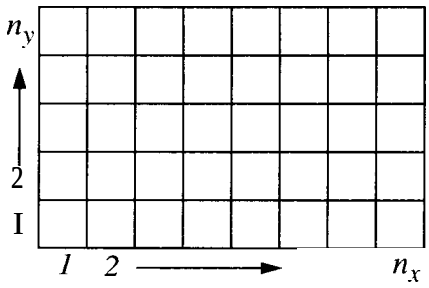


Figure 3-9. Even distribution of PXS's mid-point.

The division into sub-areas is determined by solving the the following function group:

$$\begin{cases} n = n_x n_y \\ \dfrac{n_x}{n_y} = \dfrac{X}{Y} \end{cases} \tag{3-16}$$

where X and Y are the length and width of the structure, respectively, n_x, n_y are the numbers of sub-areas along x and y directions, respectively, and n is defined from Equation 3-14 or Equation 3-15 as appropriate. Each center of a sub-area will hold one mid-point of an axis (Figure 3-9).

At Step 2, the angle between PXS and the x-axis, denoted as a $(-\pi/2 < a \leq \pi/2)$, is generated. The generation is governed by a von Mises distribution. At Step 3, based on the element's horizontal orientation and length, the position of PXS can be decided, which is represented by the x-y coordinates of the two ends (xl , y1) and (x2 , y2).

```
x1=(int)(x0-L*cos(alpha)/2+0.5);
y1=(int)(y0-L*sin(alpha)/2+0.5);

x2=(int)(x0+L*cos(alpha)/2+0.5);
y2=(int)(y0+L*sin(alpha)/2+0.5);
```

in which **L** is the length of the element modified with Equation 3-l1, and **alpha** represents the angle a in programs.

It is worth noting that the length of PXS actually should be the length of an element modified by the possible vertical orientation and deformation. However, at this point in the simulation process, since it is impossible to obtain the information, the length of the element has to be used. At Step 4, function `FormingBox ()` is called to check whether one or both of the ends are outside the range ($-e$ Ix $\leq xsize +$ e and $-e \leq$ y $\leq ysize + e$). If so, `FormingBox ()` will adjust the positions by rotating PXS along the mid-point. This rotation mimics the restraining effect of a physical side boundary.

For some area a mid-point can never be located due to geometrical confines (Figure 3-10). For these situations, the mid-points will be regenerated.

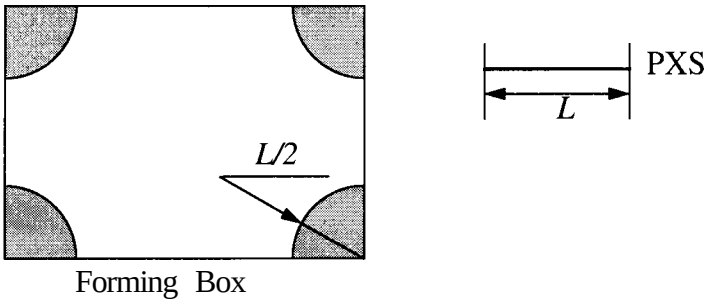


Figure 3-10. Possible position of PXS's mid-point (unshaded area).

At the final step, the parts of PXS outside the range $0 \leq x \leq X$ and $0 \leq y \leq Y$ are truncated by function `SampleEdge ()`. The coordinates of the end outside an edge is replaced by the coordinates of the intersection between PXS and the edge (Figure 3-8).

Object 3. *Horizontal Projection of an Element* (PRJ)

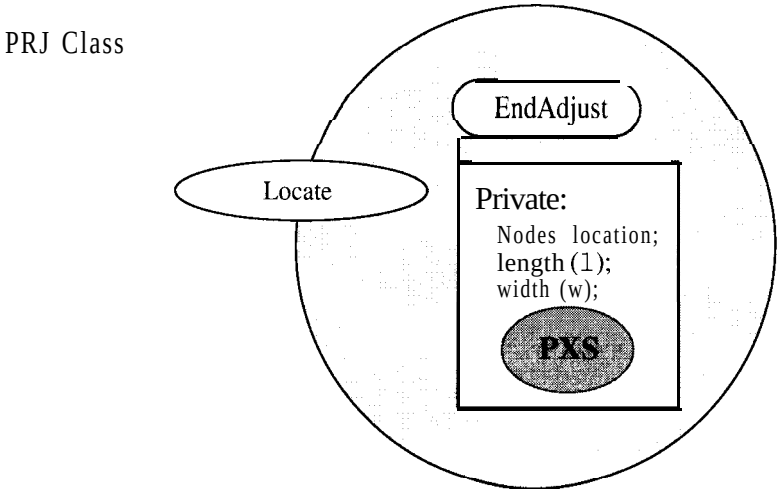


Figure 3-l1. Design of Horizontal *Projection* of an element (PRJ).

The projection of a cylinder on a plane will be a rectangle if the major axis is parallel to the plane. The projection of a cuboid on a plane will also be a rectangle if one of its edges is parallel to the plane. At this point in the simulation, it is assumed that the element is free of deformation and parallel to the horizontal plane so that its horizontal projection is rectangular. With this assumption, if the length and width of the projection are 1 and w, respectively, a reasonable design of PRJ should be a two-dimensional **(w+l** rows and 1+1 columns) array of points (Figure 3-11). The data structures of PRJ include the lower level object, PXS, a two-dimensional dynamic array of nodes and the sizes (length and width) of the array (Figure 3-12). Each node is a C++ record data structure which records the x and y coordinates of the node.

```
struct  PRJNodeType
{
   int x;
   int y;
};
```

The definition of PRJ is:

```
class PRJ
{
  public:
    void Locate(...);
    ...
  private:
    PRJNodeType (*prj) [MAX-SZ];
    int w;
    int 1;
    PXS  axis;
};
```

The maximum possible length of the projections in the structure is determined as the value of **MAX-sz** since usually the virtual structure should be large enough to contain a whole body of a simulated element in both x and y directions.

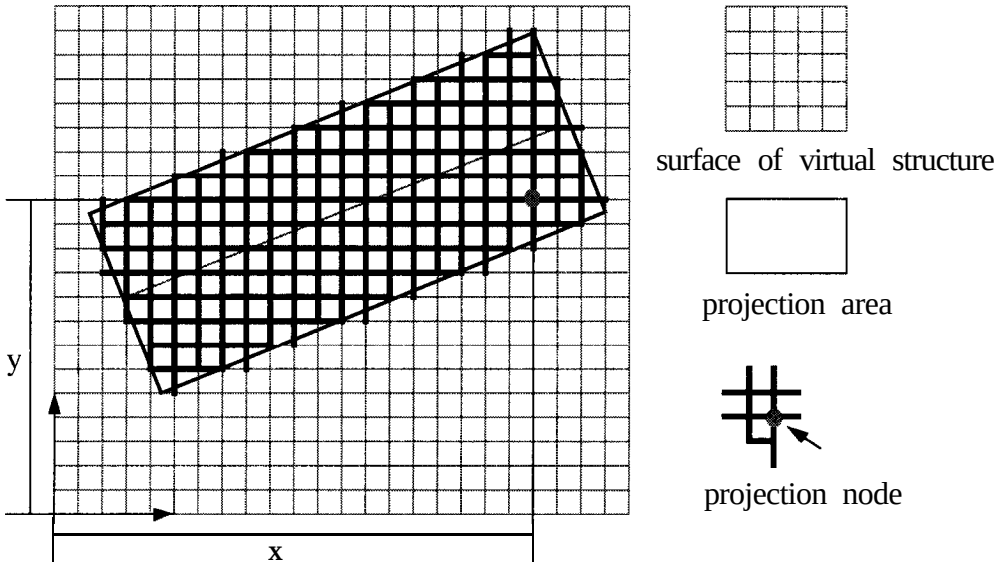


Figure 3-12. The Projection of an element is designed as a 2D array of nodes.

The main operation on PRJ is to locate all the nodes on the surface of structure (SFS) covered by the projection. It is performed by a member function `Locate ( ) . Locate ( )` decides the length (1 or *l*) and width (w or *w*) of the projection first. The length is calculated according to the coordinates of PXS. The width is calculated from the width of an element, adjusted by the orientation of the PXS (Figure 3-13):

```
if(alpha>0.25*pi | alpha<-0.25*pi)
  {
    w=(int)(W/abs(sin(alpha)+0.5);
    l=abs(y2-y1);
  }
else
  {
    w=(int)(W/abs(sin(alpha)+0.5);
    l=abs(x2-x1);
  }
```

where `pi` represents the value of π in the program, **`W`** is the width of the element (modified with Equation 3-11), and **`abs`** `()` is a C function which returns absolute value of an input variable.

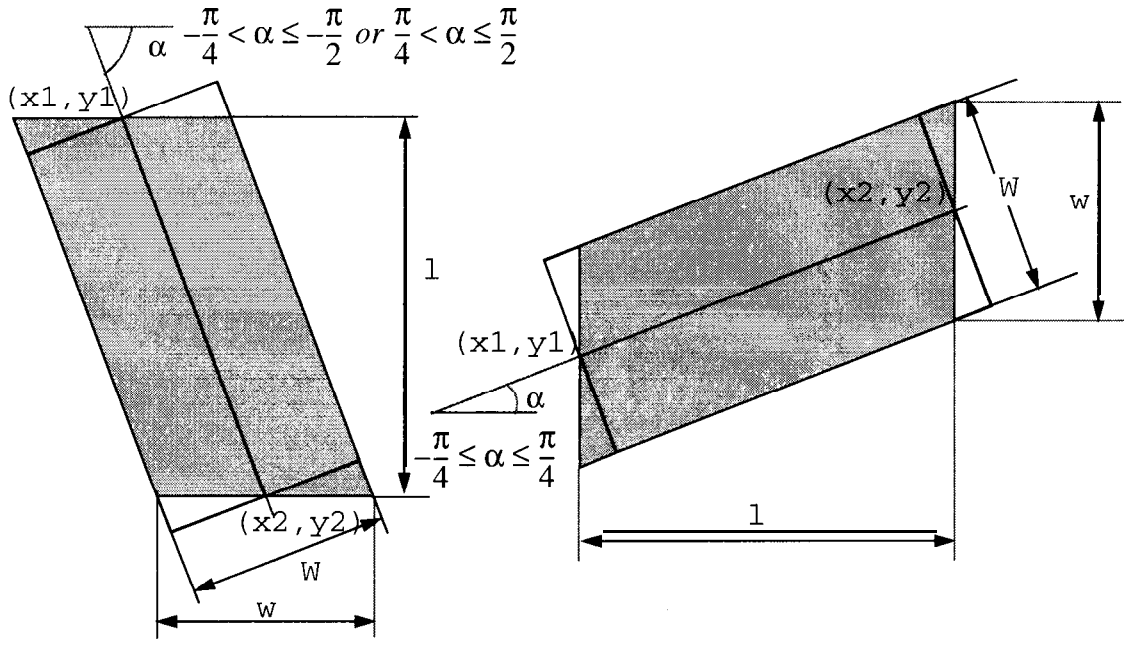


Figure 3-13. Defining sizes of a PRJ.

After these two sizes are decided, a parallelogram is drawn first. A function, `EndAdjust()`, is then called to modify the parallelogram into a rectangle by adjusting the ends (Figure 3-14). It seemed that this process could be done more easily by transforming a rectangular array of nodes. However, by doing that, some nodes on the projection would not coincide with the nodes on the surface of the virtual structure due to rounding in the transformation process. In the parallelogram to rectangle process, `EndAdjust()` translates the node on parallelogram projection outside the range between two end lines (shaded area in Figure 3-14) with the following distances in x and y directions:

```
Xt=(+/-)(x1-x2);
Yt=(+/-)(yl-y2);
```

where Xt and Yt are the translation in x and y directions, respectively, xl, yl and x2, y2 are the coordinates of PXS, and signs represent the directions of translation.

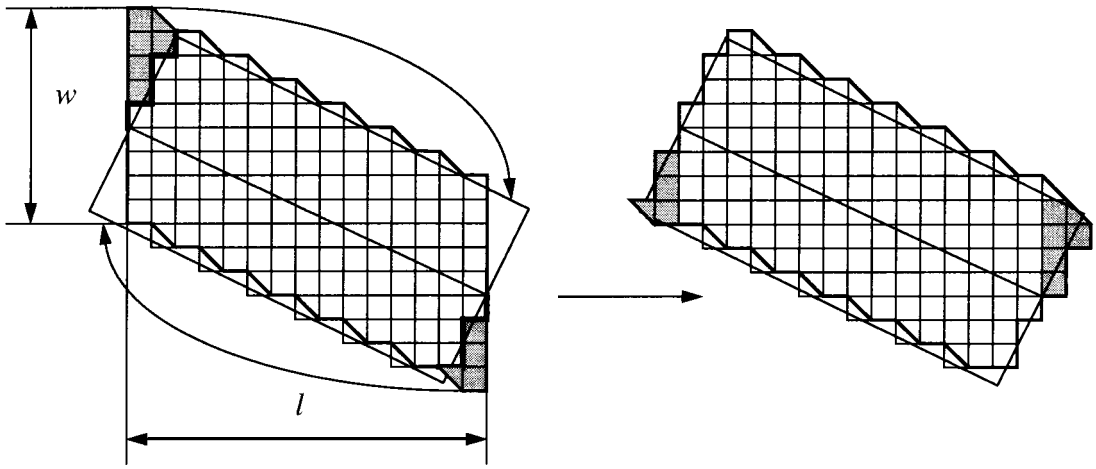


Figure 3-14. Process on adjusting the ends of the Projection.

After translation, the nodes on the projection have to be reordered. For example, consider a node whose positions are (i, j) on the projection (i = 0, 1, 2, ... w and j = 0, 1, 2, ... l) and (x, y) in the *Building Area,* which can also be termed as local coordinates and global coordinates respectively. After translation, the global coordinates (x', y ') may be expressed in terms of the original coordinates as:

$$\begin{cases} x' = x + X_t \\ y' = y + Y_t \end{cases} \tag{3-17}$$

The local coordinates have not been changed. This would result in a structural disconnection in later processes. Since all the nodal translations are along the longitudinal axis of the element, only parameterj should be reordered. The algorithm for reordering point (i, j) is:

Step 1. Calculating two end edges of the rectangle.

Step 2. If on row i, the point on an end edge is (i, j_e), for each point in this row, if $j + j_e \geq l$, set $j' = j + j_e - l$; if $j + j_e < 0$, set $j' = j + j_e + l$.

The reordered local position is (i, j').

Object 4. Surface of an Element

Physically, SFE on a cuboid element could be either the top or the bottom surface. For a hollow cylindrical element, SFE is defined as the virtual surface between the top and bottom halves, which goes through the major axis of the element (Figure 3-15).

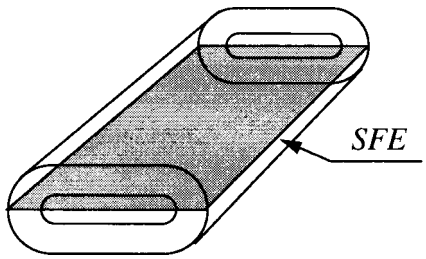


Figure 3-15. A SFE on a cylindrical element.

The data structures of SFE include the lower level object, PPJ, a two-dimensional dynamic array of nodes and the sizes (length and width) of the array (Figure 3-16). Each node is a record data structure of the form:

```
struct SFENodeType
{
   int x;
   int y;
   int z;
   int  touch;
};
```

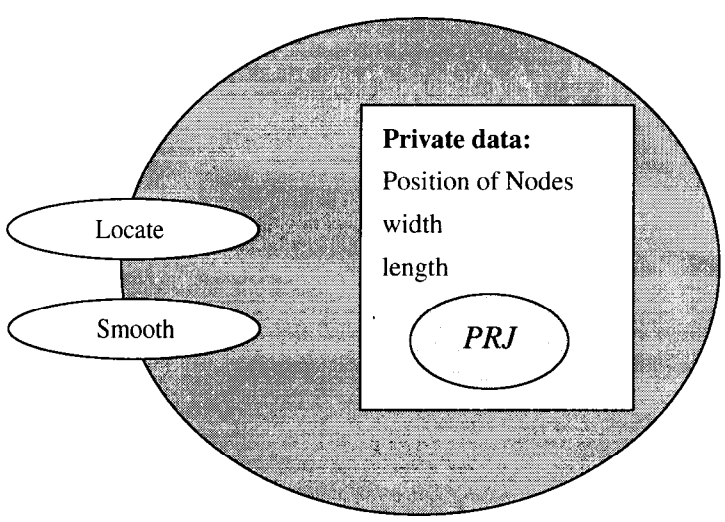


Figure 3-16. Design of a *Surface* in an element (SFE).

The variables, x, y, and z, define the location of a node in the 3D space. The variable `touch` records the identification of the element underneath the node, which would be contacted by the current element. This value is already recorded on SFS.

The definition of SFE is:

```
class SFE
{
  public:
    void Locate(...);
    void Smooth(...);
```

```
    ...
  private:
    SFENodeType (*node)[MAX_SZ];
    int w;
    int 1;
    PRJ projection;
};
```

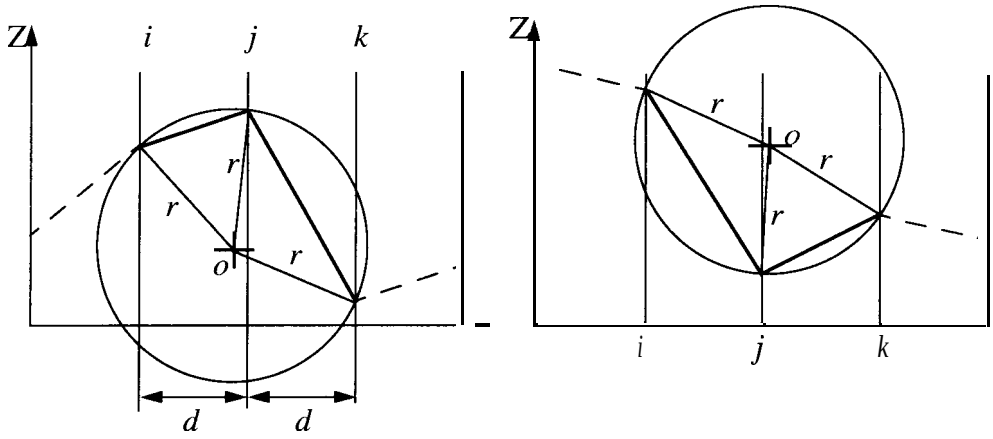
Since PRJ is also the projection of SFE, the horizontal position of each node on SFE are the same as the position of the correspondent node on PRJ. The vertical position of each node is temporarily set as the height of the node on SFS at the same horizontal position:

```
node[i][j].x = projection.node[i][j].x;
node[i][j].y = projection.node[i][j].y;
node[i][j].z = SFS[I][J].height;
node[i][j].touch = SFS[I][J] .id;
```

in which I = `node [ i ][ j ].x` and `J = node [ i] [j ]. y`. The above process is carried out by the member function `Locate ( )` in the class.

The vertical position of each node must be adjusted according to the flexibility or stiffness of the element and the possible load. The flexibility of an element is represented by a geometric value, i.e. the minimum radius of curvature of the element in vertical direction. The adjustment is performed by another member function in this class, `Smooth ()`. Function `Smooth ()` adjusts the curvature on every *row* and every column of the SFE nodes. Each *row* or ***column*** is idealized as an elastic beam. Hence, the SFE contains w+l beams in the length direction and *l*+1 beams in the width direction.

Every three successive nodes, e.g. ***Node i, Node*** j and ***Node k,*** may form a triangle on a vertical plane if their heights are different (Figure 3-17).



Convex Concave

Figure 3-17. Calculate the curvature of a beam in the simulation,

The radius of the circumscribed circle of the triangle is defined as the radius of curvature r for the section of the beam between *Node i* and *Node k*. **Smooth** () compares r with the minimum radius of curvature of the beam, R, which is determined with:

$$R = \frac{EI}{M} \tag{3-18}$$

where E is the Young's Modules of the material, I is the moment of inertia of the beam, and M is the bending moment applied on the beam. If $r < R$, one or two nodes must be raised up until $r = R$ (Figure 3-18).

The limitation for applying this method is $\rho \leq R$ because if ROS is greater than R, the radius of a circumscribed circle for any three successive nodes will also be greater than R.

The moment of inertia, I, depends on the geometry of the cross section, Generally,

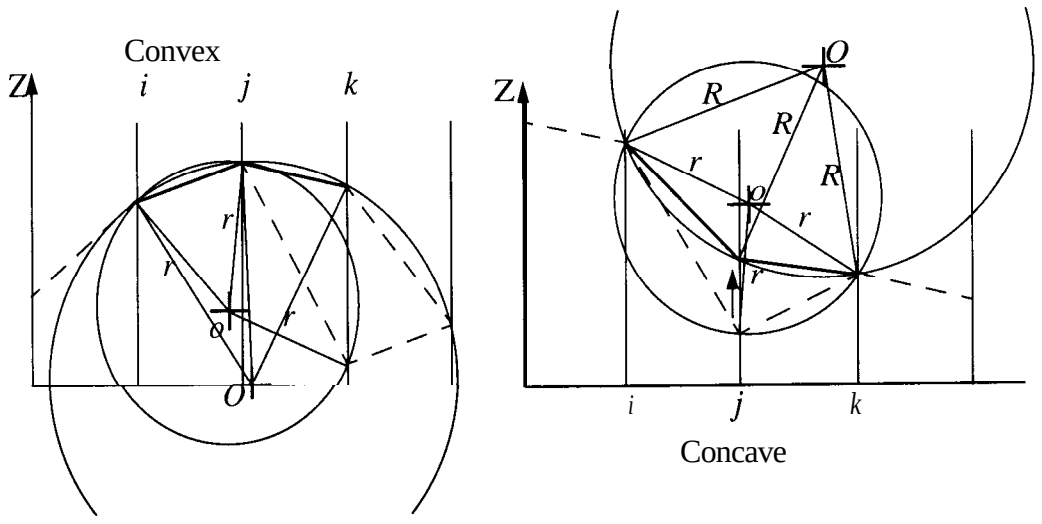


Figure 3-18. Adjusting curvature of SFE according to the minimum radius of curvature.

$$I = \int z^2 dA \tag{3-19}$$

where A is the cross-sectional area of the beam, and z is the distance from any point on the cross section to the principal axis of the cross section, which is always parallel to the x-y plane (horizontal plane).

For a cylindrical element, the values of moment of inertia in longitudinal and transverse directions (Figure 3-19) can be derived respectively as

$$I_1 = \frac{\pi[T^4 - (T - \omega D)^4]}{64} + \frac{(W - T)(T^3 - (T - \omega D)^3)}{12} \tag{3-20}$$

$$I_2 = \frac{L'T^3 - L'(T - \omega D)^3}{12} \tag{3-21}$$

where W is the width of the element calculated with Equation 3-12, T is the height of the element calculated with Equation 3-13, and L' is the length of the element after truncated by `SampleEdge()`.

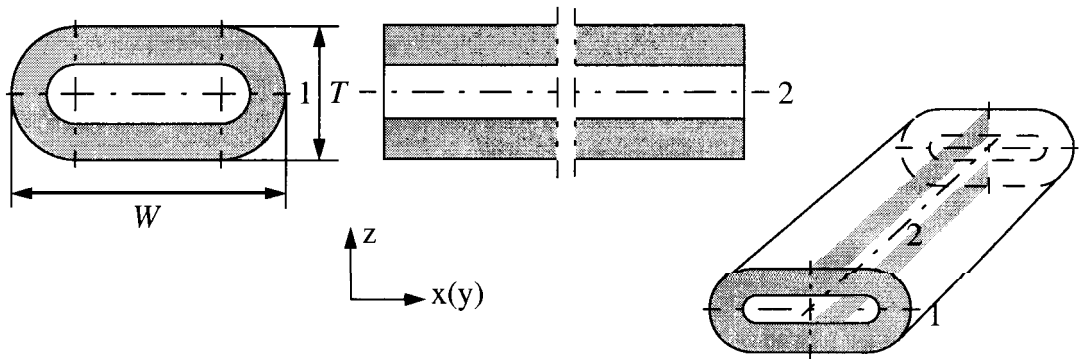


Figure 3-19. Cross sections of a cylindrical element in longitudinal and transverse directions.

For a cuboid element, the values of moment of inertia in longitudinal and transverse directions are, respectively

$$I_1 = \frac{WT^3}{12} \tag{3-22}$$

$$I_2 = \frac{L'T^3}{12} \tag{3-23}$$

where W is the width, T is the height, and L' is the length of the cuboid element after truncatedby `SampleEdge()`.

Therefore, the moment of inertia of a beam in the direction of length and the moment of inertia of a beam in the direction of width could be respectively calculated as

$$I'_1 = \frac{I_1}{w + 1} \tag{3-24}$$

$$I'_2 = \frac{I_2}{l + 1} \tag{3-25}$$

where w and l are the width and length of the PRJ, respectively.

Since the elastic modulus and the moment of inertia are constant, the radius of curvature of a beam is purely a function of the moment applied. Based on the assumptions, the load on an element is always uniformly distributed. Therefore, any of the beams in an element can be modeled as shown in Figure 3-20.

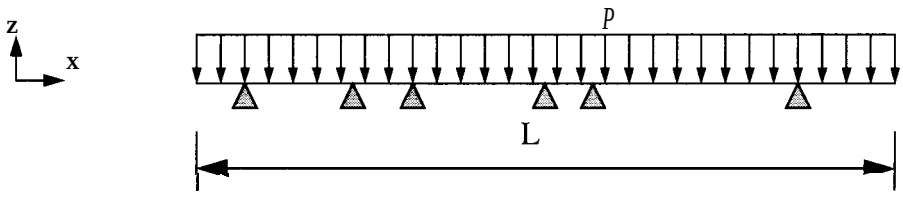


Figure 3-20. Modeled loading and supporting on a beam.

The beam is simply supported. The number and location of the supports, which determine the value of the moment along the beam, depend upon how the element to which the beam belongs interacts with other elements underneath its body. Since at this point in the simulation process, the element has not settled down, the number and location of supports are unknown. To solve this problem, the *maximum* possible moment can be used instead, which corresponds to the *minimum* radius of curvature. The maximum moment occurs when the beam is simply supported at its two ends (Figure 3-21), which is calculated as:

$$M_{max} = \frac{PL'}{8} \qquad (3\text{-}26)$$

where L is the length of the beam (If the beam is parallel to the major axis of the element, $L = l$, and if the beam is perpendicular to the major axis, $L = w$.), and p is the load on a node calculated based on the applied load to the composite, $\boldsymbol{P}$ and the gravity of the element, G,

$$p = \frac{\boldsymbol{P}}{(\texttt{xsize} + \text{l})(\text{ysize} + 1) + (l + 1)(w + 1)} + \boldsymbol{G}, \qquad (3\text{-}27)$$

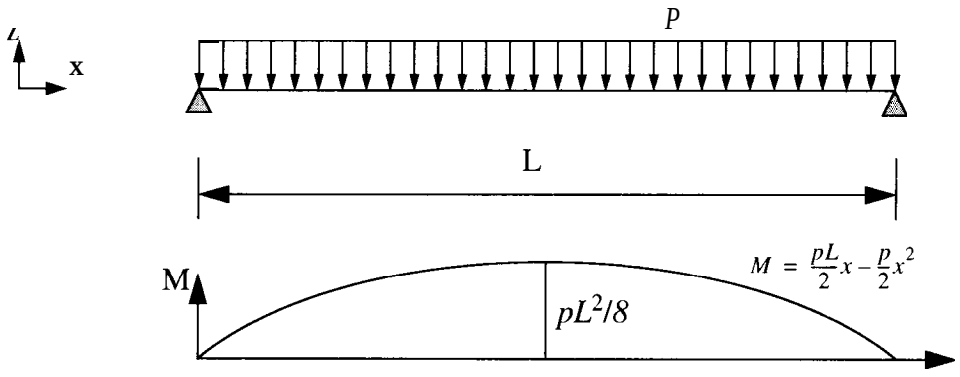


Figure 3-21. Maximum moment occurs when a beam is simply supported at the two ends.

If a node is adjusted by `Smooth ()`, it will lose the contact with the elements underneath. The variable `touch` has to be modified,

```
node[i][j].touch = 0;
```

Object 5. 3D ***Body*** of an Element (BDE)

The data in the top level object include the information required for morphological description of an element in a 3D space. It includes the lower level object, SFE, a two-dimensional dynamic array of nodes and the sizes (length and width) of the array (Figure 3-22). Each node is a record data structure. The contents of node data are different between a cylindrical element and a cuboid element.

For a cylindrical element, each node is

```
struct  CylinderNodeType
{
   int x;
   int y;
   int z;
   int touch;
   int o_s;
   int  i-s;
};
```

in which x, y and z are the coordinates of the node, `touch` is the identification of the element contacted. The `o_s` and `i-s` represent respectively the outer and inner surface boundary on half the cross section of a hollow cylinder. These can be calculated according to the diameter, wall thickness and vertical deformation ratio of the element (Figure 3-23).

The definition of a cylindrical BDE is

```
class CylinderBDE
{
  public:
    void Locate(...);
    void Store(...);
    ...
  private:
    int  id;
    CylinderNodeType (*node) [MAX-SZ];
```

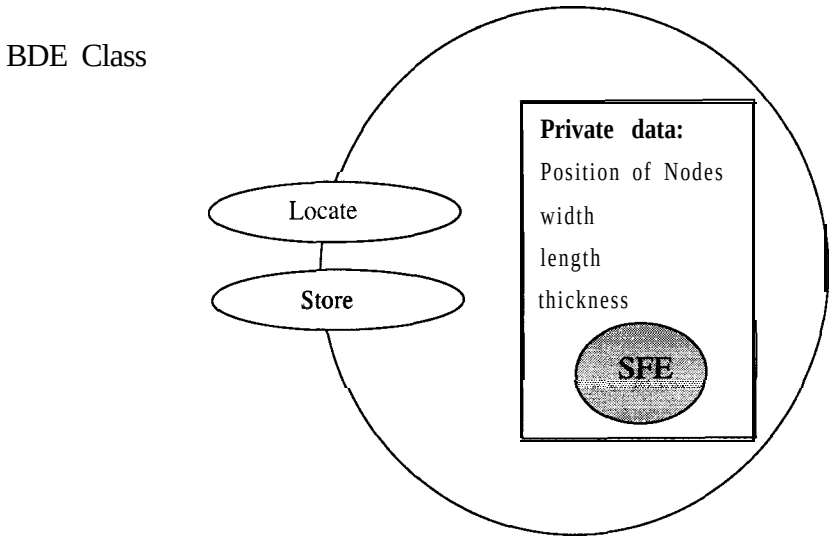


Figure 3-22. Design of 3D Body of an Element (BDE).

```
  int w;
  int l;
  int h;
  SFE surface;
};
```

in which `h` represents half of the maximum height (thickness) of the element (Figure 3-23). `id` is the identification of the element; 1, w are respectively the length and width of the element.

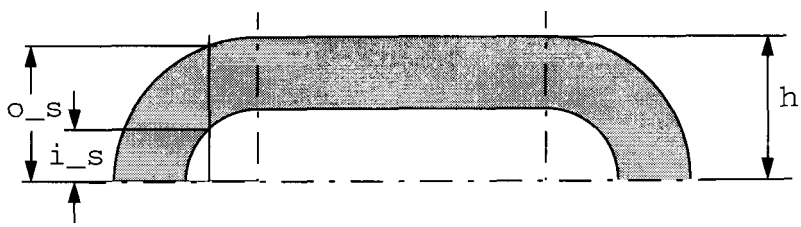


Figure 3-23. Parameters for describing the half cross section of a cylindrical element.

For a cuboid element, the node is

```
struct CuboidNodeType
{
   int x;
   int y;
   int z;
   touch;
};.
```

The definition of the BDE is

```
class CuboidBDE
{
  public:
    void Locate(...);
    void Store(...);
    ...
  private:
    int id;
    CuboidNodeType (*node) [MAX_SZ];
    int w;
    int 1;
    int t;
    SFE surface;
  };
```

in which `id` is the identification of the element; 1, w and t are respectively the length, width and thickness of the element.

The member function `Locate ()` in the class determines the variables on each node. The horizontal position (x and y) and contact information `(touch)` of a node can be copied directly from the correspondent node on SFE. The vertical position (z) is calculated based on the vertical distance between the node and SFE (Figure 3-24).

The operation by function `Store ()` is to write all the information about the element into a file that represents the entire simulated structure. The format of the file is shown in

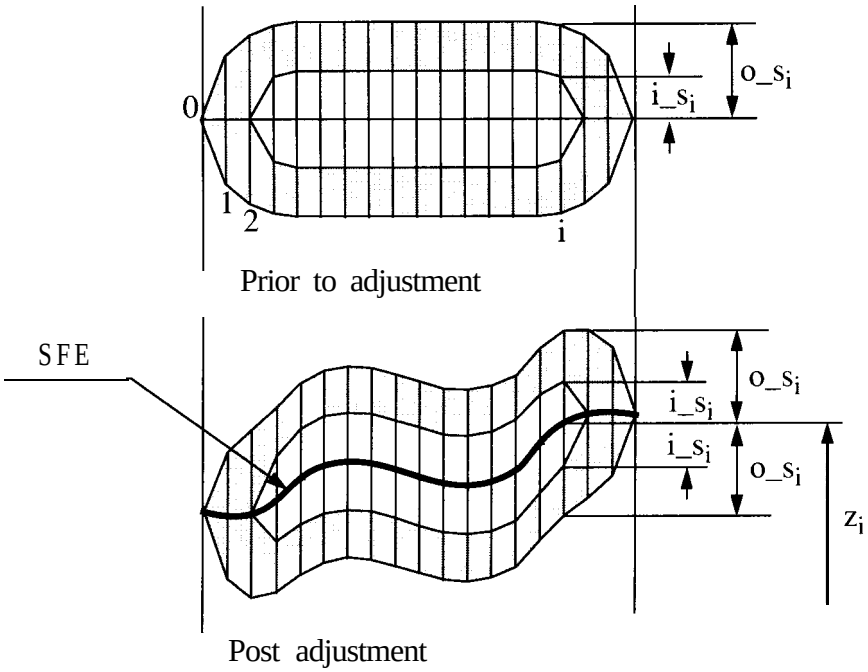


Figure 3-24. Describing deformed cross section of a cylindrical element.

Figure 3-25. The format is specially designed for a graphical software, Tecplot, which is used to visualize the virtual composite structures.

Another operation performed by `Store ()` is to update SFS with the upper surface of the element:

```
SFS[I][J].height = SFS[I][J].height+node[i][j].o_s;
SFS[I][J].id = node[i][j].id;
```

in which `I= node[i][j].x and J= node[i]` [j].y (Figure 3-26).

```
ZONE T="FIBER 86"                                  ──► Element Identification (id)
I=10, J=98, K=4, F=POINT                           ──► 1, w, 2h
DT=(SINGLE SINGLE SINGLE SINGLE)
19731300                                           ──► x, y, z, touch
198 313 0 0
I9931300
20031300
...
207223277
208 223 2 77
209 223 2 77
210223 2 77
201222276
202 222 3 0
...
ZONE T="FIBER 87"
I= 10, J=96, K=4, F=POINT
DT=(SINGLE SINGLE SINGLE SINGLE)
215218277
215217277
216217277
21721720
...
```

Figure 3-25. Sample data file for storing elements.

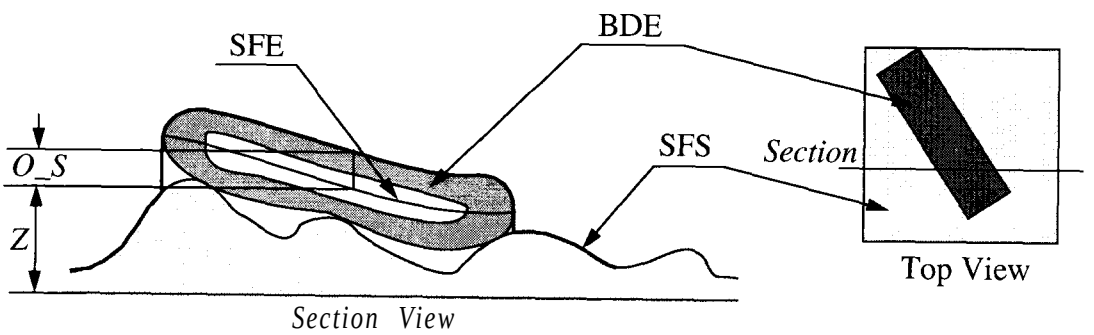


Figure 3-26. Deformed cross section of a cylindrical element.

3.2.4. Simulation Tool - VComp

Similar to laboratory procedures of manufacture and evaluation of composite specimens, the simulation process includes two major functions, to Build a virtual composite structure and then *Test* the simulated structure. The correspondent components are designed and implemented in *VComp*, the software for the simulation process (Figure 3-27).

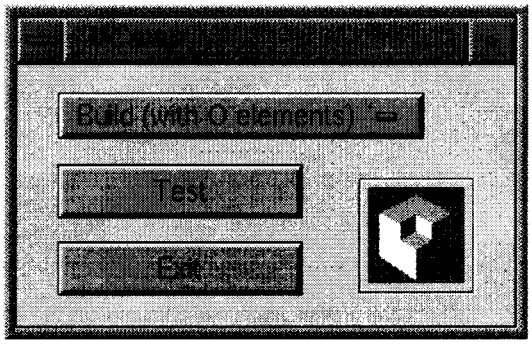


Figure 3-27. The main panel of *VComp*.

3.2.4.1. Building Process

VComp is designed to *Build* two types of composite structures according to different geometries of composite elements, specifically with cylindrical elements, which can be used to simulate fiber based composites, such as fiberboard, MDF and paper, and cuboid elements, which is suitable for simulation of flakeboard, OSB, etc.

The *Building* process consists of three major objects, Control Panel, Generator and Depositor. The relationship among these three objects are shown in Figure 3-28.

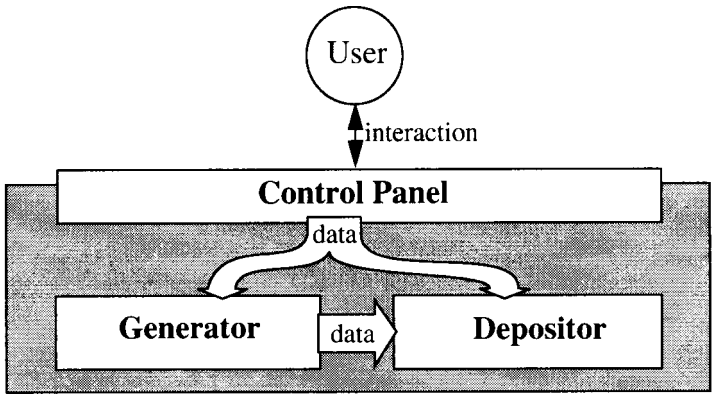


Figure 3-28. Program to *Build* composite structures.

Object 1. Control Panel

Control Panel allows users set up simulation conditions and input necessary parameters (Figure 3-29 and Figure 3-30). All the input parameters shown on the two panels are either self-explained or have been expained in the previous sections.

The operations involved are mainly Graphic User Interface (GUI) functions which will not be listed here. The ViewKit, a GUI library, and RapidApp, a GUI building tool, supplied by Silicon Graphics, were employed in the programming.

Object 2. Generator

The Generator outputs the basic geometrical properties described in Section 3.2.3.1 for each element, specifically, the length, diameter and cell wall thickness for a hollow cylinder or the length, width and thickness for a cuboid. Each of these properties is a random variable within certain probabilistic distributions. The input information for Generator includes the parameters of these distributions (means and standard deviations). The major

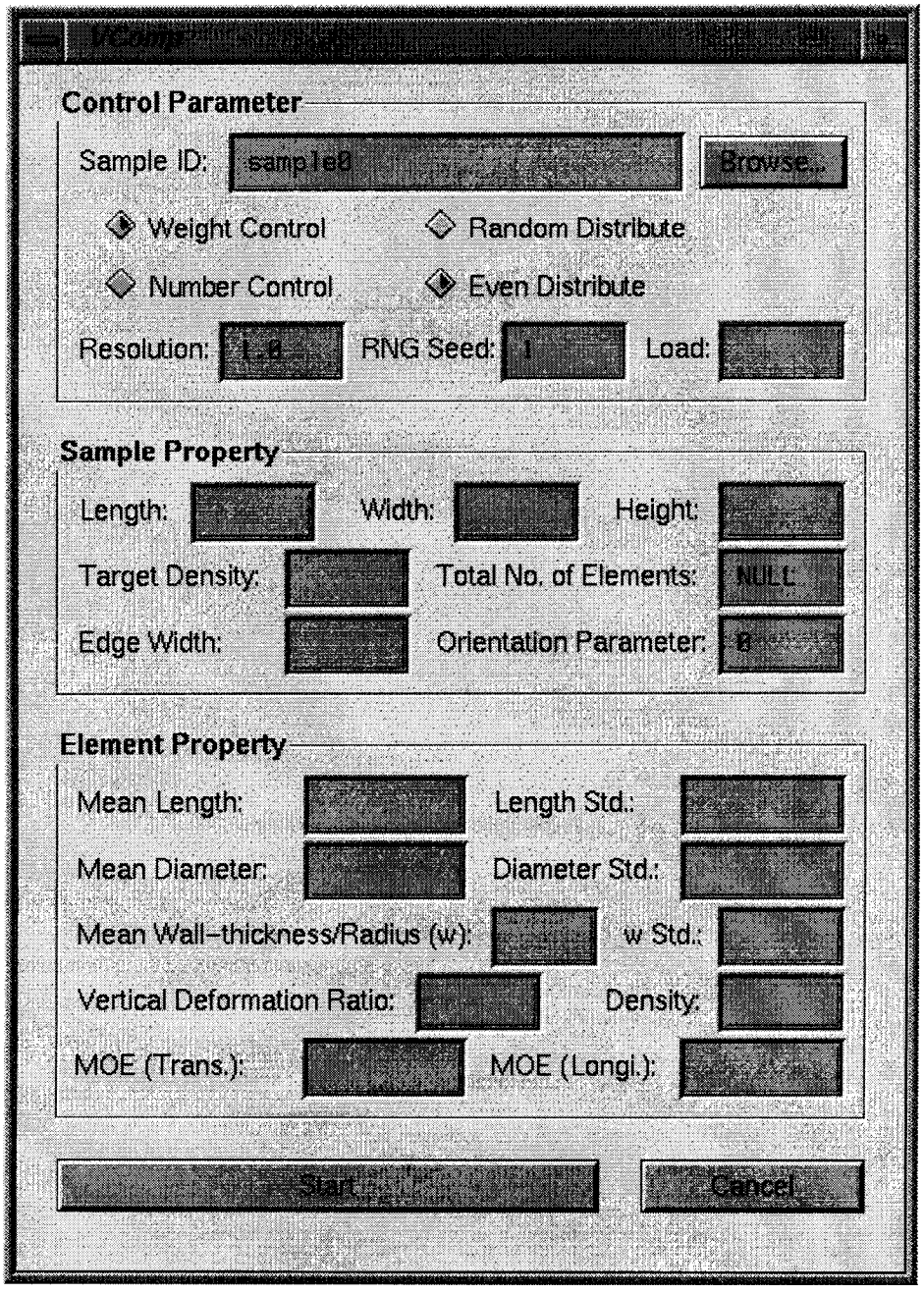


Figure 3-29. The control panel of Building process with cylindric elements.

activity of the object is to generate random variables. Normal or lognormal deviates are output based on Monte Carlo simulation (Section 3.1.2).

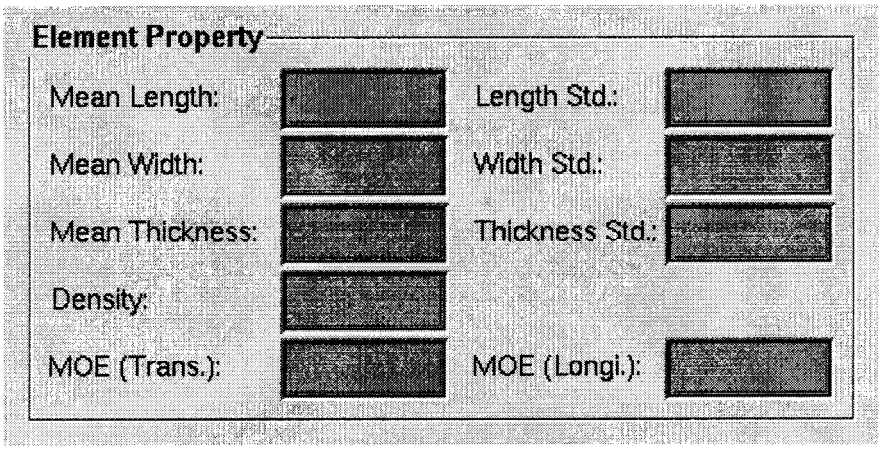


Figure 3-30. Part of the control panel of *Building* process with rectangular elements.

Object 3. Depositor

The Depositor determines the morphology of the element. All the parameters shown on Control Panels, except for the Element Properties, are included in the input information to Depositor. The output of Generator is part of the input to Depositor. The main operation of Depositor has already been elaborated through the evolution of an element in Section 3.2.3.2.

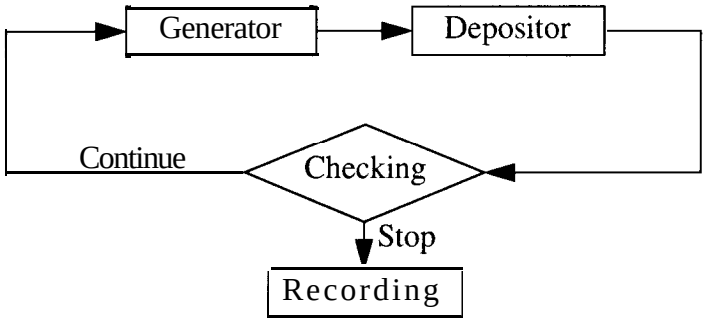


Figure 3-31. Processing line of *Building* a virtual composite structure

The processing line of Building a virtual composite structure is diagrammed in Figure 3-31. Generator sequentially produces elements with random dimensional properties. The elements are then transferred to Depositor. Depositor drops the elements into *Sample Space* and also identifies the morphology of the elements and stores the information in a data file. The weight of each element is measured. If *Weight Control* is chosen on the Control Panel, after each element is generated and deposited, the total weight of the structure assembly is compared with the final desired weight. If the value is greater than the desired value, Generator and Depositor will stop the operations. If *Number Control* is chosen, Generator and Depositor will quit when the total number of elements generated reaches the desired number. The input information, total number of elements in the structure, maximum height of the structure will be recorded in a file. The final status of the top surface of structure (SFS) will also be dumped into a data file.

3.2.4.2. Testing Process

In the *Test*ing Process, the composite structure produced in the *Building* Process is divided into smaller specimens. The void fraction of each specimen is calculated and the coverage, which represents the number of elements at a horizontal point within the composite structure, is counted at one corner of each specimen. In addition, virtual slices may be produced in the process. Two major objects are involved in this process, Control Panel and Tester.

Object 1. Control Panel

The Control Panel allows users to choose a simulated composite structure to test by inputting the identification, file name, and set up test conditions which include the location of

the specimens in the original structure and the dimensions of the specimens. By inputting the data file name, the properties of a simulated composite structure can be retrieved and displayed on the Control Panel (Figure 3-32).

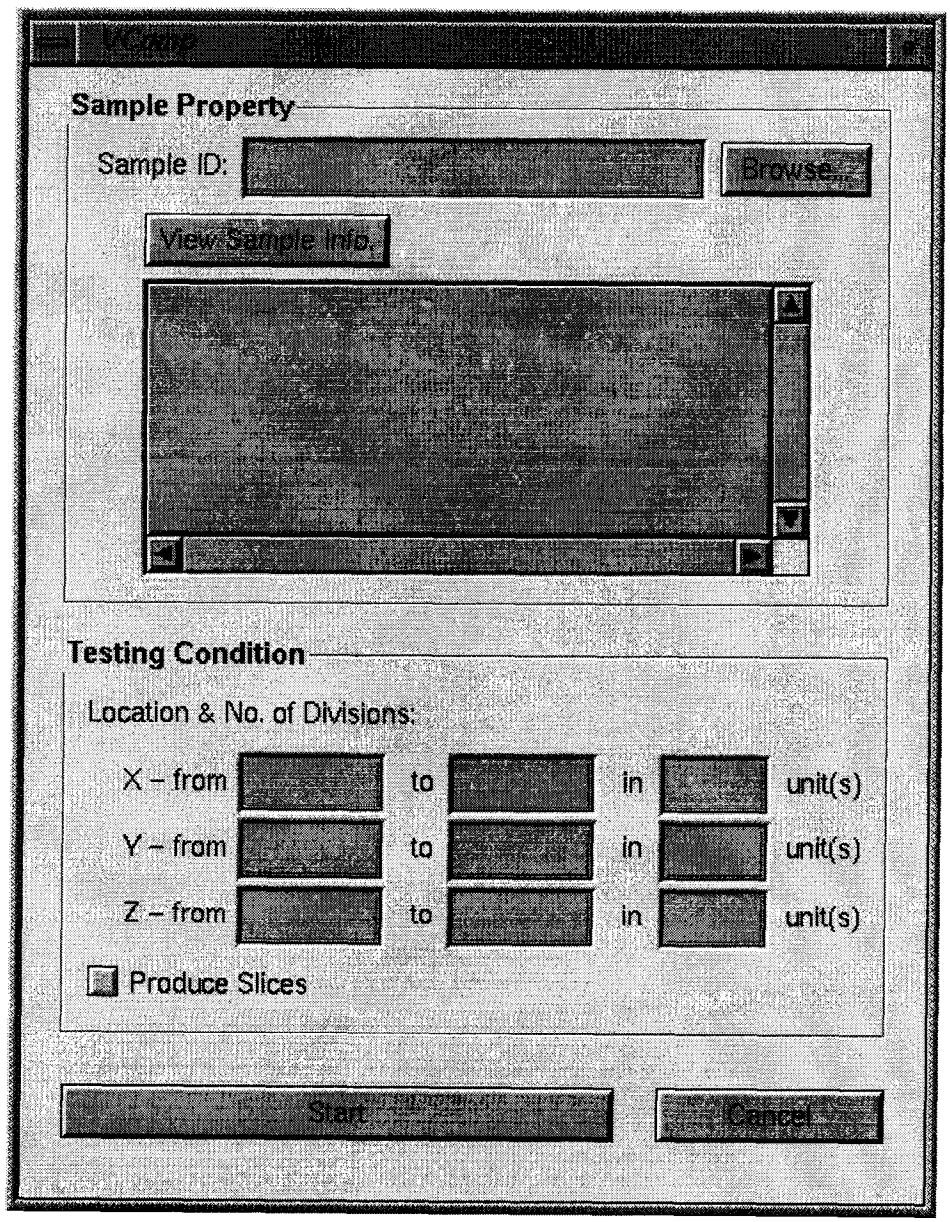


Figure 3-32. The control panel of testing process.

Object 2. Tester

According to the testing conditions set up through Control Panel, Tester takes the following steps to test a virtual structure.

Step 1. Load the elements and reconstruct the volumetric structure.

Step 2. Load the surface of structure.

Step 3. Divide the reconstructed structure into specimens.

Step 4. Divide the surface into specimen-size units.

Step 5. Calculate and output material volume and bonded area in each specimen.

Step 6. Output coverage and height of each unit surface.

Step 7. Produce slices (optional).

At Step 1. Tester reads the morphological data of each element from the file generated in the Building process. The data are then used to reconstruct the volumetric structure. The reconstruction is necessary because in the data file each element is recorded according to its local coordinate system position. After reconstruction, all elements are merged into one 3D buffer. The 3D buffer is designed as a stack of slices (SLC). Each slice is a 2D dynamic array. Each element of the array is a node that records the material status of the virtual structure at this location.

```
struct  SLCNodeType
{
  unsigned short id;
  unsigned short touch;
};


struct SLCType
{
  SLCNodeType (*node) [MAX_SZ];
};
```

A dynamic array of pointers is declared during the execution time.

```
SLCType *slice;
```

Each pointer points to one slice. By doing this, the 3D buffer has two adjustable dimensions, the length and height, so that the limit of stack size and wasted memory are minimized. The buffer is initialized with nested `for` loops,

```
slice=new SLCType[zsize];
for(k=0; k<zsize; k++)
{
  slice[k].node = new SLCNodeType[xsize] [MAX-SZ];
  for(i=0; i<xsize; i++)
    for(j=0; j<ysize; j++)
      {
        slice[k].node[i][j].id=0;
        slice[k].node[i][j].touch=0;
      }
}.
```

in which `i, j` and `k` are loop counters.

The reconstruction is implemented with the following two lines,

```
slice[z].node[x][y].id=ID;
slice[z].node[x][y].touch=TOUCH;
```

in which x, `y` and z are the coordinates of each node within a BDE, which are read from the first three columns of the data file, `ID` is the identification of the BDE, and `TOUCH` is identification of the BDE contacted, which is the fourth column in the data file (Figure 3-25).

At Step 2, the SFS data structure is used. At Step 3 and Step 4, both the stack of slices and SFS are divided horizontally according to the input horizontal size of the specimens (Figure 3-33).

At Step 5, the number of nodes with the property

```
slice[z].node[x][y].id!=0;
```

and the number of nodes with the property

```
slice[z].node[x][y].touch!=0;
```

are recorded for each specimen. If these two numbers are denoted as N_s and N_b respectively, the material volume, V_s, of a specimen will be

$$V_s = N_s \cdot \rho^3 \qquad (3\text{-}28)$$

and the bonded area, β, in the specimen will be

$$\beta = N_b \cdot \rho^2 \qquad (3\text{-}29)$$

where ρ is the resolution of simulation (ROS).

The void fraction, ϖ, can be estimated as

$$\varpi = \frac{N - N_s}{N} \qquad (3\text{-}30)$$

in which N is the total number of nodes in a specimen.

Coverage on any point within SFS can be obtained directly from the SFS data file. However, for convenience of later model evaluation experiments, Tester divides the SFS into rectangular units and records the value at one corner as the coverage (e.g. circled position

in Figure 3-33) at Step 6. Therefore each unit has one coverage value. The height of a unit surface is calculated as the arithmetic mean of heights of all the points on the surface.

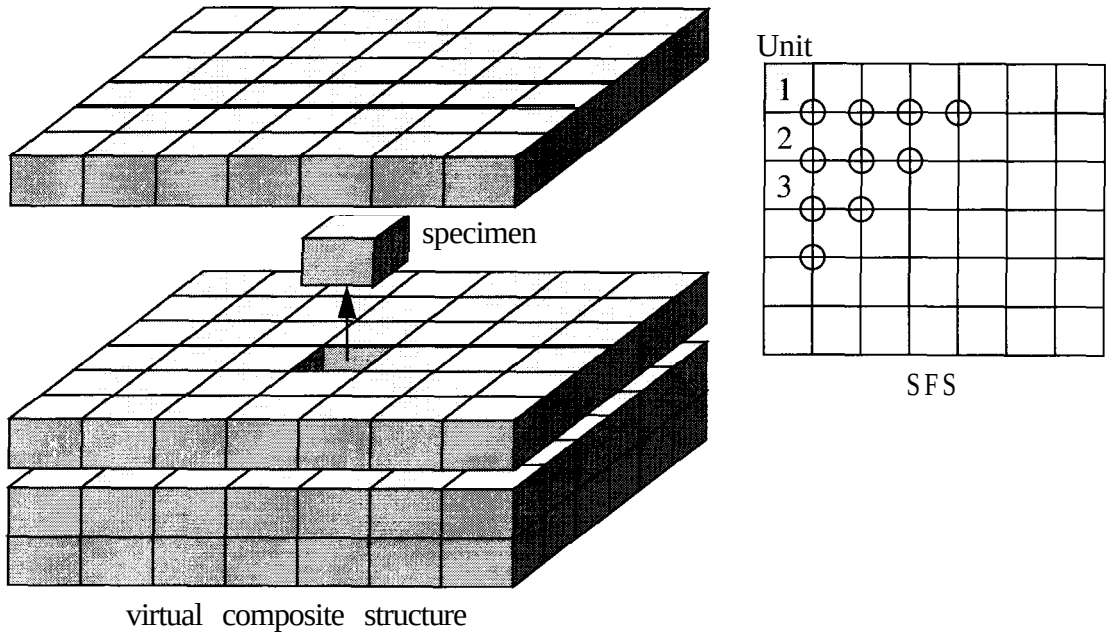


Figure 3-33. Operations in Testing Process.

Step 7 is optional. If Produce ***Slices*** is chosen on the Control Panel, Tester will sequentially dump slices into sequentially named files. On each slice, 0 represents void area and 1 (or 255 for S-bit color, or value of each element's identification) can be used to represent element occupied area. These files can be loaded by other software as input data for further simulation, analysis or visualization.

4 EXPERIMENTAL VERIFICATION

The reliability and reasonability of the simulation methodology rely on whether VComp can produce the digital or virtual microstructures similar to those of the real wood particulate composite materials. To verify that, two experiments were conducted, which are elaborated in the following sections. Both experiments were based on the hypothesis that similar elements and manufacturing conditions would result in similar structures. The similarity was judged based upon the statistical comparison of the microstructural features of the virtual composites generated with VComp and those of the structures made from real materials.

4.1. Verification with Hand-made Composite Structures

To evaluate the performance of VComp, the ideal way would consist of the following steps:

Step 1. Obtain the properties of the constituent elements, wood fibers or flakes.

Step 2. Produce the wood composite samples under precisely measured and controlled conditions,

Step 3. Generate the virtual composite structures with the input parameters representing the properties of real wood fibers or flakes, and the manufacturing conditions.

Step 4. Statistically compare the simulated microstructural features with those of the real wood composite samples.

Obviously, commercial composite products are not suitable because information about the elements and the manufacture is usually unknown. Making wood composites in this ideal way is impractical since the process, especially for wood fibrous composites, involves complicated equipment, instrument and technology and also consumes labor and time resources. Furthermore, the procedure is unnecessary, too. VComp is not restricted to wood materials. According to its design, any long, thin, cylindrical or rectangular and flexible particles can be simulated. This makes it possible to simplify the verification experiment.

Therefore, the following three major steps were taken in the experiment:

Step 1. Construct and test composite structures made with real (non-digital), easy-handling and easy-measuring material.

Step 2. Generate and test virtual composite structures under same conditions.

Step 3. Statistically analyze the testing results from hand-made structures and virtual structures.

4.1.1. Hand-making Composite Structures

4.1.1.1. Materials

Double sided mounting foam tape was chosen to substitute for wood fibers or wood flakes. The tape was made of polyurethane foam and originally came in rolls with the dimension

144” x 1” x 0.125”. Non-drying instant (unknown) adhesive covers both of the 1” wide surfaces. The adhesive bond between two pieces of tape was strong enough to prevent the relative movement. The density of the tape was measured as 0.0108 $lb./inch^3$. The tape was flexible. The modulus of elasticity (MOE) was estimated as 33.5 *psi* through a tensile test. The base material (excluding the adhesive surface) was assumed homogeneous. A similar experiment was conducted by Dodson (Deng and Dodson, 1994), in which rubber bands were used as the elements. Extra rubber cement had to be applied, which made the process less controllable.

For observing the possible influence of the element dimensions and testing the generality of the simulation procedure, three types of composite elements were prepared employing different widths as shown in Table 4-1.

Table 4-1. Dimensions of elements for hand-made composite structures

(unit: inch)

Dimension	**Type I**	**Type II**	**Type III**
length	5 (0)[a]	5 (0)	5 (0)
width	1.0 (0)	0.5003 (0.0262)	0.3387 (0.0223)
thickness	0.125 (0)	0.125 (0)	0.125 (0)

a. Numbers outside and inside the parentheses are the values of the mean and standard deviation, respectively

The length was assumed constant since the error of the measured length of elements (a0.012”) was small relative to the length. The width of Type I composite elements was the original width of the tape, 1 ". Elements for Type II and Type III composites were hand-cut from the 1" wide tape. To account for the effect of variance, 50 randomly selected ele-

ments from each of the two types were measured. Descriptive statistical analysis was carried out on the two sets of width data first. Two-sided Kolmogorov Goodness of Fit Tests were then conducted respectively against the hypothesized normal distribution functions with the means and standard deviations obtained from the descriptive statistical analysis. The test statistics were 0.080 and 0.177, while the 0.9, 0.95 and 0.99 quantiles of statistic for sample size 50 were 0.172, 0. 192 and 0.23, respectively. Hence, hypothesized distribution functions were accepted as the appropriate descriptions of the element width. The test results are plotted in Figure 4-1.

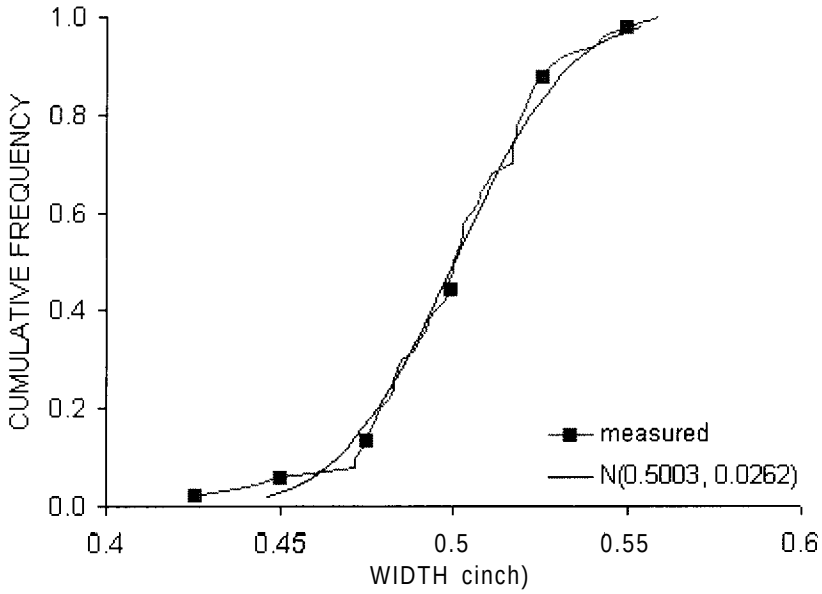
4.1.1.2. Building

To keep the target density constant, the number of elements in each type of structure was calculated according to the dimensions of the elements. The formats of the structures were shown in Table 4-2.

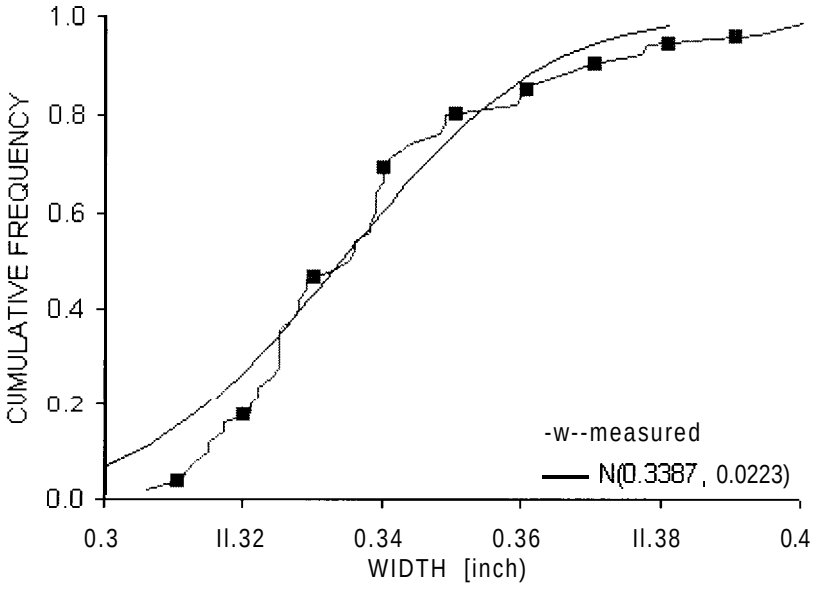
During the forming process, elements were dispersed on a paperboard surrounded by a *Forming* Box (16" x 16") (Figure 4-2). Elements bonded together instantly into a final structure due to contact forces. Extra pressure was not necessary. A total of 15 structures were built, five for each type.

4.1.1.3. Testing

Each structure was then horizontally divided with a razor blade into 64 2" x 2" specimens (Figure 4-3). Each specimen was then weighed. The volume of material in each specimen was calculated based on the weight and the density of tape. The number of elements at one



(a)



(b)

Figure 4- 1. Distributions of widths of elements.

(a) Type II, (b) Type III.

Table 4-2. Properties of hand-made composite structures (unit: inch).

Property	Type I	Type II	Type III
length	16	16	16
width	16	16	16
total No. elements	58	116	174

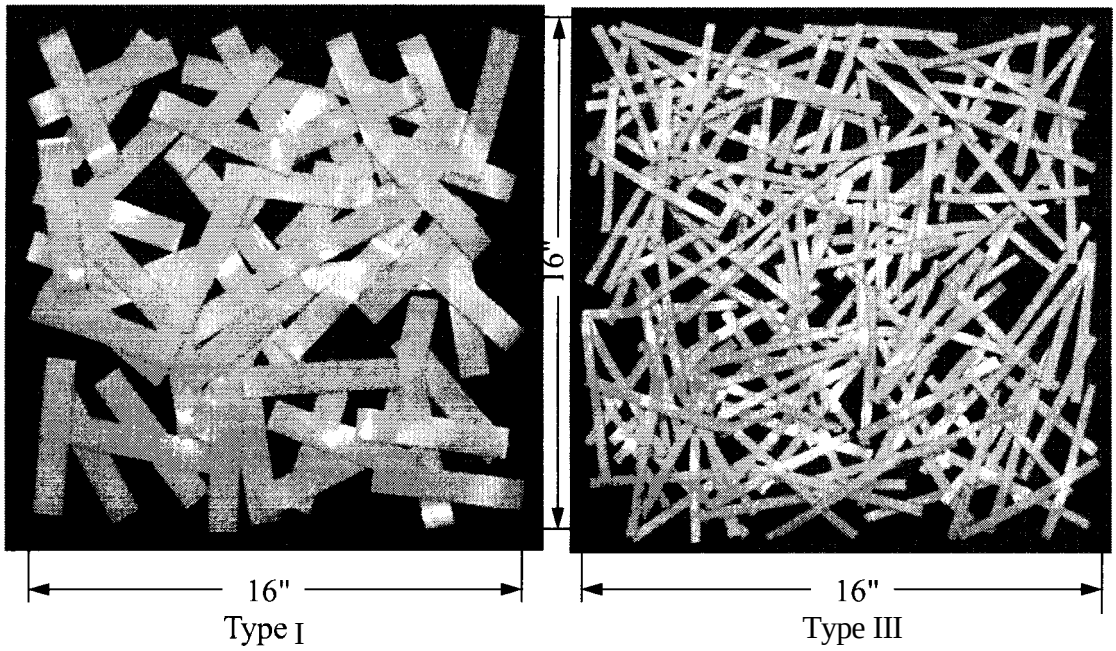


Figure 4-2. Hand-made composite structures.

comer (shaded) (see in Figure 4-3) of each specimen was counted and used to define the values of coverage for that specimen. Comers of specimens located on the periphery of the structure (unshaded) (see in Figure 4-3) were not evaluated due to edge effects. Thus the coverage for each structure was measured at 49 locations. The bonded area and the height of the specimens were not measured due to lacking of efficient measuring method. Therefore, these two properties of simulated structures were not evaluated.

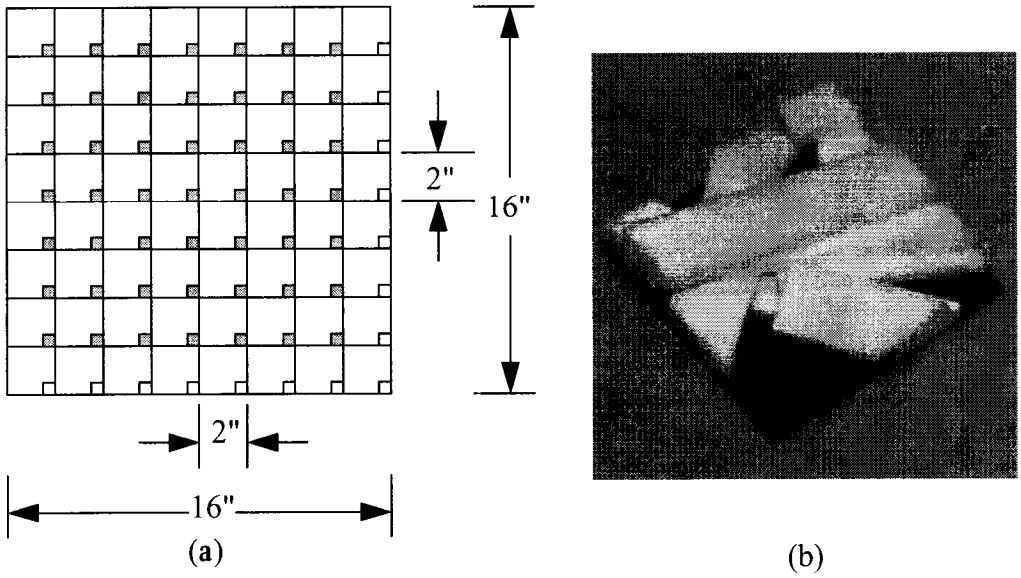


Figure 4-3. (a) Procedure for dividing the composite structure into specimens and (b) one specimen cut from a Type II model.

4.1.2. Making and Testing Virtual Composite Structures

Based on the same conditions shown in Table 4-2 and the material properties stated in Section 4.1.1, five evenly-distributed and five randomly-distributed virtual composite structures were generated for each type. The Resolution of Simulation (ROS) for these structures was 0.02 inch. The virtual structures were then tested with VComp under the same condition as for testing the hand-made structures, which means that the virtual structures were also divided into 2" x 2" specimens and the values of substance volume and coverage were calculated and recorded for each specimen, Figure 4-4 shows a three-dimensional view of a virtual composite structure. All types of virtual composite structures are presented in Figure 4-5.

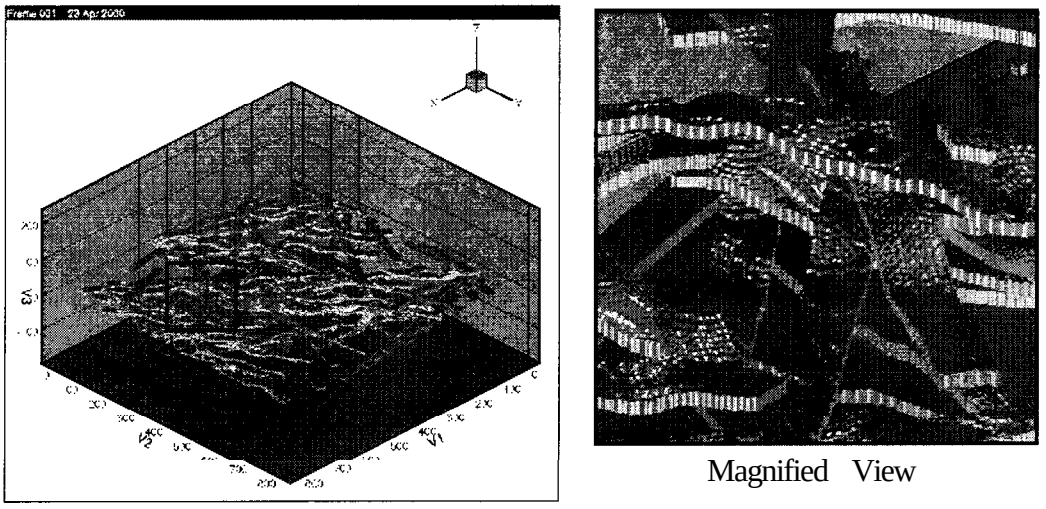


Figure 4-4. A Type II virtual composite structures.

4.1.3. Statistical Comparison

Testing results of the local material volume and local coverage from both the virtual composite structures and hand-made composite structures within each type of formats were statistically compared with one another. A non-parametric statistical method, the Smirnov test, was used since the properties of these data sets were unknown. Based on the difference of the formats and building methods, the composite structures were categorized into nine groups: three types of format in which each type contained structures built in three ways, hand-making, simulation (evenly-distributed) and simulation (randomly distributed) (Table 4-3).

4.1.3.1. Intra-group Comparison

The comparisons were first conducted, respectively, among the five samples of volume data and five samples of coverage data of each group of composite. The two-sided five-sample Smirnov test method was used. The statistics shown in Table 4-3 were all smaller

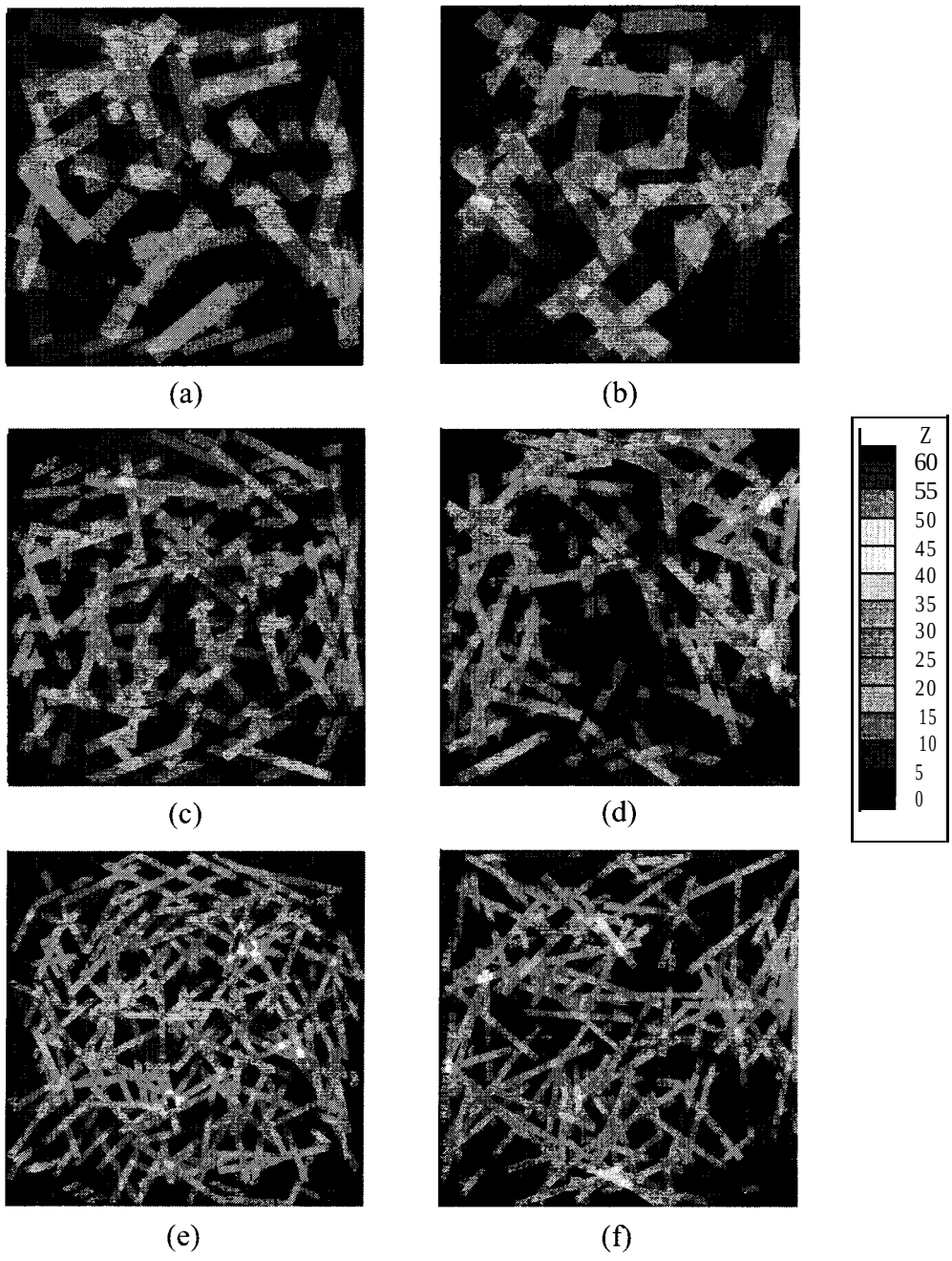


Figure 4-5. Top view of virtual structures simulated by VComp.

(a) Type I - even (b) Type I - random (c) Type II - even (d) Type II - random (e) Type III - even (f) Type III - random. The horizontal dimensions of all structures are 800 x 800, with ROS = 0.02".

Table 4-3. Smimov test statistics from the tests on comparison of different types of composite samples.[a]

	Hand-made		Simulated (even)		Simulated (random)	
	Volume	Coverage	Volume	Coverage	Volume	Coverage
Type I	0.203	0.163	0.141	0.102	0.203	0.122
Type II	0.187	0.163	0.141	0.184	0.156	0.122
Type III	0.172	0.122	0.141	0.204	0.156	0.224

a. The 0.90, 0.95 and 0.99 quantiles of statistic for samples with equal size 64 are 0.19, 0.216 and 0.269, respectively. The 0.90, 0.95 and 0.99 quantiles of statistic for samples with equal size 49 are 0.224, 0.265 and 0.327, respectively.

than the correspondent 0.95 quantiles, which means that the structures within each group were identical. The results indicated that all the sample making methods and procedures were consistent. Figure 4-6 illustrates the consistency of the hand-making method for the Type II composite structures. Figure 4-7 shows the consistency of the simulated Type I composite with randomly distributed elements.

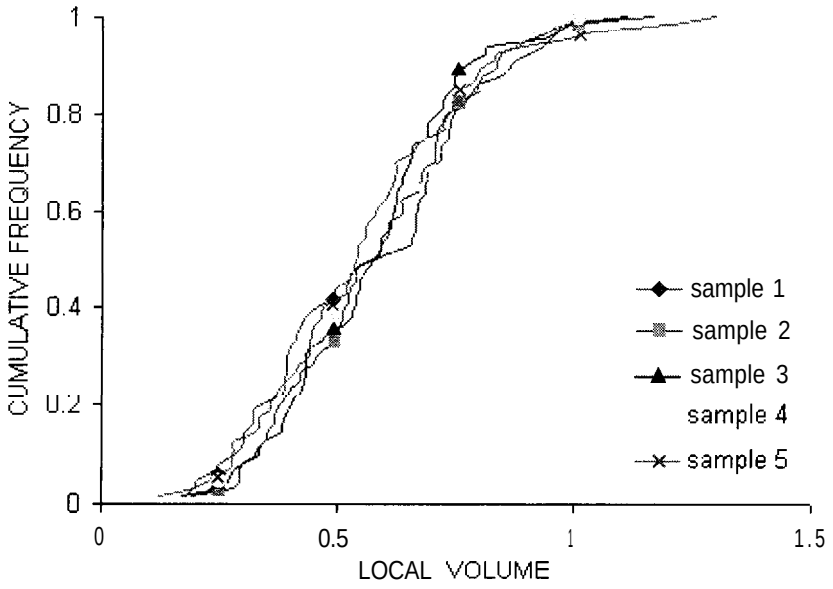


Figure 4-6. Intra-group comparison of volume distributions of hand-made Type II composites.

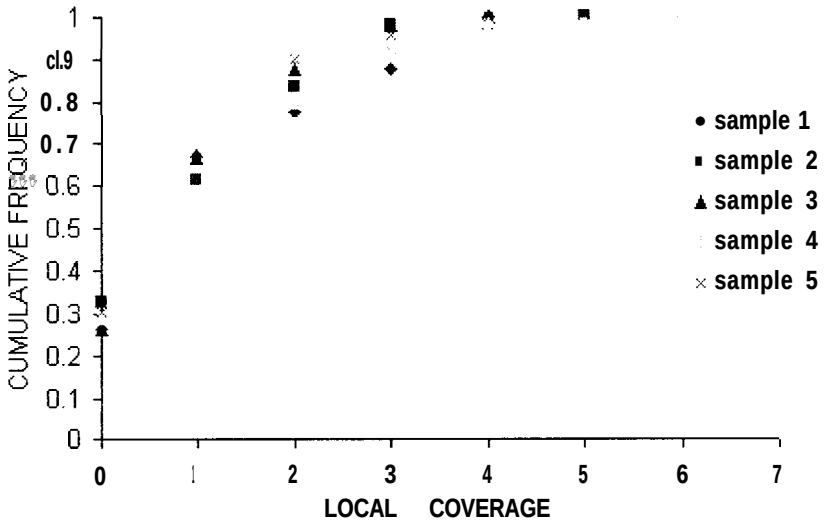


Figure 4-7. Intra-group comparison of local coverage distributions on simulated Type I (*random*ly distributed) composites.

4.1.3.2. Inter-group Comparison

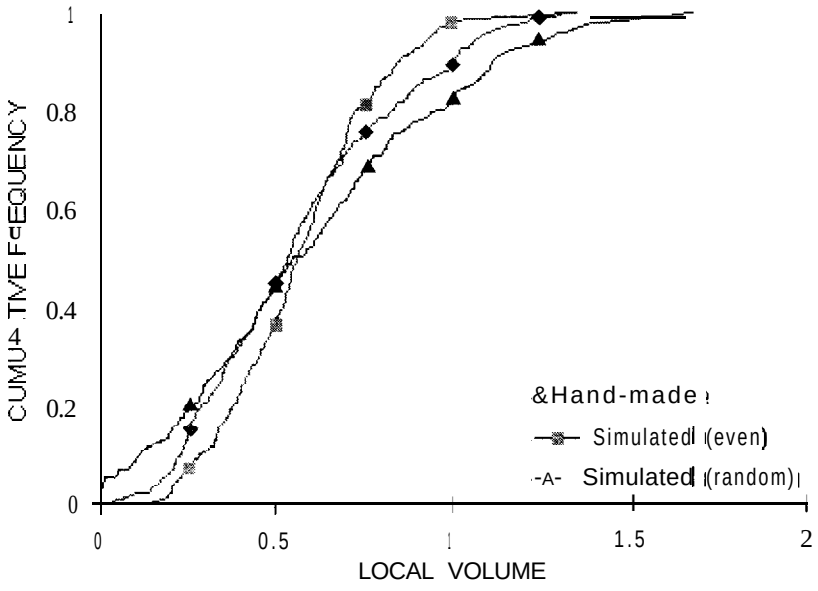
After the consistencies of both the hand-making and the simulation procedures were confirmed, The virtual composite structures were then compared with the hand-made composite structures. The five volume data sets in each group were consolidated into one set with sample size 320, while the five coverage data sets were consolidated into one set with sample size 245. Volume and coverage data sets from hand-made composite structures, evenly-distributed and randomly-distributed virtual composites structures were compared with one another. The null hypothesis was that the three volume populations and three coverage populations had identical distribution functions, separately. Table 4-4 shows the resultant test statistics of two-sided two-sample Smimov tests.

Table 4-4. Smirnov test statistics from the tests on the comparisons between hand-made models and virtual composites.[a]

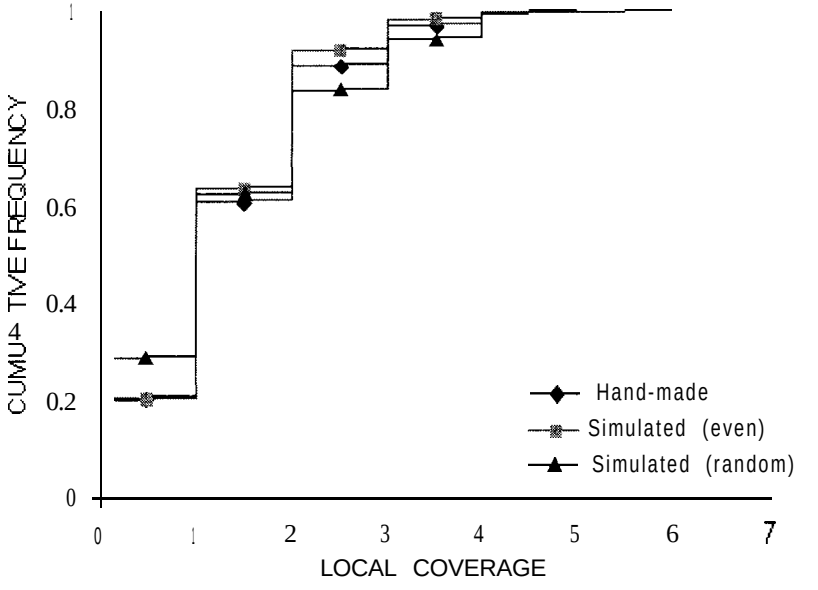
	Hand-made vs. Virtual (even)		Hand-made vs. Virtual (random)		Virtual (even) vs. Virtual (random)	
	Volume	Coverage	Volume	Coverage	Volume	Coverage
Type I	0.116	0.033	0.097	0.086	0.172	0.09
Type II	0.109	0.041	0.100	0.057	0.153	0.061
Type III	0.109	0.045	0.153	0.139	0.144	0.094

a. The 0.90, 0.95 and 0.99 quantiles of statistic for samples with equal size 320 are 0.096, 0.107 and 0.128, respectively. The 0.90, 0.95 and 0.99 quantiles of statistic for samples with equal size 245 are 0.11, 0.122 and 0.147, respectively.

The results showed that the simulated structures of Type I and Type II virtual composites with randomly distributed elements and the correspondent hand-made structures could be considered identical since the tests on both volume data and coverage data did not reject the null hypothesis at 0.05 significance level. The properties of the evenly-distributed Type III virtual composite were fairly close to those of the hand-made composite as the statistic from the test on the volume data was smaller than the 0.99 quantile and the statistic from the test on the coverage data was smaller than the 0.90 quantile. The evenly-distributed structures were different from the randomly-distributed structures since the statistics from the test on the volume were all larger than the 0.99 quantile. By analyzing the plots of empirical distribution functions of these data samples, one may see that the properties of the hand-made structures are generally between those of evenly-distributed structures and randomly-distributed structures as being shown in Figure 4-8, Figure 4-9 and Figure 4-10. The discrepancies are reasonable since the property distributions largely depend on how randomly or how uniformly the element were deposited on the *Sample Area* and it was impossible to obtain either the absolute uniformity or the total randomness with the hand-



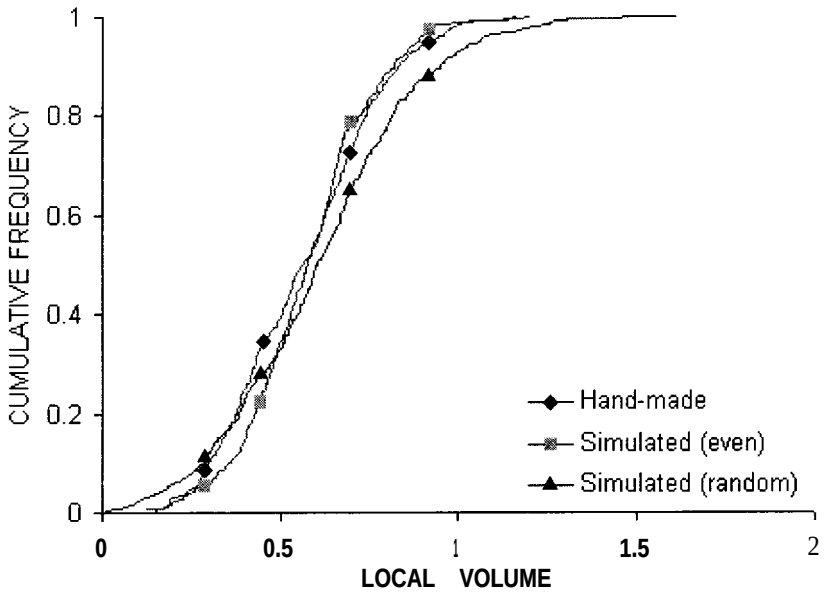
(a)



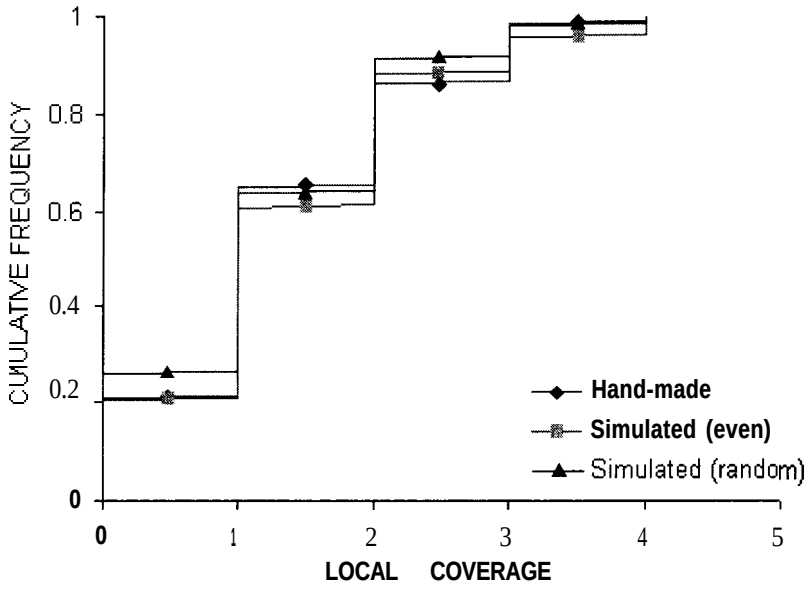
(b)

Figure 4-8. Inter-group comparison of Type I composites.

(a) volume (b) coverage.



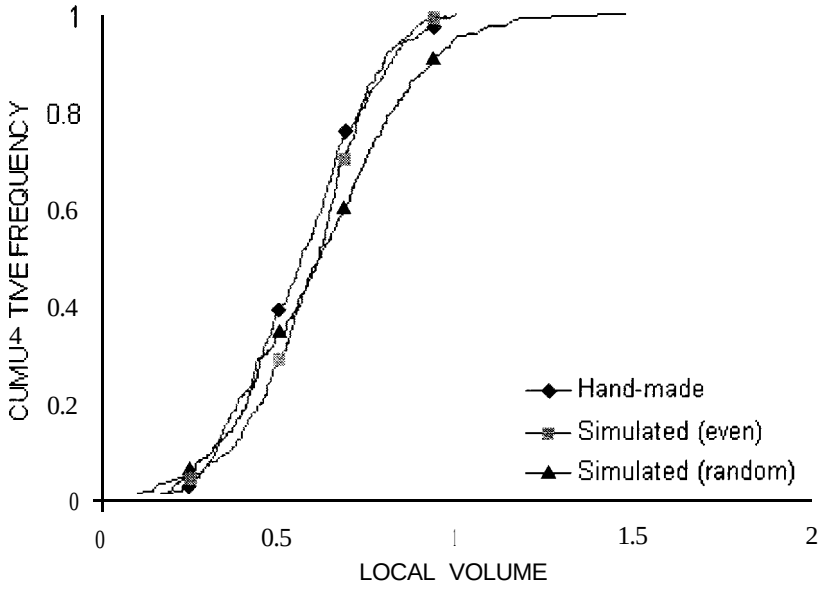
(a)



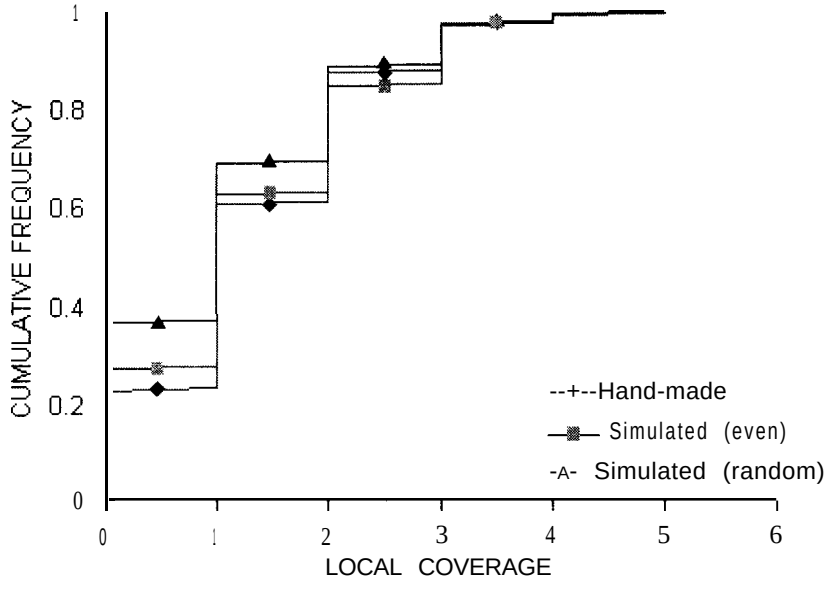
(b)

Figure 4-9. Inter-group comparison of Type II composites.

(a) volume (b) coverage.



(a)



(b)

Figure 4- 10. Inter-group comparison of Type III composites.

(a) volume (b) coverage.

making procedure. This is why robots were used to form composite panels in some experiments (Wang and Lam, 1997).

4.2. Comparison with Kallmes and Corte's 2D Sheet

Kallmes and Corte (1960) made three paper sheets to verify their statistical simulation of paper structures. The published experimental results from sheet No. 1 and sheet No.3 (The independent fiber and sheet properties of sheet No. 2 was identical to those of sheet No. 1.) were used in current research to compare with two simulated structures, respectively. The sheets were portions of 2.5 g/m^2 handsheets made from an unbeaten spruce sulfite pulp. The properties of the fibers and the sheet are listed in Table 4-5.

Table 4-5. Properties of 2D sheets (Kallmes and Corte, 1960).

Property	Sheet No. 1	Sheet No. 3
Mean Fiber Length (cm)	0.292	0.227
Mean Fiber Width (cm)	0.003 1	0.003 1
Mean Curl Ratio	1.1	1.1
Area of Sheet (cm^2)	200	200
Total No. of Fibers	82000	104000

Among several distributions considered in Kallmes and Corte's experiment, the distribution of the number of crossings per *sq. mm* over the sheet for both sheets and distribution of number of fiber segments per *sq. mm* for sheet No.3 were chosen as references. Since VComp is unable to handle 200 cm^2 dimension with a reasonable resolution value, which will be discussed in next chapter, smaller dimensions (2 x 2 cm^2) were used for in the simulation. Fifty specimens were produced for each sheet with the input information shown in Table 4-6.

Table 4-6. Input parameters for simulation of Kallmes and Corte's sheets (1960).

Input Parameter	Sheet No. 1	Sheet No. 3
ROS (mm)	0.002	0.002
Element Length (mm)	2.92 (0.3)[a]	2.27 (0.3)
Element Width (mm)	0.03 1 (0.003)	0.031 (0.003)
Sample Size (mm^2)	2.0 × 2.0	2.0 × 2.0
Width of Edge Zone (mm)	9.5	9.5
Number of Elements	1640	2080

a. Numbers outside and inside the parentheses are the values of the mean and standard deviation, respectively.

Since the values of the standard deviation were not reported in Kallmes and Corte's publishing, estimated values were used in the simulation. Two simulated specimens that respectively represent two extreme cases, most crossings and least elements, are shown in Figure 4-11.

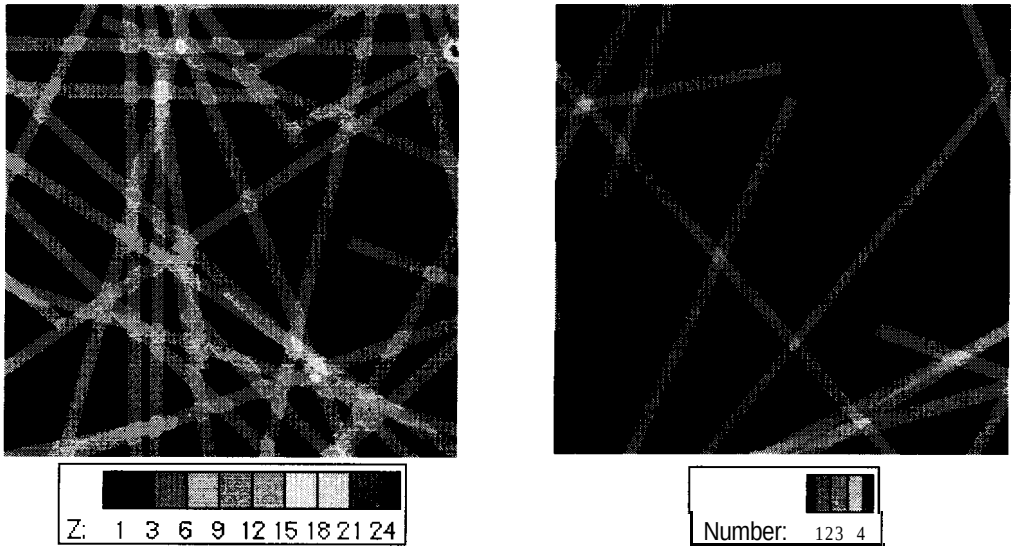


Figure 4- 11. Two simulated sheets

(a) 30 elements, 27 crossings (b) 13 elements, 11 crossings. Sample dimensions: 2.0 x 2.0 mm^2, ROS: 2.0 μm. Z: thickness, Number: coverage.

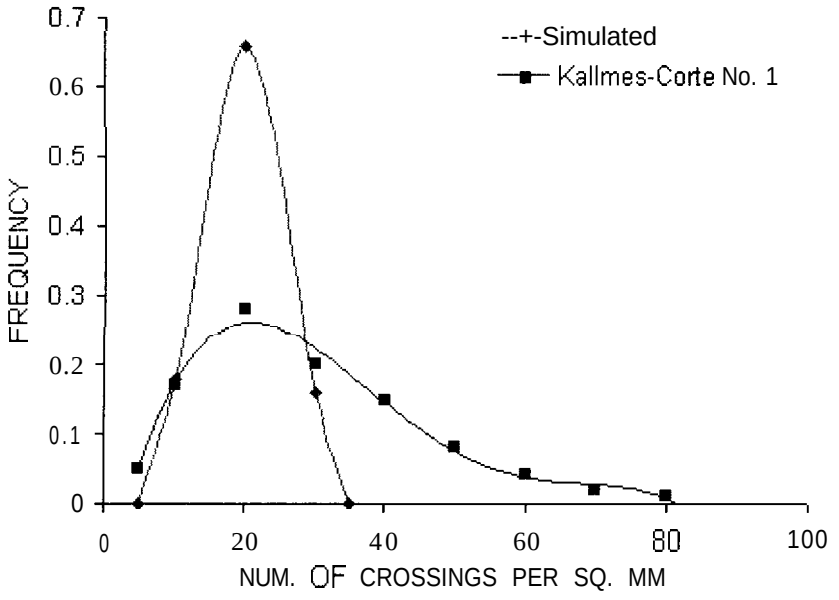
The number of crossings in each simulated specimen was recorded. The distributions of number of crossings per *sq. mm* in simulated structures and the comparisons with Kamlles and Corte's results are shown in Figure 4-12, which indicate that the simulated structures had compatible means but much less variances in terms of number of crossings per *sq. mm*, comparing with the 2D handsheets. It may imply that the real paper structures were much less uniform than the simulated structures due to the flocculation. The significant variances of the handsheets could also be due to the fiber curl. Kallmes and Corte's results have shown that the distribution of number of crossings skew towards larger number of crossings. Two straight fibers can only cross each other once but a curled fiber may cross another fiber once, twice or more times. The curl ratio of the fibers for making the handsheets was 1.1 (average). If a curled fiber formed a segment of a circle, the height of the arc would be

$$h \approx 0.433 l \sqrt{1 - \frac{1}{C^2}} \tag{4-1}$$

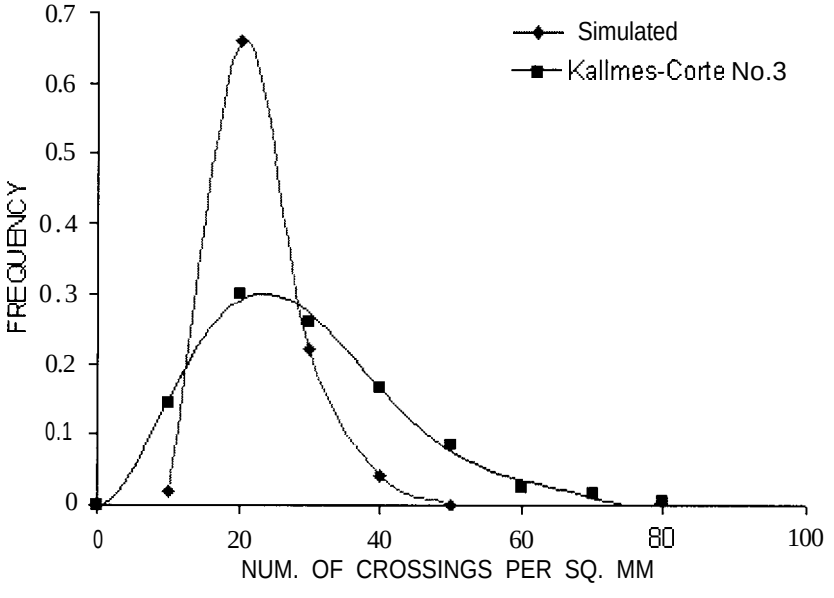
in which C is the fiber curl ratio, and l is the fiber length. Hence, in Kallmes and Corte's experiment, when C was 1.1 and I was 2.5 *mm*, h would be 0.06 *mm*, which was about double that of the mean fiber width. This might increase the chances of crossing.

The predicted distribution of number of fiber segments per *sq. mm* was also tested against Kallmes and Corte's data. The comparison is shown in Figure 4-13. Only the result of sheet No. 3 was reported in the publishing. The mean number of fiber segments per *sq. mm* was 14.3 in sheet No. 3, while in the simulated sheets, the value was 23. It is obvious that the

simulated structure was denser than the handsheet according to the data. Intuitively, one may think that the more fiber segments appear in a area, the more crossings will form in that area. This trend has been observed in the simulated structures (Figure 4-14). The contradiction lies on that Kallmes and Corte's handsheet had less fiber segments and more crossings per unit area while VComp simulated samples had more fiber segments and less crossing per unit area.



(a)



(b)

Figure 4- 12. Distributions of number of fiber crossings.

(a) comparing with Kallmes & Corte's sheet No. 1 (b) comparing with Kallmes & Corte's sheet No. 3. Kallmes and Corte's curves were estimated from the published graphs in the published article (Kallmes and Corte, 1960).

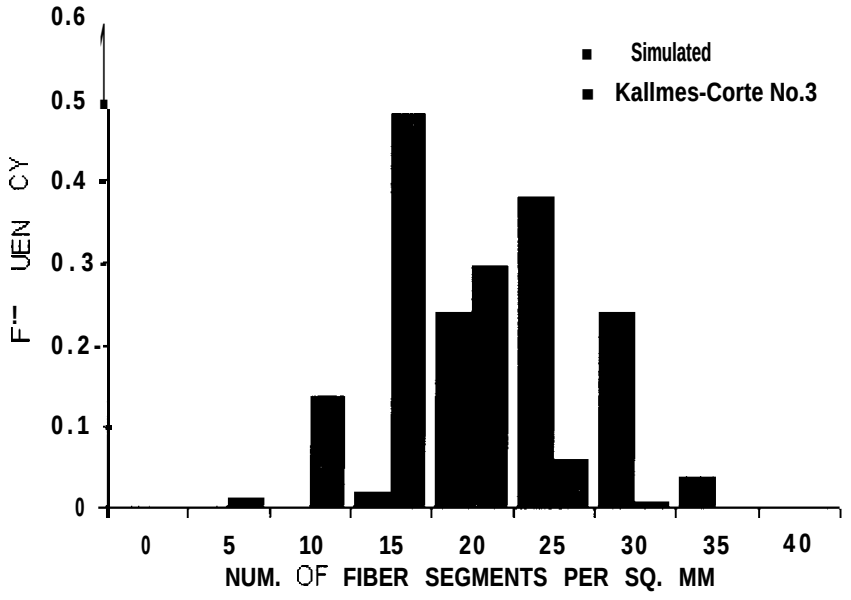


Figure 4-13. Distributions of number of fiber segments.

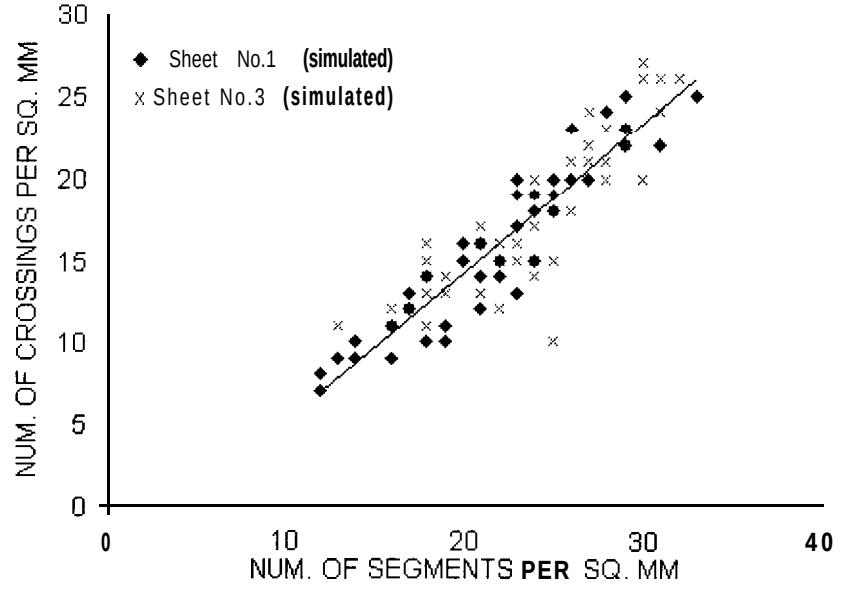


Figure 4- 14. Linear relationship between number of fiber segments and number fiber crossings.

5 EVALUATION OF SIMULATION

In this chapter, the distributional properties of the virtual composite structures and potential effects of the input parameters on the properties are discussed. The purpose of this is to further evaluate the reliability, and also the capability and limitations of the simulation procedure. Several simulations were performed. The physical and mechanical properties, such as the density and modulus of elasticity, of the foam tape used in the verification experiment were used in most of the simulation processes since the correspondent simulation conditions for the material had been experimentally tested. Although it is only a special case, it may represent a large variety of materials and particle geometries. A broader range of simulation will be possible once the relevant properties of wood fibers, flakes and other particles have been elucidated. All virtual structures were randomly-distributed structures since this was more realistic. English units were used where necessary in all the simulations.

5.1. Statistical Description of Microstructural Properties

Statistical descriptions were based on the evaluation of the properties of a virtual composite structure, which included the distributions of local material volume, coverage, volume density and the relative bonded area (RBA). The data sets about the properties were tested against the hypothesized probability functions with the two-sided Kolmogorov goodness

of fit test method. The input parameters for the simulation are listed in Table 5-l. The dimensions of the elements were constant. Therefore, the input standard deviations were all zero. Each virtual structure was horizontally divided into 36 testing specimens. The coverage, average height (or thickness), material volume and bonded area were recorded for each specimen, Therefore, four data samples, each with sample size of 36, were obtained.

Table 5-l. Simulation conditions of the virtual structures for describing distributional properties of microstructure.

Parameter	Value
ROS	0.02
Element Length	5.0
Element Width	0.2
Element Thickness	0.1
Orientation Concentration	0
Dimensions of Structure	30 × 30
Width of Edge Zone	10
Total No. of Elements	6000
Dimensions of Testing Specimen	**5x5**

5.1.1. Local Coverage

The local coverage was tested against a Poisson distribution function, with the rate determined based on a descriptive statistical analysis on the data. The cumulative frequency of the coverage and the cumulative probability density function (cdf) are plotted in Figure 5-1. The statistic of the Kolmogorov goodness of tit test was 0.173. This did not exceed the 0.80, 0.90 and 0.95 quantiles for a two-sided test which were 0.174, 0.199 and 0.22 1, respectively (Conover, 1971). It indicated that the local coverage could be described with the Poisson distribution.

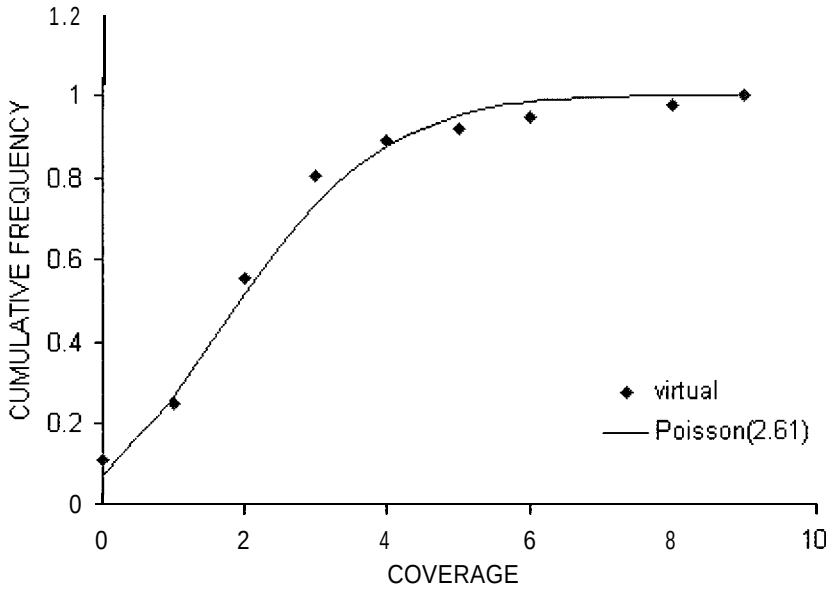


Figure 5-1. Distributions of local coverage of the virtual structure.

5.1.2. Local Material Volume

A test similar to the one on the coverage data was perform on the local material volume data. A normal distribution was predicted. The test result did not reject the hypothesized distribution function since the statistic of the test was 0.094, far below the quantiles. The cumulative frequency of local material volume and the predicted cdf are shown in Figure 5-2.

5.1.3. Local Volume Density

Local volume density is not part of the output of VComp. It can be calculated based on the material volume and the dimensions of each specimen:

$$d = \frac{V}{xy\bar{z}} \tag{5-1}$$

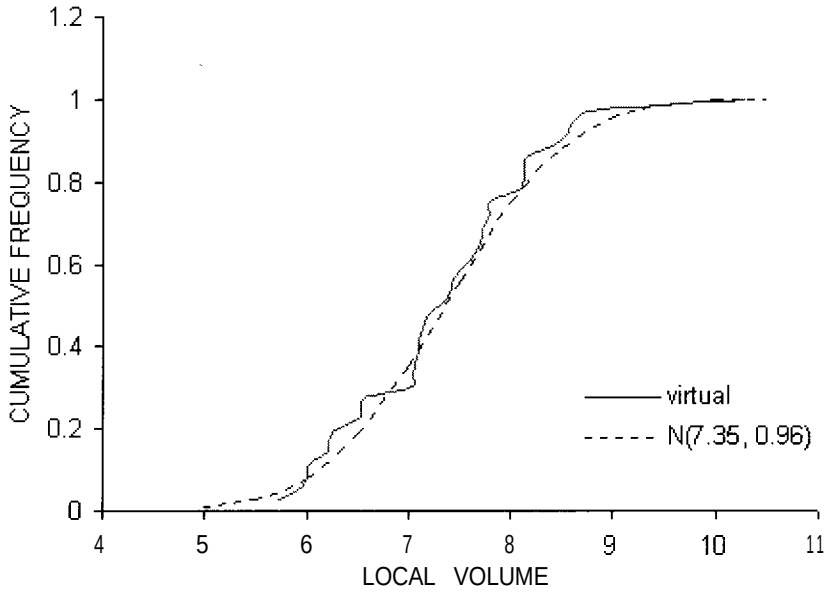


Figure 5-2. Distribution of local material volume of the virtual structure.

where $\boldsymbol{d}$ is the volume density of a specimen, V is the material volume, x and y are the horizontal dimensions of the specimen, which are included in the input parameters, and $\bar{z}$ is the average height of the specimen, which is part of the output of VComp.

The Kolmogorov test statistic yielded from the test on the local volume density data and a predicted normal distribution function was 0.097. Hence the normal distribution function could reasonably represent the volume density distribution of the virtual structure. The comparison of the simulated property and the predicted function are shown in Figure 5-3.

51.4. Local Relative Bonded Area (RBA)

Kallmes and Eckert (1964) defined RBA as the fraction of the external fibrous surface area bonded to other fibers. Based on the data of bonded area in each specimen, which is part

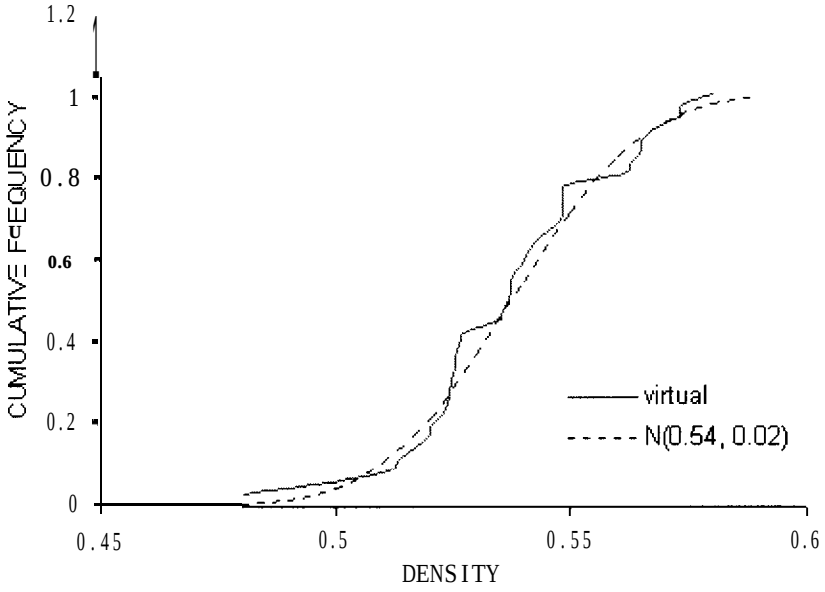


Figure 5-3. Distributions of local volume density of the virtual structure.

of the output of VComp, for cuboid elements, RBA can be estimated with the following equation

$$\xi = \frac{\beta}{\left(\frac{V_s}{T}\right)} \tag{5-2}$$

in which ξ is the relative bonded area (RBA), β is the total bonded area of elements in a specimen, V_s is the material volume of the specimen, and T is the thickness (or mean thickness when the dimension is not constant) of elements. The value of V_s/T would be the estimation of the total bottom surface (or top surface) of elements in the specimen. It is worth noting that only the bonded area at the bottom surface of each element is counted in the simulation procedure.

The Kolmogorov test showed that the distribution of RBA could be described by a normal distribution function (Figure 5-4) since the test statistic, 0.109, did not exceed the quantiles.

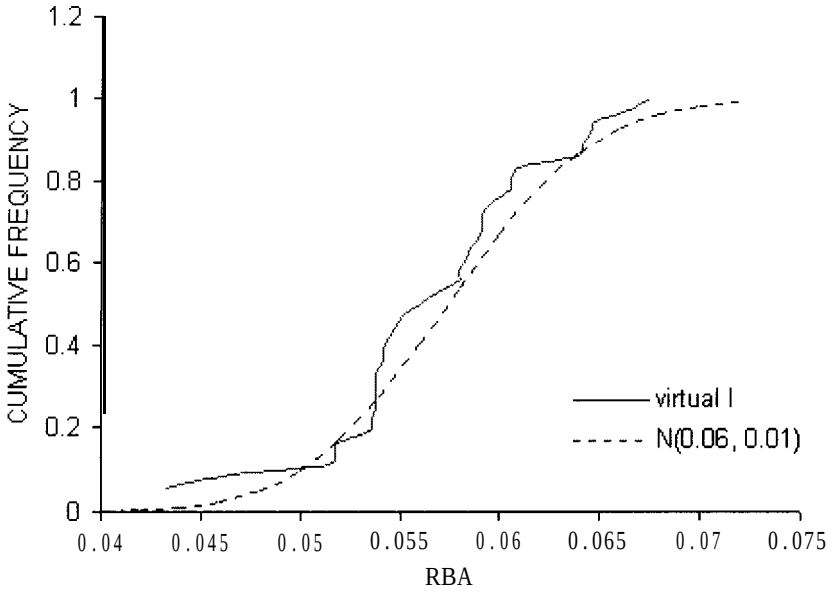


Figure 5-4. Distributions of local RBA of the virtual structure.

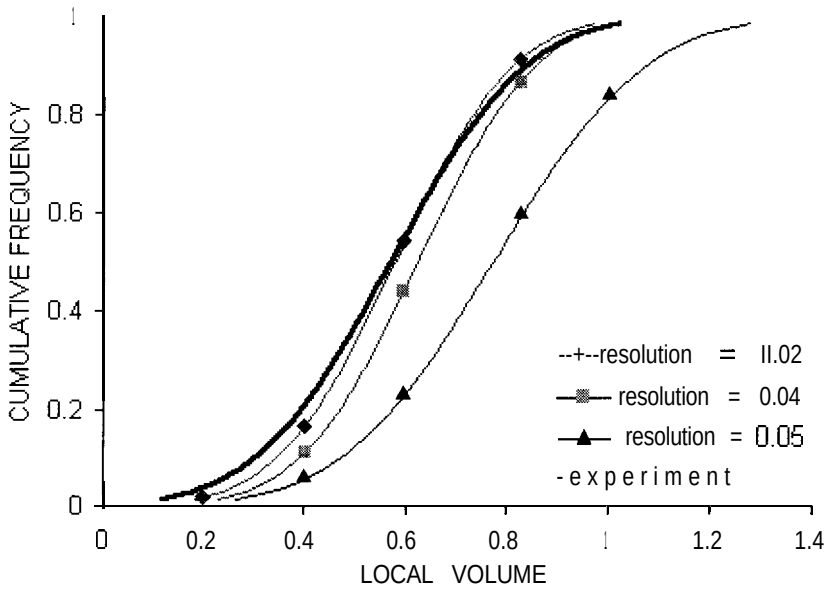
5.2. Parametric Analysis

5.2.1. Resolution of Simulation

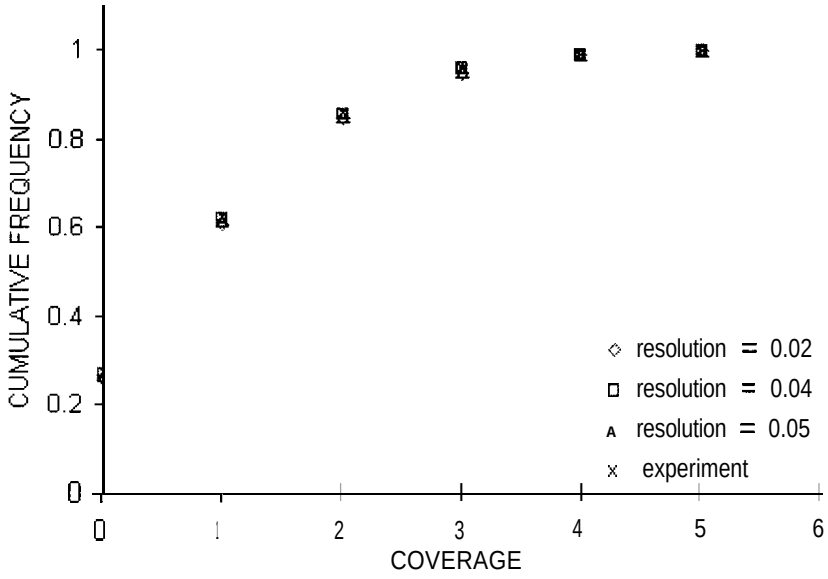
Resolution of simulation (ROS) plays a critical role in current simulation method. It governs the discretization of the 3D sample space and the body of each element. If one of the dimensions of a feature to be simulated is less than the value of the resolution, it will be shown as zero in the simulated structure so that the feature will be invisible. Smaller resolution means finer virtual structures, more visible features and simulated structures that better represent the continuous system. However, when resolution is increased, more nodes

are needed to represent an element and consequently the virtual structure. This will lead to more computational resource usage, which is limited by the capacity of the computer on which the simulation is performed. If the usage exceeds the capacity, the program will crash. The trade-off for higher resolution is the smaller dimensions of the virtual structures.

To demonstrate the influence of resolution to the simulated microstructures, as an example, three ROS values were chosen for the simulation of the Type II tape composites described in Chapter 4. This test was performed before the experimental verification in order to decide an appropriate ROS. Figure 5-5 shows the effect of ROS on distributions of local material volume and local coverage of the simulated microstructures. Comparisons between the properties of the simulated structures and hand-made model structures are also shown in the figure. The results indicated that the resolution had a significant impact on the distribution of the local material volume, but little impact on the coverage. Both the mean and variance of local volume increased with the increase of ROS. These trends are more directly presented in Figure 5-6. The different reactions of the local material volume and local coverage to the effect of ROS are due to the difference between continuous data and discrete data. The data of the local material volume consists of continuous variables while the coverage data consists of discrete variables. The function of ROS is to convert the continuous system into discrete system. The smaller the value of ROS is, the less rounding error will be. The local coverage represents the number of elements above a certain location in the *Sample Area*. The impact of ROS can be observed only when the status of visibility of the elements at that location is changed by the conversion process.



(a)



(b)

Figure 5-5. Effect of resolution on the properties of virtual structures. (a) distribution of local material volume (b) distribution of local coverage.

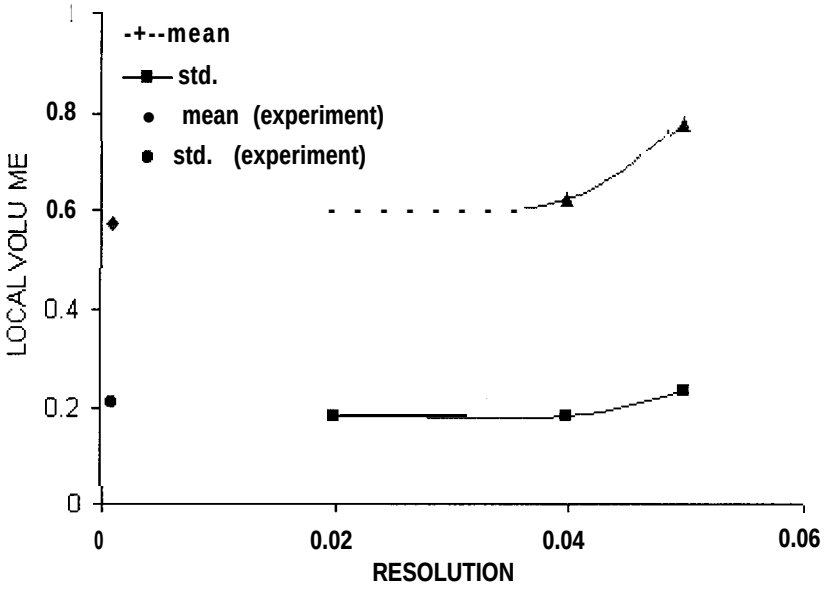


Figure 5-6. Effect of ROS on the distribution of local material volume.

5.2.2. Dimension of Virtual Structures

The dimension of a virtual structure is restricted by the capacity of the computer on which the simulation is performed. More specifically, it is restricted by the memory size and the capacity of the hard drive. With the limitation, it is desirable to simulate virtual structures which are small in dimensions but can represent the properties of larger structures. Several tests were carried out to investigate the impact of simulation dimensions. First, a large virtual structure (named BG) was constructed under the condition shown in Table 5-2.

The structure was then divided into 36 specimens with dimensions 5.0 x 5.0. The coverage, thickness, material volume and bonded area of each specimen were recorded during the division process. Another two sets of structures (named SML- 1 and SML-2) were then constructed under the conditions also shown in Table 5-2. Thirty six SML-1 structures

Table 5-2. Input parameters of the simulation processes for testing of sample dimension.

	BG	SML-1	SML-2
ROS	0.02	0.02	0.02
Element Length	5.0	5.0	5.0
Element Width	0.2	0.2	0.2
Element Thickness	0.1	0.1	0.1
Dimensions of Structure	30 x 30	5.0 x 5.0	10 x 10
Width of Edge Zone	10	22.5	20
Number of Elements	6000	6000	6000

were constructed. No division was performed on the structures. Nine SML-2 structures were built. Each was divided into four specimens. As was the case for the BG structure, the coverage, thickness, material volume and bonded area on these two types of structures were recorded. In addition, RBA and volume density of each specimen were calculated.

Output properties of SML- 1 and SML-2 were compared with the correspondent properties of BG. The two-sample two-sided Smimov test method was employed. The results showed that all the properties but the bonded area and RBA of SML-1 structure were identical to the properties of BG structure. The SML-1 structures had significantly more bonded area and higher RBA. All the properties of the SML-2 structures were identical to the properties of the BG structure. Based on the scrutiny into the simulation algorithm, it was concluded that the discrepancies on the bonded area and RBA were due to an edge effect, a different type from the ***edge*** effect which the ***Edge*** Zone was designed for. The effect is illustrated in Figure 5-7. The SML-1 structure had more bonded area since each specimen was a whole simulated structure. Therefore, all the edges of a specimen were exposed to the ***edge*** effect. SML-2 specimens had only two affected edges. To confirm the conclusion, Another set of 36 structures were generated under the same conditions as those for the SML- 1 struc-

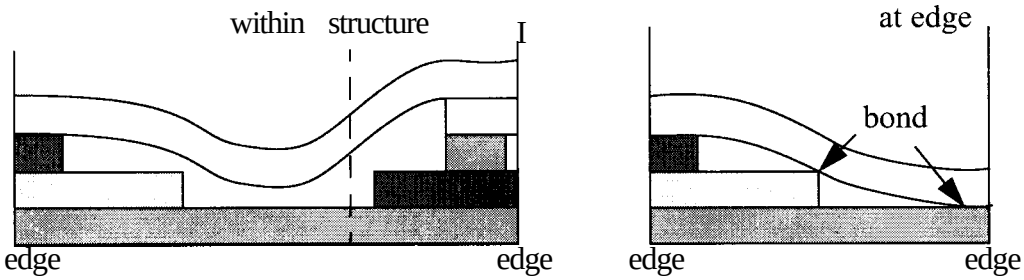


Figure 5-7. Lacking of support on one end of an element results in more bonded area.

tures, except for the larger dimensions, 6.0 x 6.0, and different width of Edge Zone, 22. Only the space with dimensions 5.0 x 5.0 in the center of each structure was then tested. The properties of the structures were same as those of the BG structures. This result has indicated that it is possible to overcome the limitation of the simulation environment with only the trade-off of more operation time.

5.2.3. Width of Edge Zone

The Edge *Zone* is the area between the *Sample Area* and the *Forming* Box. The function of the *Forming* Box is to control the amount (number and volume) of elements in the simulation process. *Forming* Box causes the change of the orientation of the elements which contact its sides. This change may consequently affect the properties of the structure adjacent to the *Forming Box*. Edge *Zone* is designed to eliminate this effect. From Figure 5-8, it can be seen that if the length of an element is less than the width of Edge *Zone*, the rotation around the centroid of the element will not make its body appear within the *Sample Area*. Therefore, the width of Edge *Zone* should be at least as large as the maximum length of all the elements:

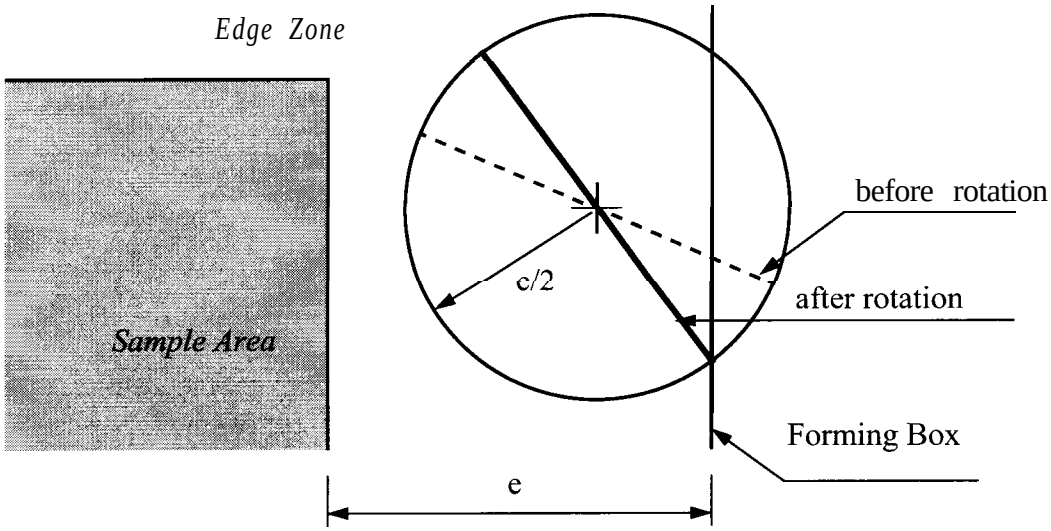


Figure 5-8. The width of Edge Zone is determined based on the maximum length of elements.

$$e \geq L_{max} \tag{5-3}$$

where e is the width of *Edge* Zone, and L_{max} is the maximum length of the elements. To be more conservative, the width of Edge Zone could be even larger. However, with larger *Edge Zone,* more processing time would be needed to simulate a structure with certain dimensions because longer time would have to be spent to process the elements within the *Edge* Zone, no matter whether they affect the final simulated structure or not.

If the length of the elements is normally distributed, the maximum length can be estimated as

$$L_{max} = \bar{L} + 3s_L \tag{5-4}$$

where $\bar{L}$ is the mean length of the elements, s_L is the standard deviation since about 99% of the values of element length will be included in the range of $\bar{L} \pm 3s_L$. If the length is governed by a lognormal distribution, the maximum length will be modified to

$$L_{max} = \bar{L}^2 \sqrt{\bar{L}^2 + s_L^2} \cdot \left[exp\left(\sqrt{ln \frac{\bar{L}^2 + s_L^2}{\bar{L}^2}} \right) \right]^3 . \quad (5\text{-}5)$$

where $\bar{L}$ and s_L are the mean and standard deviation of the lognormal distribution, respectively.

5.2.4. Length of Elements

To evaluate the influence of element length on the structure of composites, five types of virtual structures were generated with part of conditions shown in Table 5-3.

Table 5-3. Formats of virtual composite structures for testing effect of element length.

	Length of Elements	Total No. of Elements
Type A	1.0	6000
Type B	2.0	3000
Type C	3.0	2000
Type D	4.0	1500
Type E	5.0	1200

The width and thickness of all the elements were 0.2 and 0.1, respectively. Sample sizes were 30 x 30. The widths of Edge Zone were 5. Three replicate structures were generated for each type. Each virtual structures was divided into 36 specimens with dimensions of

5.0 x 5.0. Two-sided Smirnov tests on the local coverage, material volume, RBA and local volume density indicated that the properties of the composite samples were consistent within each type when the statistics were compared with the 0.95 quantile (Conover, 197 1). Three data sets within each type of the virtual structures were then consolidated into one set with sample size of 108 for each properties. The five data samples of the five composite types were then under two-sided five-sample Smirnov test. The test results rejected the hypothesis that all the properties were identical among the different types. This implied that length of the elements did influence the virtual structures. Further analysis on the output of the Smirnov tests showed that the local coverage was fairly consistent except for one extreme case which brought the test statistic up to 0.185, while the 0.90, 0.95 and 0.99 quantiles are 0.146, 0.166 and 0.206, respectively.

Significant influence of element length was found on the distribution of local material volume. Figure 5-9 shows that the variance of the local volume decreases but the mean value remains little changed when the length of elements is reduced. The phenomenon is reasonable since more elements have to be generated for the structures made of shorter elements, as shown in Table 5-3, in order to keep the total material volume of structures identical. This causes the structure to be more uniform. A nonlinear relationship between element length and the standard deviation of the local material volume distribution was derived, which is plotted in Figure 5-10.

The length of element impacted the distribution of local volume density on both its mean and variance (Figure 5-11). The mean of the local volume density linearly increased along

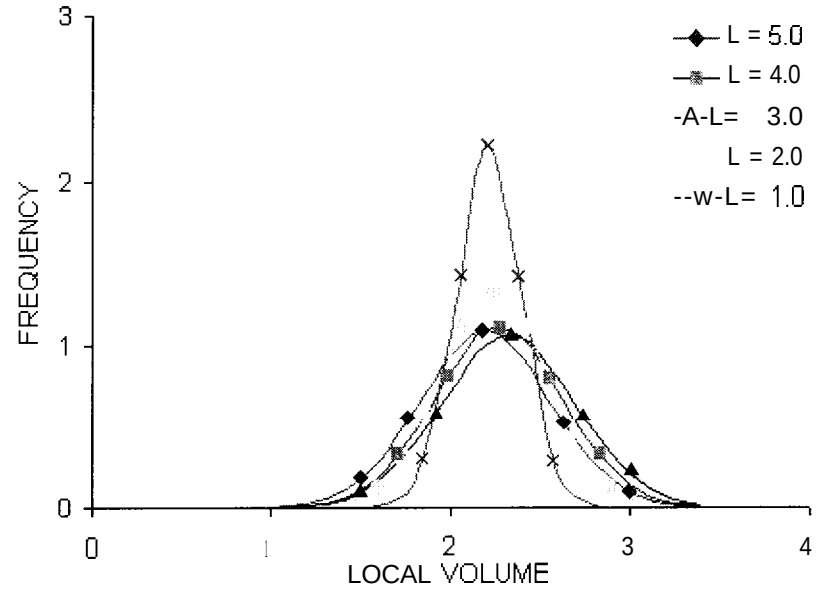


Figure 5-9. Effect of element length on the distribution of local volume.

L is the mean lengths of elements.

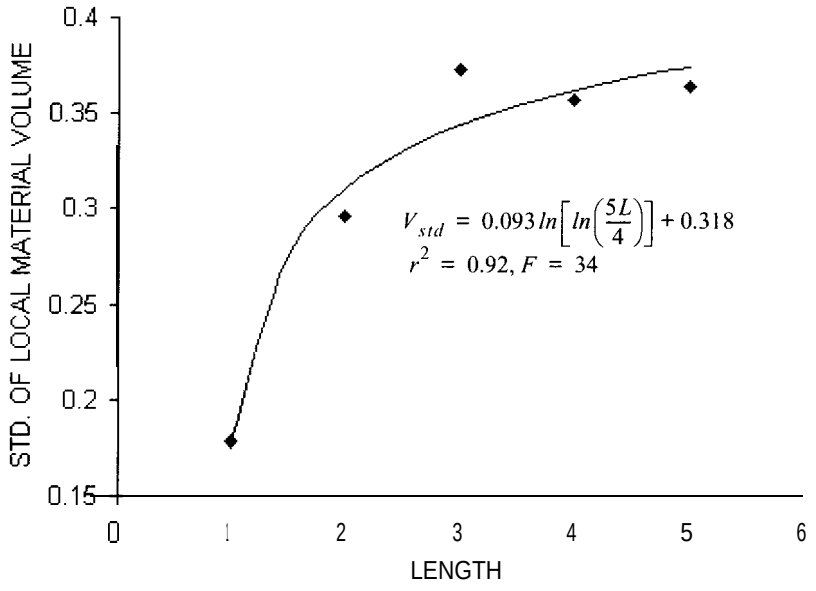


Figure 5- 10. The variation (standard deviation, V_{std}) of the local material volume increases with the increase of the element length (L).

with the increasing of element length (Figure 5-12). A nonlinear relationship existed between the standard deviation of the volume density distribution and the element length (Figure 5-13).

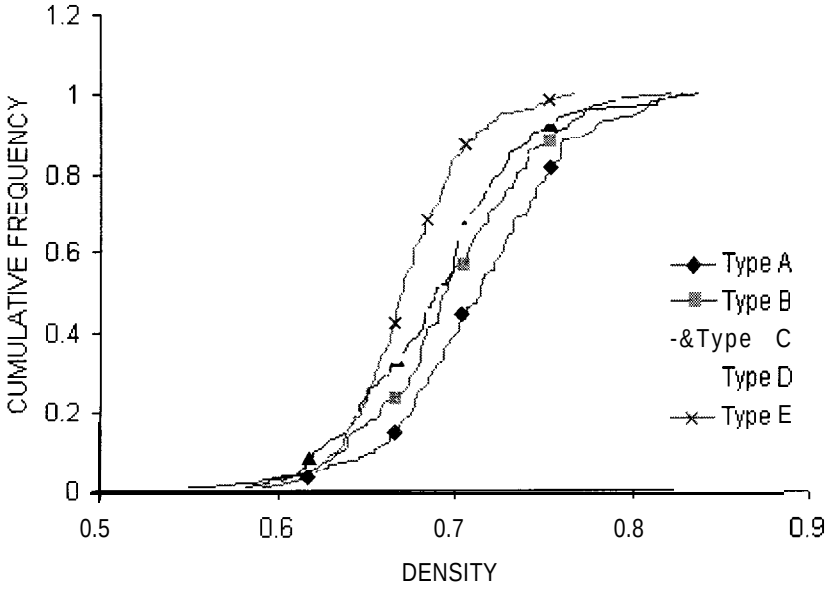


Figure 5- 11. Distributions of local volume density of the structures with different element length.

5.2.5. Width and Thickness of Elements

The width and thickness of an element determines the moment of inertia of the cross section area of an element. The moment of inertia consequently determines the flexure of the element under any load applied to the composite structure. To determine how the width and thickness of elements affect virtual structures, 16 structures with the combination of four different width and four different thickness levels were generated. Each type of structures was made of elements with certain width and thickness. The number of elements varied with the width and thickness of elements so that the total material volume of the structures

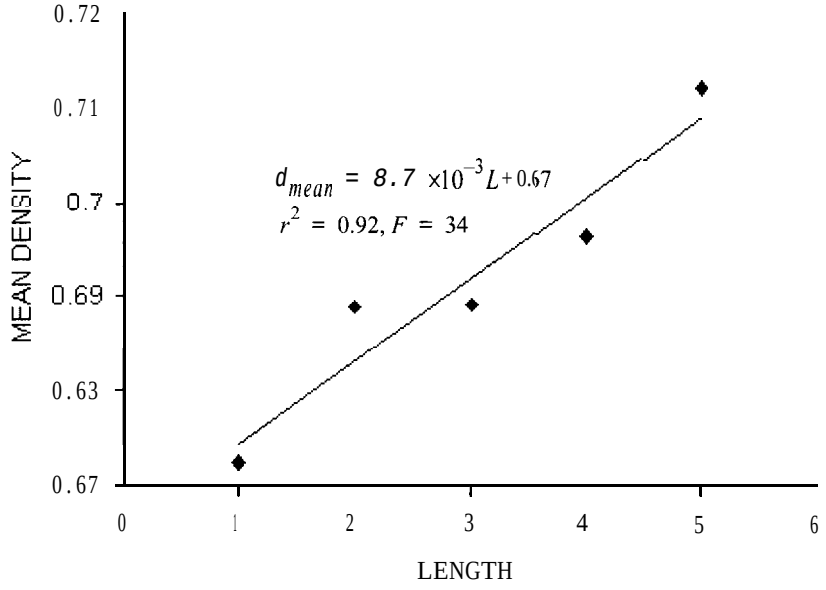


Figure 5-12. Effect of element length on the mean of local density distribution.

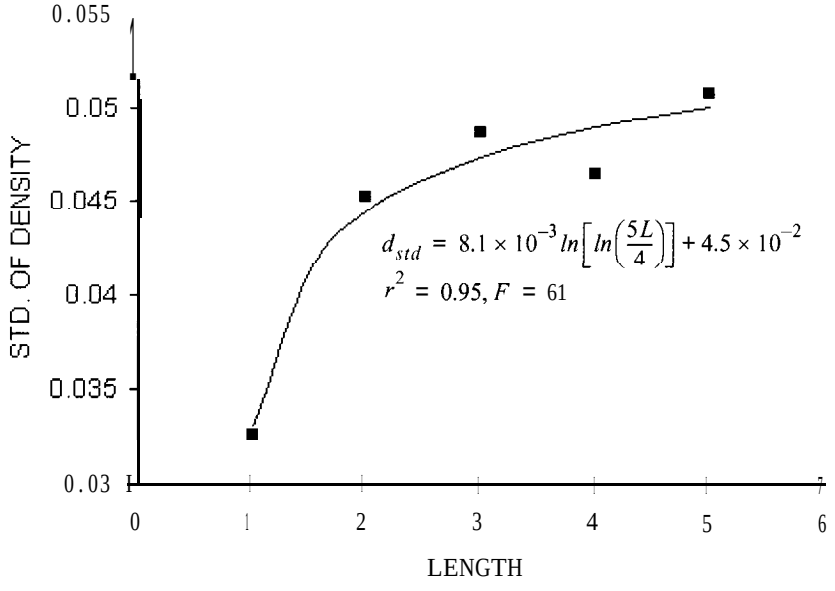


Figure 5-13. Effect of element length on the standard deviation of the local volume density distribution.

was constant for all structures. The length of all elements was 5.0. The other conditions were same as those for testing the effect of element length in Section 5.2.4. Table 5-4 lists the formats of the virtual structures.

Table 5-4. Formats of composite samples for testing effect of element width and thickness.

thickness \ width	0.2	0.4	0.6	0.8
0.08	1500[a]	750	500	375
0.1	1200	600	400	300
0.12	1000	500	333	250
0.14	857	429	286	214

a. The numbers are the total number of elements simulated.

Local coverage, material volume, density and RBA of each sample were compared to one another through two-sided Smirnov tests. Statistically, the distributions of local coverage were identical at the 0.05 significance level. Only two structures were found with significantly different distributions of the local material volume, which made the statistics (0.361, 0.333 and 0.306) exceed the quantile, 0.305, at the 0.05 significance level (Conover, 1971). It seemed that the width and thickness of elements did not have significant impact on the distributions of local coverage and local material volume. However, the distribution of RBA varied with both the width and thickness of elements. From Figure 5-14, it can be observed that both the mean and the variance of the distribution of RBA are influenced by the width of elements. Figure 5-15 shows the change of the distribution with the thickness of the elements. Results of regression analysis showed that mean value of RBA related to the width and thickness of elements in the following way:

$$\xi = -0.035 - 0.019W + \frac{0.021}{T} + 0.0062\frac{ln.w}{T} \quad (5\text{-}6)$$

where ξ is the mean of RBA, W and T are the width and thickness of the elements respectively ($0.986 \le r^2 \le 0.999$ and $137 \le F \le 22023$). The combined influence is shown in Figure 5-16. It indicates that structures with thicker elements have lower RBA. This can be explained as that larger thickness causes higher moment of inertia and consequently lower flexibility of elements. Stiffer elements may result in less bonded area as shown in Figure 5-17. Increase of the element width will also make the element stiffer, but the effect may be much less significant than that of the thickness according to Equation 3-22. Besides, wider elements will have more available area for bonding. Apparently, the increase of the bonded area overcame the loss due to the decrease of flexibility in the virtual structures. Therefore, the RBA increased a little while the width of elements increased.

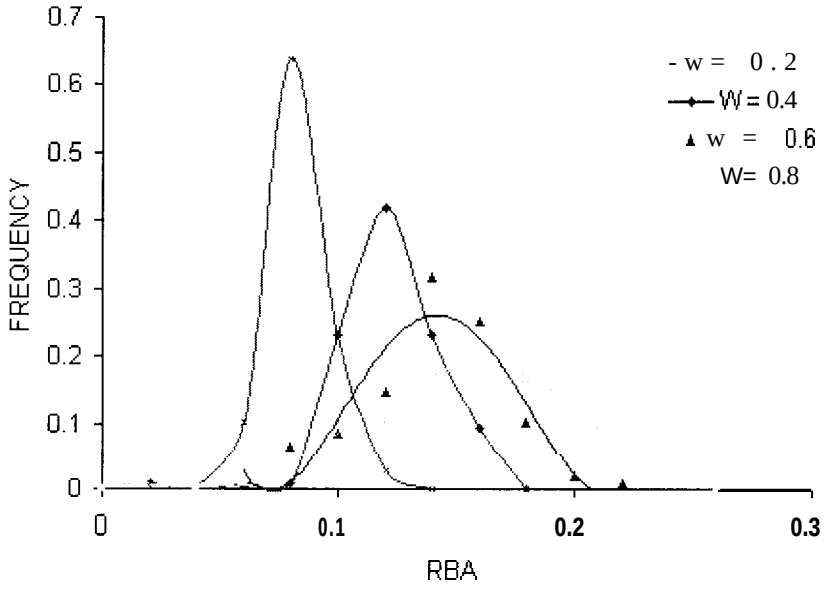


Figure 5-14. Effect of element width on the distribution of RBA.

(Thickness of elements: T=0.1).

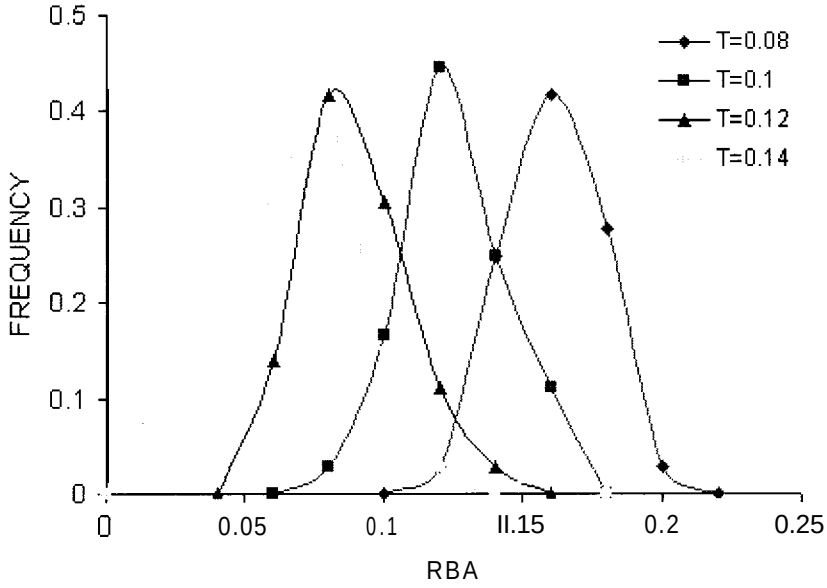


Figure 5-15. Effect of element thickness on the distribution of RBA.

(Width of elements: $W = 0.4$)

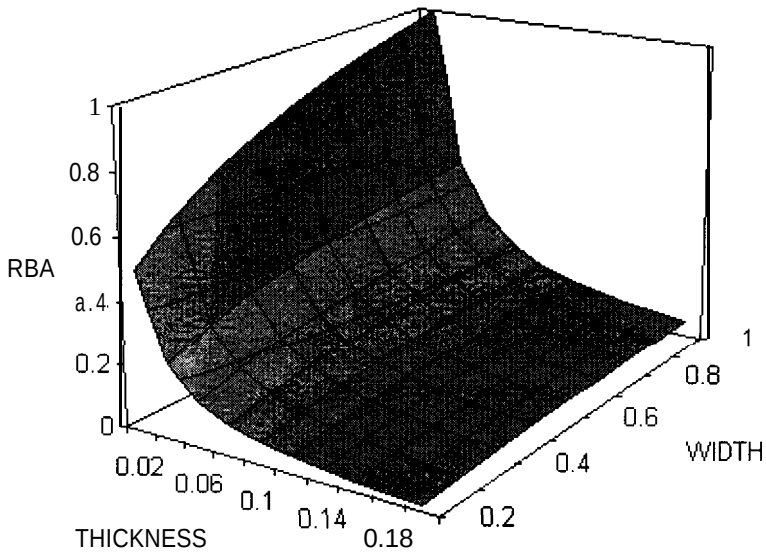


Figure 5-16. Relationship between mean of RBA distribution and the width and thickness of elements.

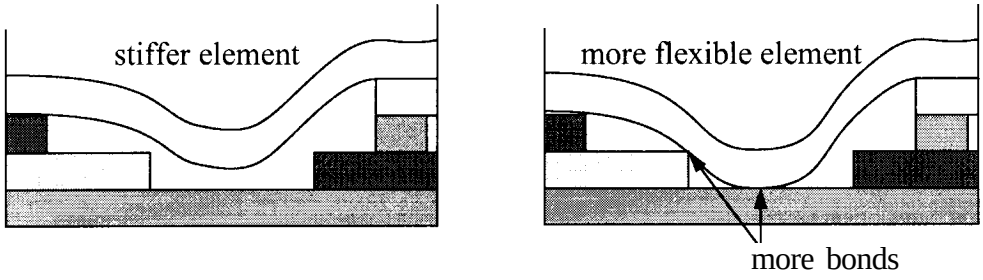


Figure 5-17. Stiffer elements result in less bonded area.

Significant positive linear relationships between the standard deviation of the RBA and the width of elements were found at all element thickness levels ($0.914 \leq r^2 \leq 0.997$ and $21 \leq F \leq 667$). Comparatively, the element thickness had less impact on the variance of the RBA ($0.550 \leq r^2 \leq 0.934$ and $2.44 \leq F \leq 28.5$). These effects are illustrated in Figure 5-18 and Figure 5-19. The distributions considered were the horizontal descriptions of the properties. The variance of any distribution discussed was on the horizontal direction. Less elements resulted in higher variation. Although an increase of thickness and width of elements could both cause less elements in a structure, since width of elements was in the horizontal direction, it had stronger impact on the horizontal properties.

5.2.6. Elasticity of Elements

In the literature, fiber flexibility refers to the combination of the Young's Modulus of wood fibers and the moment of inertia of the cross sections (Tam Doo and Kerekes, 1982). In current research, these two properties were considered separately for more detailed evaluation. The factor of moment of inertia has been discussed in the previous section with the width and thickness of elements. The relationship between RBA and the elasticity (MOE)

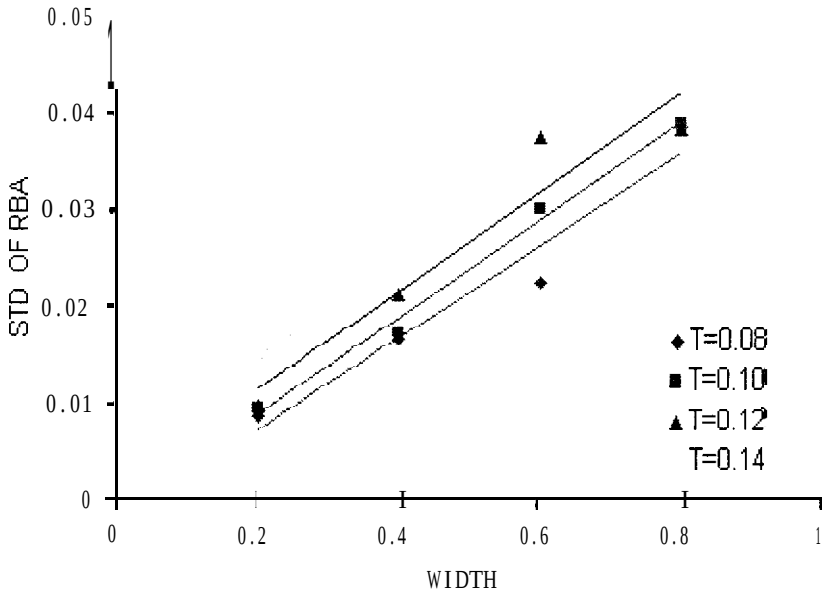


Figure 5-l8. Effect of element width on the standard deviation of RBA distribution.

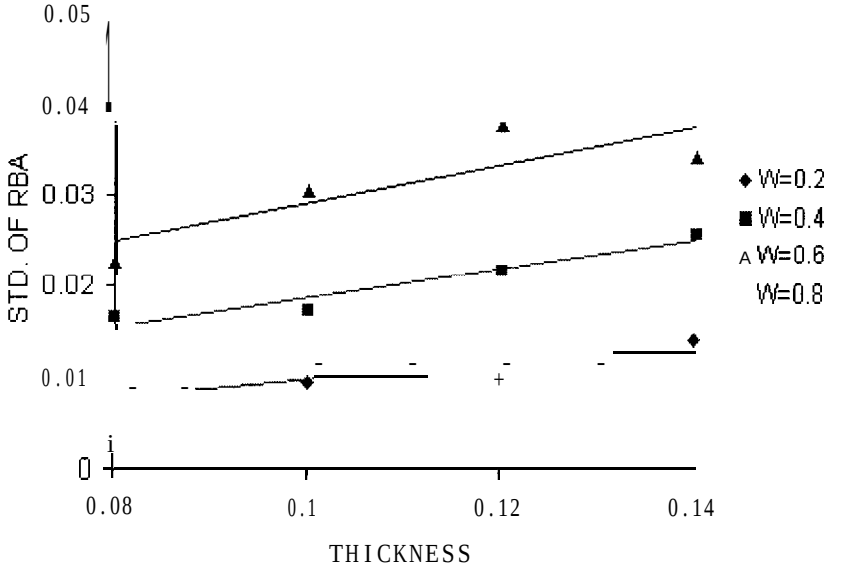


Figure 5-19. Effect of element thickness on the standard deviation of RBA distribution.

of the elements was investigated by the simulation of virtual structures composed of element with different MOE levels. Eight levels of MOE were chosen based on the knowledge of MOE of the foam tape. All input parameters but the material property, MOE, were the same as those used for the Type II foam tape structure in the experimental verification. Extra pressure, 1 .O pound, was applied to enhance the significance of RBA. The density change of the foam tape due to the pressure was not considered. The relationship between RBA in the structures and the correspondent MOE level is shown in Figure 5-20. As shown, RBA decreased when MOE of the elements increased. Linear regression analysis on RBA versus the natural logarithm of MOE yielded an $r^2 = 0.98$ and $F = 276$. This phenomenon may also be explained with Figure 5-17.

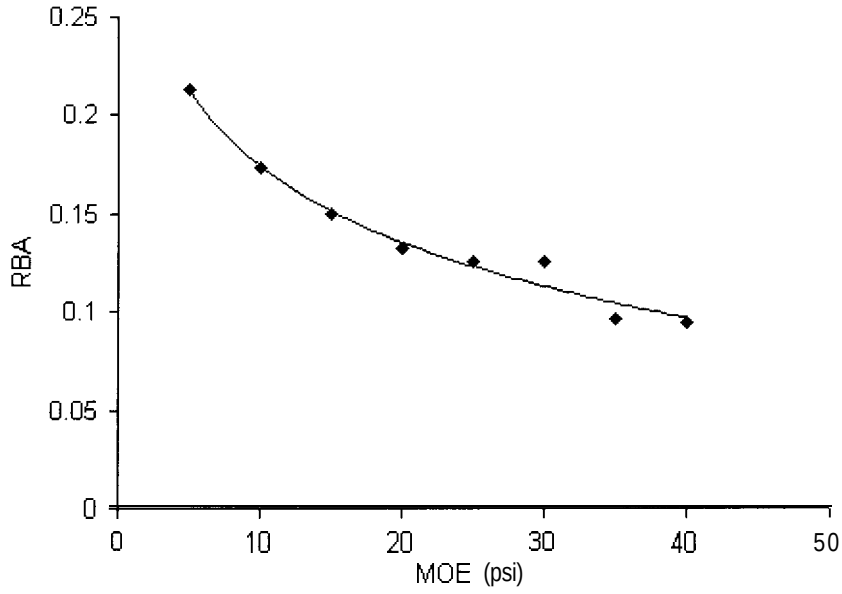


Figure 5-20. Effect of MOE of element on mean value of RBA.

5.2.7. Orientation

The effect of element orientation preference was tested with virtual composite structures. The parameters for the simulation are listed in Table 5-5.

Table 5-5. Input parameters of simulation for testing effect of orientation.

Input Parameter	Value
Element Length (inch)	5.0
Element Width (inch)	0.5
Element Thickness (inch)	0.125
Target Density of Sample ($lb./inch^3$)	3.6×10^{-3}
Sample Size ($inch^3$)	30 x 30 x 0.25
Width of Edge Zone (inch)	5.0
ROS (inch)	0.02

Five orientation concentration levels defined by the concentration parameter, k, of a von Mises distribution, 0, 0.4, 1 .0, 2.0, and 3.0, were tested. The change of the virtual structures and the orientation distribution according to the concentration parameter is illustrated by Figure 5-21 and Figure 5-22.

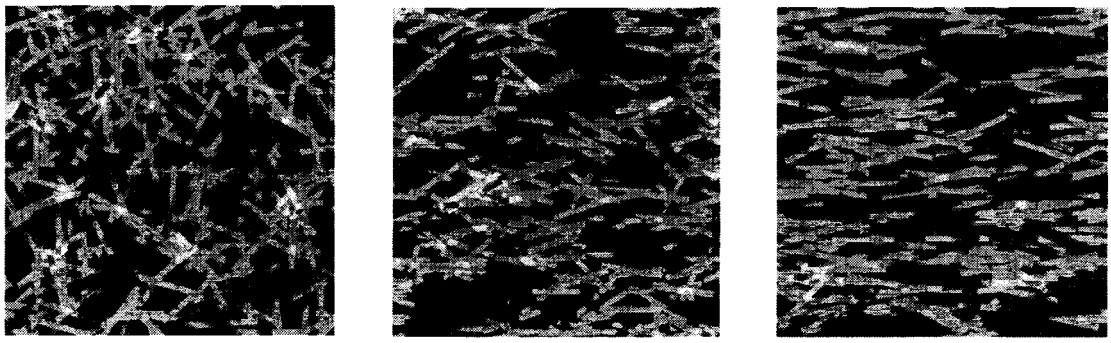


Figure 5-21. Change of the simulated structures with the change of the element concentration parameter, k.

(a) k = 0, (b) k = 1.0, (c) k = 3.0.

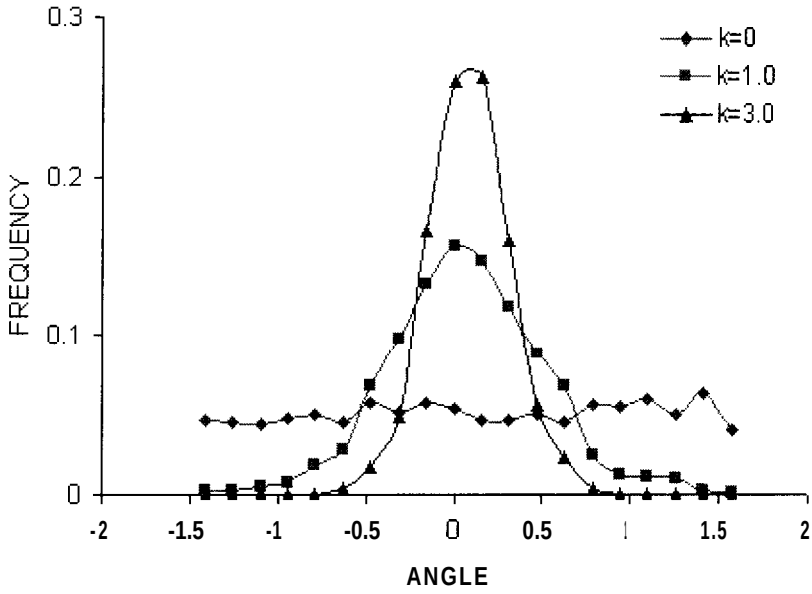


Figure 5-22. Distributions of element orientation at different concentration levels.

At each concentration level, four replicate structures were made. Each virtual structure was divided into specimens with horizontal dimensions of 5.0 x 5.0. The average thickness of each specimen, the material volume and the bonding area in the specimen were recorded. Descriptive statistical analysis was conducted on the data. Figure 5-23 shows the mean and standard deviation of the local material volume versus the orientation concentration level.

It can be seen from the figure, that the mean and the standard deviation of the local material volume distribution does not change significantly with the concentration parameter. Results of a regression analysis had shown r^2 values of 0.0028 and 8.48 x 10^{-5} for the mean and standard deviation, respectively. This further indicated that orientation concentration did not influence the volume distribution significantly.

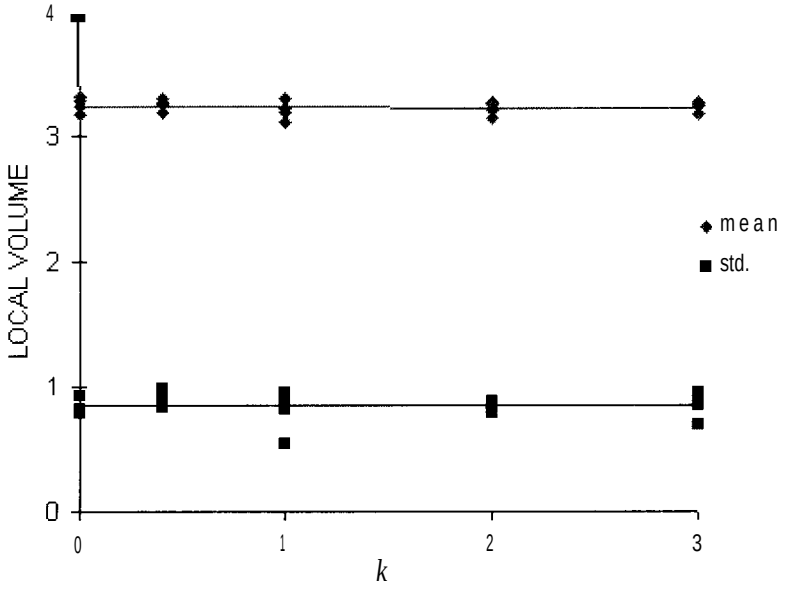


Figure 5-23. Effect of orientation concentration on distribution of volume.

The local volume density distribution was also checked against the orientation concentration. The results of a regression analysis showed that the volume density was a exponential function of the orientation concentration level. The relationship between the mean volume density and the concentration level was:

$$d = \frac{0.066}{2^k} + 0.852, \ (r^2 = 0.96). \tag{5-7}$$

where d is the volume density and k is the concentration parameter of von Mises distribution. Figure 5-24 shows the relationship. Similar trend was predicted by Amiri, et al. (1994) with a combination of an interactive multi-planar model and a fiber orientation function.

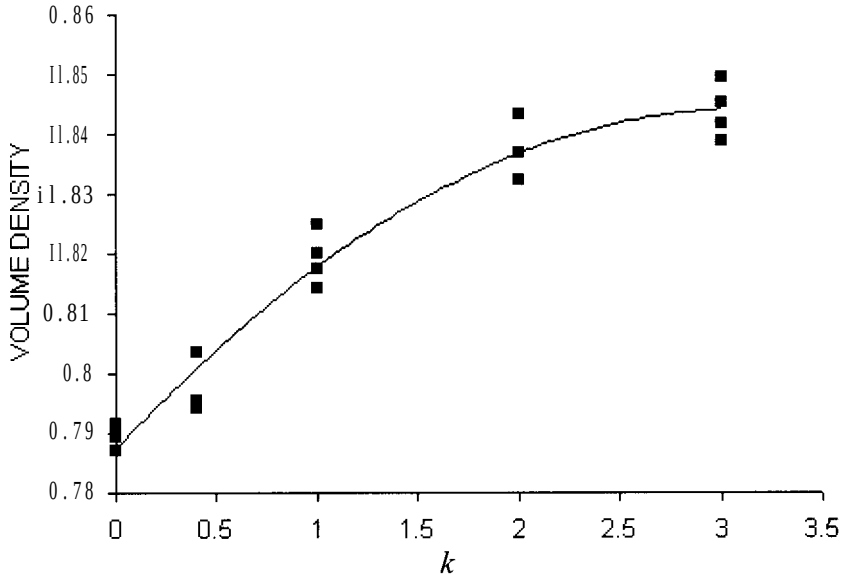


Figure 5-24. The relationship between the mean of local volume density distribution and the orientation concentration parameter.

The variance of volume density was not influenced by the orientation concentration (r^2 = 6.44 x 1 0^{-6} and F = 1.16 x 1 0^{-4}). This result was contradictory to Dodson and Fekih's prediction (Dodson and Fekih, 199 1) which said that the variance of mass density of paper linearly increased with the increase of the eccentricity of the fiber axis orientation distribution.

While material volume distribution remained unchanged, the bonded area in the structure did change along with the concentration level (Figure 5-25). The mean of the distribution of RBA was a positive linear function of the concentration parameter. Results of regression analysis showed that the r^2 value was 0.83 with F = 141.09. The standard deviation of the distribution of RBA was not strongly related to the concentration level since the r^2 value

was only 0.567 with $F = 39.24$. The trends were different from what Lu and Carlsson predicted, which stated that the volume fraction of bonds at a given apparent sheet density was affected very little by fiber orientation (Lu and Carlsson, 1996).

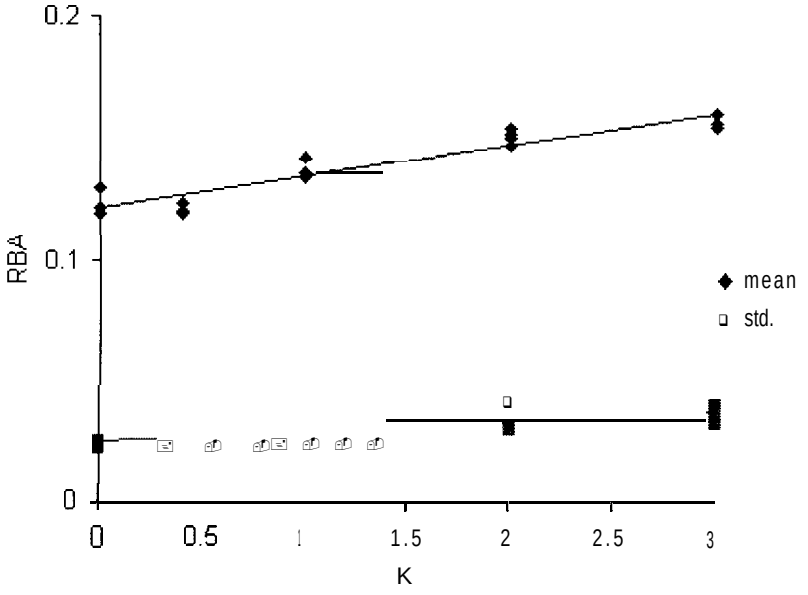


Figure 5-25. Effect of orientation concentration on distribution of bonded area.

The influence of element orientation is a complicated phenomenon which can not be explained with simple analytical models. Contradictory model predictions will require experimental verification to validate or modify prediction schemes.

6 CONCLUSIONS AND RECOMMENDATIONS

6.1. Conclusions

A computer simulation software has been developed. This software, based on stochastic principles and simulation of the properties of individual composite elements and the interactions among elements, can be used to produce and test virtual composite structures. The simulation process was implemented by imitating the scenario of laboratory procedures for making wood composites. Users are allowed to specify element properties and some basic manufacturing conditions. Information contained in the virtual structures includes the morphological description of each individual element in the virtual structure and the surface contour of the virtual structure. Distributional properties of local material volume, coverage and bonded area can be obtained through the testing process of the software.

Since it is based on very fundamental rules, this simulation methodology guarantees much more realistic descriptions of the microstructures as compared with other models identified in the literature. The reliability of the model was verified by comparing the properties of virtual composite structures with hand-made composite structures. The comparison of the distribution of material volume and coverage indicate that the virtual structures can reasonably represent their real world counterparts. Although the verification was not performed

with real wood materials, the capability and reliability of the simulation procedure could be extended to wider range of materials and element geometries.

Some microstructural features were statistically characterized based on the output of the simulation process. The effects of several input parameters were also analyzed. Some findings are listed below. Although these phenomenon were observed in the simulation of the special composite, they certainly supplied qualitative descriptions about the mechanism of the change on the microstructures and possible effects from the element geometry and manufacturing conditions.

The distributions of local coverage are Poisson distributions. The local material volume, volume density and relative bonded area (RBA) are all normally distributed.

The resolution of the simulation turned out to be the most important factor affecting the simulated structures. It has a strong impact on the continuous data describing element geometry. The choice of resolution level is restricted by the capacity of the computer on which the simulation is performed. The smaller dimensions of the virtual structures are a trade-off for higher resolution. At certain resolution levels, larger virtual structures may be simulated by substituting a number of smaller structures.

The length of elements influences the variance but not the mean of the local material volume. It also affects the mean and variance of the volume density distribution. The distributions of coverage and bonded area are not changed by the length of elements. Structures

constructed with longer elements show larger variation on local material volume and volume density. The average volume density increases linearly with the length of elements.

The width and thickness of elements significantly changes the amount of bonded area and its variation. Both the mean and variance of the local bonded area increase with the element width. The mean RBA decreases while the variance increases as the element thickness is increased.

Relative bonded area (RBA) is a function of the elasticity of element. Increasing the MOE will cause less bonded area in the composites.

Orientation of elements does not change the distributional manner of local material volume. However, structures with more oriented elements may have more bonded area and larger variation on RBA. The mean of volume density also increases with more concentrated element orientation, but the variance of volume density remains constant.

The superiority of the current simulation procedure can be clearly seen from the output information. The output of the virtual composite structures includes direct descriptions of some important features, such as the void structure and bonded area. Other existing models can only describe the distributions of number of crossings or free fiber lengths, which are only meaningful in pure two-dimensional structures and are only rough estimators for the voids and bonded area even in the two-dimensional structures. The superiority also lies in the intuitive reasonability of all the predictions on the effects of element properties and manufacturing conditions, which can not be always seen from other models. The three-

dimensional visualization of the virtual composite structures also gives researchers a basis for judging the reasonability of the simulation processes.

6.2. Recommendations

There are a few limitations on the current simulation program.

The simulation program may supply suggestions for experiments on wood particulate composites. However, since the bottom-up simulation relies upon the knowledge of the elements and their interactions with each other, and the properties of wood materials vary significantly, the advantage of using the software may be decreased by the difficulty of implementing the verifying experimental work.

The current program is not able to simulate the large variations due to flocculation, fines, or fiber bundles in the real paper or other composites. This feature may be added by introducing a multiple-step stochastic simulation of the element geometries and locations.

It is almost impossible to control the thickness of virtual structures since pressure transferring through element-to-element contacts has not been implemented. The implementation of this would require the inclusion of horizontal deformation and densification of elements. In addition, the bending-beam assumption in the simulation may have to be modified.

In paper or wood fiber composite, fibers may have various level of initial curl. It is possible to introduce certain curve functions into the fiber generation procedures.

REFERENCES

Alexander, S. D. and R. Marton. 1968a. Effect of beating and wet pressing on fiber and sheet properties, I. Individual fiber properties. Tappi. 5 1 (6): 277-282.

Alexander, S. D. and R. Marton. 1968b. Effect of beating and wet pressing on fiber and sheet properties, II. sheet properties. Tappi. 5 1 (6): 283-288.

Amiri, R., J. R. Wood, A. Kamis, and J. Gorres. 1994. The apparent density of paper. Journal of Pulp and Paper Science. 20 (5): 142-148.

Casti, J. L. 1997. *Would-Be Worlds - How Simulation Is Changing the Frontiers of Science.* John Wiley & Sons Inc. New York. 242pp.

Chou, T. *1992. Microstructural Design of Fiber Composites.* Cambridge University Press. New York. 569pp.

Conover, W. J. 197 1. *Practical Nonparametric Statistics.* John Wiley & Sons Inc. New York. 462pp.

Corte, H. and E. H. Lloyd. 1966. Fluid flow through paper and sheet structure. In *Consolidation of the Paper Web, Transactions of the Fundamental Research Symposium,* Cambridge. 1965 (Edited by F. Bolam). Technical Section of BPBMA, London. 981-1009.

Dai, C. and P. R. Steiner. 1993. Compression behavior of randomly formed wood flake mats. Wood and Fiber Sci. 25(4): 349-358.

Dai, C. and P. R. Steiner. 1994a. Spatial structure of wood composites in relation to processing and performance characteristics, Part 2. Modelling and simulation of a randomly-formed flake layer network. Wood Sci. Technol. 28 (2): 135-l 46.

Dai, C. and P. R. Steiner. 1994b. Spatial structure of wood composites in relation to processing and performance characteristics, Part 3. Modelling the formation of multi-layered randomly flake mats. Wood Sci. Technol. 28 (3): 229-239.

Dale, N., C. Weems and M. Headington. 1996. *Programming and Problem Solving With* C++. D. C. Heath and Company. Lexington, MA.

Deng, M. and C. T. J. Dodson. 1994. Random star patterns and paper formation. Tappi. 77 (3): 195-199.

Deng, M. and C. T. J. Dodson. 1994. *Paper-An Engineered Stochastic Structure.* Tappi Press. Atlanta, GA. 284pp.

Dodson C. T. J. 199 1. The statistical evolution of paper in three dimensions. International Paper Physics Conference. Tappi Proceeding.

Dodson C. T. J. and K. Fekih. 1991. The effect of fiber orientation on paper formation. Journal of Pulp and Paper Science. 17 (6): 203-206

Dodson, C. T. J. 1992. The effect of fiber length distribution on formation. J. Pulp and Paper Sci. 18 (2): 74-76.

Dodson, C. T. J. and W. W. Sampson. 1996. The effect of paper formation and grammage on its pore size distribution. Journal of Pulp and Paper Science. 22 (5): 165-169.

Dodson, C. T. J. and W. W. Sampson. 1997. Modeling a class of stochastic porous media. Appl. Math. Lett. 10 (2): 87-89.

Famood, R. R., C. T. J. Dodson, and S. R. Loewen. 1995. Modeling flocculation: A gallery of simulated flocculated papers. Nordic Pulp and Paper Research Journal. 12 (2): 86-89.

Famood, R. R., N. Yan, M. T. Kortschot, and C. T. J. Dodson. 1997. Modeling flocculation. Part I: Radom disk model. Journal of Pulp and Paper Science. 2 1 (10): 348-355.

Gatchell, C. J., B. G. Heebink and F. V. Hefty. 1966. Influence of component variables on properties of particleboard for exterior use. Forest Products J. 16 (4): 46-59.

Ghosh, S. and S. N. Mukhopadhyay. 199 1. A two-dimensional automatic mesh generator for finite element analysis for random composites. Computers & Structures. 41 (2): 245-256.

Gorres, J., C. S. Sinclair, and A. Tallentire. 1989. An interactive multi-planar model of paper structure. Paperi ja Puu - Paper and Timber. 71 (1): 54-59.

Got-r-es, J. and P. Luner. 1992. An apparent density model of paper. Journal of Pulp and Paper Science. 18 (4): 127-130.

Harless, T. E. G, et al. 1987. A model to predict the density profile of particleboard. Wood and Fiber Sci. 19 (1): 81-92.

Harris, R. A. 1977. Measuring particle alignment in particleboard and predicting selected mechanical properties of oriented board. Ph. D. desertion.

Hartler, N. and Nyren, J. 1970. Transverse compressibility of pulp fibers. II. Influence of cooking method, yield, beating and dring. Tappi. 53 (5): 320-323.

Hasuike, M., T. Kawasaki and K. Murakami. 1992. Evaluation method of 3-D geometric structure of paper sheet. Journal of Pulp and Paper Science. 18 (3): 114-120.

Heebink, B. G. and R. A. Hann. 1959. How wax and particle shape affect stability and strength of oak particleboards. Forest Products Journal. 9 (7): 197-203.

Hermans, P. H. 1949. *Physics and Chemistry of Cellulose Fibers.* Elsevier Pub. Co. New York. 534pp.

Jang, H. F., A. G. Robertson and R. S. Seth. 1991. Optical sectioning of pulp fibers using confocal scanning laser microscopy. Tappi. 7 1 (10): 2 17-2 19.

Jangmalm, A and S. Ostlund. 1995. Modelling of curled fibers in two-dimensional networks. Nordic Pulp and Paper Research Journal. 10 (3): 156- 16 l(166).

Kallmes, 0. and H. Corte. 1960. The structure of paper, Part I. The statistical geometry of an ideal two dimensional fiber network. Tappi. 43 (9): 737-752.

Kallmes, O., H. Corte and G. Bemier. 1961. The structure of paper, Part II. The statistical geometry of a multiplanar fiber network. Tappi. 44 (7): 5 19-528.

Kallmes, 0. and G. Bemier. 1963. The structure of paper, IV. The free fiber length of a multiplaner sheet. Tappi. 46 (2): 108-l 14.

Kallmes, 0. and C. Eckert. 1964. The structure of paper, VII. The application of the relative bonded area concept to paper evaluation. Tappi. 47 (9): 540-548.

Karenlampi, P. P., H. T. Suur-Hamari, M. J. Alava and K. J. Niskanen. 1996. The effect of pulp fiber properties on the in-plane tearing work of paper. Tappi. 79 (5): 203-2 10.

Kim, C. Y., D. H. Page, F. El-Hosseiny, and A. P. S. Lancaster. 1975. The mechanical properties of single wood pulp fibers. III. The effect of drying stress on strength. J. Applied Polymer Science. 19 (5): 1549- 156 1.

Klauditz, W. and A. Buro. 1962. The suitability of sawdust for particleboard manufacture. Holz als Roh-und Werkstoff. 20 (1): 19-26.

Komori, T. and K. Makishima. 1977. Number of fiber-to-fiber contacts in general fiber assemblies. Textile Research Journal. 47 (1): 13-17.

Lang, E. M. and M. P. Wolcott. 1996, A model for viscoelastic consolidation of wood-strand mats. Part I. Structural characterization of the mat via Monte Carlo simulation. Wood and Fiber Science. 28 (1): 1 00- 109.

Laufenberg, T. L. 1984. Flakeboard fracture surface observations and correlations with orthotropic failure criteria. J. of Inst. of Wood Science. 1 O(2): 57-65.

Leopold, B. and J. L. Thorpe. 1968. Effect of pulping on strength properties of dry and wet pulp fibers from Norway Spruce. Tappi. 51 (7): 304-308.

Louis, P. and A. M. Gokhale. 1995. Computer simulation of spatial arrangement and connectivity of particles in three-dimensional microstructure: Application to model electrical conductivity of polymer matrix composite. Acta mater. 44 (4): 15 19-1528.

Lu, W. and L. A. Carlsson. 1996. Micro-model of paper Part 2: Statistical analysis of paper structure. Tappi. 79 (1): 203-210.

Maloney, T. M. 1993. *Modern Particleboard & Dry-Process Fiberboard Manufacturing.* Miller Freeman Inc. San Francisco. 68 lpp.

Mardia, *K.* V. 1972. *Statistic of Directional Data.* Academic Press. New York, N.Y. 357pp.

Mataki, Y. 1972. Internal structure of fiberboard and its relation to mechanical properties. Pages *2 19-253* in *Theory and Design of Wood and Fiber Composite Materials.* W. Cote Ed. Syracuse University Press. 4 18 pp.

Michell, J. and G. Freischmidt. 1990. Effect of fiber curl on the properties of wood pulp fiber-cement and silica sheets. Journal of Material Science. 25 (12): 5225-5230.

Mottet, A. L. 1967. The particle geometry factor in particleboard manufacturing. Proceedings of the Washington State Universtiy Particleboard Symposium. Pullman, Washington: WSU. 1: 23-73.

Nilsen, N., M. Zabihian, and K. Niskanen. 1998. KCL-PAKKA: a tool for simulating paper properties. Tappi. 81 (5): 163-166.

Niskanen, K. J. and M. J. Alava. 1994. Planar random networks with flexible fibers. Physical Review Letters. 73 (25): 3475-3478.

Nolan, G. T. and P. E. Kavanagh. 1995a. Octahedral configurations in random close packing. Powder Technology. 83 (3): 253-258.

Nolan, G. T. and P. E. Kavanagh. 1995b. Computer simulation of particle packing in acrylic latex paints. Journal of Coatings Technology. 67 (850): 37-43.

Page, D. H. 1967. The collapse behaviour of pulp fibers. Tappi. 50 (9): 449-455.

Page, D. H., F. El-Hosseiny, K. Winkler and A. P. S. Lancaster. 1977. Elastic modulus of single wood pulp fibers. Tappi. 60 (4): 114-l 17.

Page, D. H. and R. S. Seth. 1980. The elastic modulus of paper II. The importance of fiber modulus, bonding, and fiber length. Tappi. 63 (6): 113-l 16.

Panshin, A. J. and *C.* de Zeeuw. 1970. *Textbook* of *Wood Technology,* 3rd ed. McGraw-Hill, New York. 705pp.

Post, P. W. 1958. Effect of particle geometry and resin content on bending strength of oak flake board. Forest Products J. 8 (10): 3 17-322.

Press, W. H., B. P. Flannery, S. A. Teukolsky and W. T. Vetterling. 1990. Numerical *Recipes* in C. Cambridge University Press, New York. 735pp.

Pyrz, R. 1994. Quantitative description of the microstructure of composites. Part I: Morphology of unidirectional composite system. Composites Science and Technology. 50 (2): 197-208.

Quintanilla, J. and S. Torquato. 1997. Microstructure functions for a model of statistically inhomogeneous random media. Physical Review E. 55 (2): 1558- 1565.

Ramli, R. 1995. Properties and microstructure of medium density fiberbord (MDF) manufactured from oil palm empty fruit bunch fibers. M.S. dissertation. U. Maine.

Retulainen, E. 1996. Fiber properties as control variables in paper-making?. Paperi Ja Puu - Paper and Timber. 78 (4): 187-194.

Roberts, A. P. and M. A. Knackstedt. 1996. Structure-property correlations in model compsite materials. Physical Review E. 54 (3): 23 13-2328.

Roberts, A. P. and M. A. Knackstedt. 1997. Cross-property correlations for disordered materials. Physicochem. Eng. Aspects (129-130): 377-385.

Roberts, A. P. 1997. Morphology and thermal conductivity of organic aerogels. Physical Review E. 55 (2) 1286-1289.

Roberts, A. P. and E. J. Garboczi. 1999. Elastic properties of a tungsten-silver composite by reconstruction and computation. J. Mechanics and Physics of Solids. 47: 2029-*2050.*

Ross, *S.* M. 1985. *Introduction to Probability Models.* Academic Press, Inc. Orlando, Fl. 502pp.

Ross, R. F. and D. J. Klingenberg. 1997. Dynamic simulation of flexible fibers composed of linked rigid bodies. Journal of Chemical Physics. 106 (7): 2949-2960.

Seth, R. S. 1995. The effect of fiber length and coarseness on the tensile strength of wet webs: a statistical geometry explanation. Tappi. 78(3): 99-102.

Shaler, S. M. 1991. Comparing two measures of flake alignment. Wood Science and Technology. 26 (1): 53-6 1.

Shaler, S. M., D. Keane, H. Wang, L. Mott, E. Landis and L. Holzman. 1998. Microtomography of cellulosic structures. in *Proceedings of the 1998 TAPPI Process & Product Quality Conference.* Tappi Press, Milwaukee. p89-96.

Stahl, D. C., S. M. Cramer and R. L. Geimer. 1997. Effects of microstructural heterogeneity in cement excelsior board. Wood and Fiber Science. 29 (4): 345-352.

Stahl, D. C. and S. M. Cramer. 1998. A three-dimensional network model for a low density fibrous composite. Journal of Engineering Materials and Technology. 120 (2): 126-*130.*

Steiner, P. R. and C. Dai. 1993. Spatial structure of wood composites in relation to processing and performance characteristics, Part 1. Rationale for model development. Wood Science and Technology. 28 (1): 45-5 1.

Suchsland, 0. 1959, An analysis of the particle board process. Michigan Quarterly Bulletin. 42 (2): 350-372

Suchsland, 0. and H. Xu. 1989. A simulation of the horizontal density distribution in a flakeboard. Forest Products Journal. 39 (5): 29-33.

Talbott, J. W. and T. M. Maloney. 1957. Effect of several production variables on modulus of rupture and internal bond strength of board made from green Douglas-fir planer shavings. Forest Products Journal. 7 (10): 359-399

Torquato, S. and G. Stell. 1985. Microstructure of two-phase random media. V. The n-point matrix probability functions for impenetrable spheres. The Journal of Chemical Physics. 82 (2): 980-987.

Van den Akker, J. A. 1962. Some theoretical considerations on the mechanical properties of fibrous structures in *Formation and Structure of Paper, Transactions of 1961 Symposium,* Vol. I. British Paper and Board Makers Assoc., London. 535pp.

Van Wyk, C. M. 1946. Note on the compressibility of wool. Journal of Textile Institute. *37: 285-292.*

Wang, H. and S. Shaler. 1998. A three-dimensional microstructural model of wood fiber composite materials. Journal of Pulp and Paper Science. 24 (10): 3 14-3 19.

Wang, K. and F. Lam. 1999. Quadratic RSM models of processing parameters for three-layer oriented flakeboards. Wood and Fiber Science. 3 1 (2): 173-l 86.

Xu, W. and P. R. Steiner. 1995. A statistical characterization of the horizontal density distribution in flakeboard. Wood and Fiber Science. 27 (2): 160-l 67.

Yiannos, P. N. 1964. The apparent cell-wall density of wood and pulp fibers. Tappi. 47 (8): 468-47 1.

APPENDIX: SOURCE CODE (partial)

A-l cyl_grt.h - Definition of Cylinder Generator

```
#ifndef CYL_GRT_H
#define CYL_GRT_H

#ifndef  M-PI
#define   M-PI   3.14159265358979323846
#endif

class CylinderType
{
  public:
     CylinderType();
     ~CylinderType();
     float  Generate(float,  float,  float,  float,  float,  float,  float);
     float  length:       //length  of  cylinder
     float  width;        //width of flattern cylinder cross section
     float  thick;        //height of flattern cylinder cross section
     float   w-w-ratio;   //wallthick to width ratio
};
#endif
```

A-2 cyl_grt.C - Implementation of Cylinder Generator

```
#include "../MyLib/ran_num.h"
#include  "cyl_grt.h"
CylinderType::CylinderType()
{
  length=0;
  width=0;
  thick=0;
}

CylinderType::~CylinderType()
{
```

```
float CylinderType::Generate(float lmean, float lv,
                             float dmean, float dv,
                             float wmean, float wv,
                             float aspect)
{
   float vlm;
   float diameter:
   float wallthick;
   float crossarea;
   length=0;
   diameter=0;
   wallthick=0;
   // generate cylinder length with lognormal distribution function
   while(length<=0)
      length=LogNormal(lmean, lv);
   // generate cylinder width with normal distribution function
   while(diameter<=0)
      diameter=Normal(dmean, dv);
   wallthick=0.5*diameter*Normal(wmean, wv);
   width=M_PI*(diameter-2.0*wallthick)/(aspect*M_PI+2.0-
              2.0*aspect)+2.0*wallthick;
   crossarea=M_PI*wallthick*(diameter-wallthick); // cross section area
   thick=aspect*(width-2.0*wallthick)+2.0*wallthick;
   vlm=crossarea*length;
   w_w_ratio=wallthick/width;
   return vlm;
}
```

A-3 cbd_grt.h - Denfinition of Cuboid Generator

```
#ifndef CBD_GRT_H
#define CBD_GRT_H

class CuboidType
{
  public:
     CuboidType();
     ~CuboidType();
     float  Generate (float, float, float, float, float, float);
     float  length;
     float  width;
     float  thick;
};
#endif
```

A-4 cbd_grt.C - Implementation of Cuboid Generator

```
#include "../MyLib/ran_num.h"
#include  "cbd_grt.h"
```

```
CuboidType: :CuboidType()
{
   length=0;
   width=0;
   thick=0;
}

CuboidType: :~CuboidType()
{
}

float  CuboidType: :Generate(float lmean, float lv, float wmean, float wv,
                             float tmean, float tv)
{
   length=0;
   width=0;
   thick=0;
   // generate cuboid length with lognormal distribution function
   while(length<=0)
      length=Normal(lmean, lv);
   // generate cuboid width with normal distribution function
   while(width<=0)
      width=Normal(wmean, wv);
   while(thick<=0)
      thick=Normal(tmean, tv);
   return(length*width*thick);
}
```

A-5 xs_op.h - Definition of Horizontal Projection of Element's Axis (PXS)

```
#ifndef XS_OP_H
#define  XS_OP_H
class ElmntXs
{
   public:
      ElmntXs();
      void Locate(float, f loat);
      void Release(int[2][2]);
   private:
      int line [2][2];
};
#endif
```

A-6 xs_op.C - Implementation of Horizontal Projection of Element's Axis (PXS)

```
#include "xs_op.h"
#include "../MyLib/ran_num.h"
#include  <X11/Intrinsic.h>
#include <math.h>
```

```
#include <iostream.h>

Boolean FormingBox(int x0, int y0, float L, int line[2][2],
                   int xl, int xr, int yb, int yf)
{
   float x, y;
   float l2=L*L;
   if(line[0][1]<=line[1][1])
      {
        if(line[0][0]<xl)
           {
             line[1][0]=line[1][0]-(xl-line[0][0]);
             line[0][0]=xl;
             y=sqrt(l2-line[1][0]*line[1][0]);
             line[0][1]=(int)(y0-0.5*y+0.5);
             line[1][1]=(int)(y0+0.5*y+0.5);
           }
        else if (line[1][0]>xr)
           {
             line[0][0]=line[0][0]+(line[1][0]-xr);
             line[1][0]=xr;
             y=sqrt(l2-(xr-line[0][0])*(xr-line[0][0]));
             line[0][1]=(int)(y0-0.5*y+0.5);
             line[1][1]=(int)(y0+0.5*y+0.5);
           }
        else if(line[0][1]<yb)
           {
             line[1][1]=line[1][1]-(yb-line[0][1]);
             line[0][1]=yb;
             x=sqrt(l2-line[1][1]*line[1][1]);
             line[0][0]=(int)(x0-0.5*x+0.5);
             line[1][0]=(int)(x0+0.5*x+0.5);
           }
        else if(line[1][1]>yf)
           {
             line[0][1]=line[0][1]+(line[1][1]-yf);
             line[1][1]=yf;
             x=sqrt(l2-(yf-line[0][1])*(yf-line[0][1]));
             line[0][0]=(int)(x0-0.5*x+0.5);
             line[1][0]=(int)(x0+0.5*x+0.5);
           }
        else;
      }
   else
      {
        if(line[0][0]<xl)
           {
             line[1][0]=line[1][0]-(xl-line[0][0]);
             line[0][0]=xl;
             y=sqrt(l2-line[1][0]*line[1][0]);
             line[0][1]=(int)(y0+0.5*y+0.5);
             line[1][1]=(int)(y0-0.5*y+0.5);
           }
```

```
        else if (line [1] [0]>xr)
           {
              line [0] [0] =line [0] [0] + (line [1] [0] -xr);
              line [1] [0] =xr;
              y=sqrt (12- (xr-line[0][0])*(xr-line[0][0]));
              line [0] [1]=(int)(y0+0.5*y+0.5);
              line[1] [1]=(int) (y0-0.5*y+0.5);
           }
        else if (line [1][1]<yb)
           {
              line[0] [1] =line[0] [1]·(yb-line[1][1]);
              line [1] [1] =yb;
              x=sqrt(12-line[0] [1]*line [0][1]);
              line[0] [0]=(int)(x0-0.5*x+0.5);
              line[1] [0]=(int) (x0+0.5*x+0.5);
           }
        else if (line[0][1]>yf)
           {
              line[1] [1]=line[1] [1]+(line [0][1] -yf);
              line [0][1]=yf;
              x=sqrt(12-(yf-line[1] [1])*(yf-line[1][1]));
              line[0] [0]=(int) (x0-0.5*x+0.5);
              line[1] [0]=(int) (x0+0.5*x+0.5);
           }
        else:
     }
   if(line[0] [0]>=xl&&line[0] [0]<=xr&&line[1] [0]>=xl&&line[1][0]<=xr&&
       line[0] [1]>=yb&&line[0] [1]<=yf&&line[1] [1]>=yb&&line[1][1]<=yf)
       return  TRUE;
   else
      return  FALSE;
}

Boolean SampleEdge(int line[2][2], int xl, int xr, int yb, int yf)
{
   int A=line[1][0]-line[0] [0];
   int B=line[1][1]-line[0][1] ;
   int C=line[1][1]*line[0][0]-line[0][1] *line[1][0];
   if(!(line[0][0]<xl&&line[1][0]<xl)&&!(line[0] [0]>xr&&line[1][0]>xr))
      {
        if(line[0] [0]<xl&&(line[1] [0]<xr&&line[1][0]>xl))
           {
              line[0] [1]=(B*xl-C)/A;
              line[0] [0]=xl;
           }
        else if((line [0][0]>xl&&line [0][0]<xr)&&line [1][0]>xr)
           {
              line[1] [1]=(B*xr-C)/A;
              line[1] [0]=xr;
           }
        else if(line[0] [0]<xl&&line[1][0]>xr)
           {
              line[0] [1]=(B*xl-C)/A;
```

```
            line[1][1]=(B*xr-C)/A:
            line[0][0]=xl;
            line[1][0]=xr;
         }
      else
         ,
      if(!(line[0][1]<yb&&line[1][1]<yb)&&
          !(line[01[1]>yf&&line[1][1]>yf))
         {
            if ( (line[0][1]>yb&&line[0][1]<yf)&&line[1][1]<yb)
               {
                  line[1][0]=(A*yb+C)/B;
                  line[1][1]=yb;
                  return  TRUE;
               }
            else if ( (line[1][1]>yb&&line[1][1]<yf) &&line[0][1]<yb)
               {
                  line[0][0]=(A*yb+C)/B;
                  line[0][1]=yb;
                  return  TRUE;
               }
            else if(line[1][I]>yf&& (line[0][1]<yf&&line[0][1]>yb))
               {
                  line[1][0]=(A*yf+C)/B;
                  line[1][1]=yf;
                  return  TRUE:
               }
            else if (line[0][1]>yf&& (line[1][1]<yf&&line[1 1[1]>yb))
               {
                  line[0][0]=(A*yf+C)/B;
                  line[0][1]=yf;
                  return  TRUE;

            else if (line[0][1]>yf&&line[1][1]<yb)
               {
                  line[0][0]=(A*yf+C)/B;
                  line[1][0]=(A*yb+C)/B;
                  line[0][1]=yf;
                  line[1][1]=yb;
                  return  TRUE;
               }
            else if (line[1][1]>yf&&line[0][1]<yb)
               {
                  line[0][0]=(A*yb+C)/B;
                  line[1][0]=(A*yf+C)/B;
                  line[0][1]=yb;
                  line[1][1]=yf;
                  return  TRUE:
               }
            else
               return  TRUE;


return  FALSE;
```

```
}

ElmntXs::ElmntXs()
{
   line[0][0]=0;
   line[0][1]=0;
   line[1][0]=0;
   line[1][1]=0;
}

void ElmntXs::Locate(float w, float 1)
{
   extern int xsize, ysize;
   extern float edge;
   extern float lmean;
   extern float rsl;
   extern float concentrate:
   extern int idnum, n;
   extern int num, v_num;
   extern char dist_type;
   int x-n, y-n;
   float xd, yd;
   float x_y_ratio;
   int x0, y0;
   float angle;
   int box-l, box-r, box-b, box-f;
   int sample-l, sample-r, sample-b, sample-f;
   float tl, t2;
   box_l=(int)(-edge/rsl+0.5*w+0.5);
   box_r=(int)(xsize+edge/rsl-0.5*w+0.5);
   box_b=(int)(-edge/rsl+0.5*w+0.5);
   box_f=(int)(ysize+edge/rsl-0.5*w+0.5);
   sample_l=(int)(-0.5*w-0.5);
   sample-r=(int) (xsize+0.5*w+0.5);
   sample-b=(int) (-0.5*w-0.5);
   sample_f=(int)(ysize+0.5*w+0.5);
   // generate cuboid orientation with von Mises distribution function
   angle=von_Mises(0.0, concentrate, Bessel(concentrate));
   do
      {
        if(dist_type=='2')
           {
             x_y_ratio=(float)(xsize-0.5*lmean/rsl)
                       /(float)(ysize-0.5*lmean/rsl);
             if(num==0)
               x_n=(int)sqrt(v_num*x_y_ratio);
             else
               x_n=(int)sqrt(num*x_y_ratio);
               y_n=(int)(x_n/x_y_ratio);
               xd=(float)(xsize-0.5*lmean/rsl)/(float) (x-n-1);
               yd=(float)(ysize-0.5*lmean/rsl)/(float) (Y-n-l);
               x0=(int)((0.25*lmean/rsl)+(idnum%x_n)*xd);
               y0=(int)((0.25*lmean/rsl)+(int)(idnum/x_n)*yd);
```

```
        }
      else
        {
          x0=(int)(Uniform((float)(-edge/rsl+0.5*w+0.5),
                   (float)(xsize+2*edge/rsl+w+0.5))+0.5);
          y0=(int)(Uniform((float)(-edge/rsl+0.5*w+0.5),
                   (float)(ysize+2*edge/rsl+w+0.5))+0.5);
        }
      t1=cos(angle);
      t2=sin(angle);
      line[0][0]=x0-(int)(0.5*l*t1+0.5);
      line[1][0]=x0+(int)(0.5*l*t1+0.5);
      line[0][1]=y0-(int)(0.5*l*t2+0.5);
      line[1][1]=y0+(int)(0.5*l*t2+0.5);
      while(!FormingBox(x0, y0, 1, line, box-l, box-r, box-b, box-f))
        {
          x0=(int)(Uniform((float)(-edge/rsl+0.5*w+0.5),
                   (float)(xsize+2*edge/rsl+w+0.5))+0.5);
          yO=(int)(Uniform((float)(-edge/rsl+0.5*w+0.5),
                   (float)(ysize+2*edge/rsl+w+0.5))+0.5);
          t1=cos(angle);
          t2=sin(angle);
          line[0][0]=x0-(int)(0.5*l*t1+0.5);
          line[1][0]=x0+(int)(0.5*l*t1+0.5);
          line[0][1]=y0-(int)(0.5*l*t2+0.5);
          line[1][1]=y0+(int)(0.5*l*t2+0.5);
        }
      n++;
    }while(!SampleEdge(line, sample-l, sample-r, sample-b, sample-f));
    idnum++;
    return:


void  ElmntXs::Release(int axis[2][2])
{
  axis[0][0]=line[0][0];
  axis[0][1]=line[0][1];
  axis[1][0]=line[1][0];
  axis[1][1]=line[1][1];
  return;
}
```

A-7 prj_op.h - Definition of Horizontal Projection of Element (PRJ)

```
#ifndef PRJ_OP_H
#define PRJ_OP_H
#include  "global_var.h"
#include "xs_op.h"

struct PrjNdType
{
```

```
  int x;
  int y;
};

class ElmntPrj
{
  public:
    ElmntPrj();
    void  Locate(float,  float);
    void Size(int&, int&);
    void SetOffNd(int &x, int &y, int j, int k);
    void Cleanup();
  private:
    PrjNdType (*prj)[MAX_SZ];
    int w;
    int 1;
    ElmntXs xs;
};
#endif
```

A-S prj_op.C - Implementation of Horizontal Projection of Element (PRJ)

```
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include  "prj_op.h"
#include "../MyLib/transform.h"

void EndAdjust(int pt1[], int pt2[], int pt[] , int &n, float g)
{
  float fpt[2];
  int  testyl,  testy2;
  PlaneLineType line1, line2;
  line1.gradient=-1.0/g;
  line1.yintercept=pt1[1]-line1.gradient*pt1[0];
  line2.gradient=line1.gradient;
  line2.yintercept=pt2[1]-line2.gradient*pt2[0];
  fpt[0]=pt[0];
  fpt[1]=pt[1];
  if(g>0)
    {
      testy1=(int)(line1.gradient*pt[0]+line1.yintercept+0.5);
      testy2=(int)(line2.gradient*pt[0]+line2.yintercept+0.5);
      if(testy1>pt[1])
        I
          PTranslate(pt2 [0]-ptl [0],pt2 [1]-pt1 [1],fpt);
          n++;
        }
      if(testy2<pt[1])
        {
          PTranslate(pt1 [0]-pt2 [0],pt1 [1]-pt2 [1], fpt);
```

```
            n--;
          }
      }
   if(g<0)
      {
        testy1=(int)(linel.gradient*pt [0]+line1.yintercept+0.5);
        testy2=(int)(line2.gradient*pt[0] +line2.yintercept+0.5);
        if(testy1<pt[1])
          {
            PTranslate(pt2 [0] -pt1 [0], Pt2 [1]-pt1[1], fpt);
            n++;
          }
        if(testy2>pt[1])
          {
            PTranslate(pt1[01-pt2[01 , pt1[]-pt2 [1], fpt);
            n--;
          }
      }
   pt[0]=(int)(fpt[0]+0.5);
   pt[1]=(int)(fpt[1]+0.5);
   return;
}

ElmntPrj ::ElmntPrj()
   :xs 0
{
   w=1;
   1=1;
   prj=new PrjNdType[w] [MAX-SZ] ;
}

void ElmntPrj ::Locate(float  wd,  float  lng)
{
   PrjNdType  (*tmp)  [MAX-SZ];
   int j, k;
   int axis[2][2];        // axis
   float  gradient:       // gradient of projection on X-Y plane
   int *edline;           // end line of a projection
   int (*ctline)[2];      // axis line of a projection
   int (*scline)[2];      // line for scanning a projection
   xs.Locate(wd,   lng);
   xs.Release(axis);
   if(axis[1][0]!=axis[0] [0])
     gradient=(float)(axis[1][1]-axis[0][1] ) / (float) (axis[1][0]
              -axis[0][0]);
   if(abs(axis[1] [0] -axis [0][0])>=abs(axis[1][1] -axis [0][1]))
      {
        1=abs (axis [1][0]-axis [0][0]);
        w=(int)(1.0/cos(atan(gradient))*wd+0.5);
        edline=new int[w] ;        // initialize end line
        ctline=new int [1][2];     // initialize central line
        scline=new int [w][2];     // initialize scanning line
        tmp=new PrjNdType [w][MAX_SZ];
```

```
        for(j=0; j<w; j++)
          edline[j]=0;
        for(k=0; k<1; k++)
          {
            ctline[k][0]=axis[0][0]+k;
            if(gradient>=0)
              ctline[k][1]=axis[0][1]+(int)(k*gradient+0.5);
            else
              ctline[k][1]=axis[0][1]+(int)(k*gradient-0.5);
            for(j=0; j<w; j++)
              {
                scline[j][0]=ctline[k][0];
                scline[j][1]=(int)(ctline[k][1]-w/2+j+0.5);
                if(gradient!=0)
                    EndAdjust(axis[0], axis[1], scline[j], edline[j],
gradient);
                tmp[j][k].x=scline[j][0];
                tmp[j][k].y=scline[j][1];
              }
          }
      }
    else if(axis[1][0]==axis[0][0])
      {
        l=abs(axis[1][1]-axis[0][1]);
        w=(int)(wd+0.5);
        edline=new int[w];        // initialize end line
        ctline=new int[1][2] ;    // initialize central line
        scline=new int[w][2];     // initialize scanning line
        tmp=new PrjNdType[w][MAX_SZ];
        for(j=0; j<w; j++)
          edline[j]=0;
        for(k=0; k<1; k++)
          I
            if(axis[0][1]<axis[1][1])
              {
                ctline[k][0]=axis[0][0];
                ctline[k][1]=axis[0][1]+k;
              }
            else
              {
                ctline[k][0]=axis[0][0];
                ctline[k][1]=axis[0][1]-k;
              }
            for(j=0; j<w; j++)
              ‘
                scline[j][0]=(int)(ctline[k][0]-w/2+j+0.5);
                scline[j][1]=ctline[k][1];
                tmp[j][k].x=scline[j][0];
                tmp[j][k].y=scline[j][1];
              }
          }
      }
    else
      {
```

```
        l=abs(axis[1][1]-axis[0][1]);
        w=(int) (1.0/fabs(sin(atan(gradient)))*wd+0.5);
        edline=new int[w];
        ctline=new int[l][2];
        scline=new int[w][2];
        tmp=new PrjNdType[w][MAX_SZ];
        for(j=0; j<w; j++)
          edline[j]=0;
        for(k=0; k<l; k++)
           {
             if(axis[0][1]<axis[1][1])
               {
                 ctline[k][0]=axis[0][0]+(int)(k/gradient+0.5);
                 ctline[k][1]=axis[0][1]+k;
               }
             else
               {
                 ctline[k][0]=axis[0][0]+(int)(-k/gradient+0.5);
                 ctline[k][1]=axis[0][1]-k;
               }
             for(j=0; j<w; j++)
               {
                 scline[j][0]=(int)(ctline[k][0]-w/2+j+0.5);
                 scline[j][1]=ctline[k][1];
                 EndAdjust(axis[0], axis[1], scline[j], edline[j], gra
                           dient);
                 tmp[j][k].x=scline[j][0];
                 tmp[j][k].y=scline[j][1];
               }
           }
      }
   prj=new PrjNdType[w][MAX-SZ];
   for(k=0; k<l; k++)
      for(j=0; j<w; j++)
        {
          int i;
          i=k+edline[j];
          if(i>=l)
            i=i-l;
          if(i<0)
            i=i+l;
          prj[j][k].x=tmp[j][i].x;
          prj[j][k].y=tmp[j 1[i].y;
        }
   delete[] tmp;
   delete[] edline;
   delete[] scline;
   delete[] ctline;
   return:
}

void  ElmntPrj::Size(int &srfc_w, int &srfc_l)
{
```

```
   srfc_w=w;
   srfc_l=l;
   return;
}

void ElmntPrj ::SetOffNd(int &x, int &y, int j, int k)
{
   x=prj [j] [k] .x;
   y=prj [j] [k] .y;
   return;
}

void ElmntPrj::Cleanup()
{
   delete[] prj;
   return:
}
```

A-9 srfc_op.h - Definition of Surface of Element (SFE)

```
#ifndef  SRFC-OPH
#define  SRFC-OPH
#include <X11/Intrinsic.h>
#include  "global_var.h"
#include  "prj_op.h"

// definition of a node on the surface
struct SrfcNdType
{
   int x;
   int y;
   int z;
   int  touch;  //recording  the  ID  of  the  element  being  contacted
};

class ElmntSrfc
{
   public:
      ElmntSrfc();
      void Locate(float wd, float lng, float Z);
      void  Smooth(float  El,  float  Et,  float  vh,  float  p);
      void Size(int&, int&) ;
      void SetOffNd(int &x, int &y, int &z, int &touch, int j, int k);
      void Cleanup();
   private:
      SrfcNdType (*srfc) [MAX_SZ];
      int w;
      int 1;
      ElmntPrj prj;
};
#endif
```

A-10 srfc_op.C - Implementation of Surface of Element (SFE)

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
#include <X11/Intrinsic.h>
#include  "srfc_op.h"

float  PlaneDst(int  xl,  int  yl,  int  x2,  int  y2)
I
   float d=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
   return  d;
}
void ConvexAdjust(int  &h1, int  &h2, int  &h3, int  &t1, int  &t2, int  &t3,
                    float  dst,  float  R,  Boolean  &lift)
{
   float  x;
   float  r_sq, x_sq, d_sq;
   int  sl,  s2;
   int a,  b;
   int e;
   if(R<dst)
     R=dst;
   a=(int)((R-sqrt(R*R-dst*dst))+0.5);
   b=(int)(sqrt(R*R-(R-dst)*(R-dst))+0.5);
   if(h1-h2>b||h2-h3>b)
      {
        if(h1-h2>b)
           {
             h2=h1-b;
             t2=0;
             lift=FALSE;
           }
        if(h2-h3>b)
           {
             h3=h2-b;
             t3=0;
             lift=FALSE;


   else
      {
        s1=h2-h1;
        s2=h3-h2;
        if(s1>s2)
           {
             float zl, z2;
             float A;
             if(s1<=0&&s2<0)
                {
```

```
            d_sq=dst*dst+(h1-h2)*(h1-h2);
            A=0.25*d_sq;
            if(R*R>A)
               {
                  r_sq=R*R-0.25*d_sq;
                  x_sq=((h1-h2)*(h1-h2)/d_sq)*r_sq;
                  x=sqrt(x_sq);
                  if(R>x+1.5*dst)
                     {
                        z1=sqrt(R*R-(x+0.5*dst)*(x+0.5*dst));
                        z2=sqrt(R*R-(x+1.5*dst)*(x+1.5*dst));
                        e=(int)((z1-z2)+0.5);
                        if((h2-h3)>e)
                           {
                              h3=h2-e;
                              t3=0;
                              lift=FALSE;



         }
else if(s1>0&&s2<0)
   {
      if(h2-h1<=a&&h2-h3>a)
         {
            d_sq=dst*dst+(h1-h2)*(h1-h2);
            A=0.25*d_sq;
            if(R*R>A)
               {
                  r_sq=R*R-0.25*d_sq;
                  x_sq=((h1-h2)*(h1-h2)/d_sq)*R*R;
                  x=sqrt(x_sq);
                  if(R>x+1.5*dst)
                     {
                        z1=sqrt(R*R-x_sq);
                        z2=sqrt(R*R-(x+dst)*(x+dst));
                        e=(int)((z1-z2)+0.5);
                        if((h2-h3)>e)
                           {
                              h3=h2-e;
                              t3=0;
                              lift=FALSE;
                           }
                     }
               }
         }
      else if(h2-h1>a&&h2-h3>a)
         {
            h3=h2-a;
            t3=0;
            lift=FALSE;
         }
      else;
```

```
    }
  return;
}

void ConcaveAdjust(int &h1, int &h2, int &h3,
                   int &t1, int &t2, int &t3,
                   float dst, float R, Boolean&lift)
{
  float x;
  float r_sq, x_sq, d_sq;
  int sl, s2;
  int b;
  int e;
  if(R<dst)
    R=dst;
  b=(int)(sqrt(R*R-(R-dst)*(R-dst))+0.5);
  if(h1-h2>b||h3-h2>b)
    I
      if(h1-h2>b)
        {
          h2=h1-b;
          t2=0;
          lift=FALSE;
        }
      if(h3-h2>b)
        I
          h2=h3-b;
          t2=0;
          lift=FALSE;
        }
    }
  else
    {
      s1=h2-h1;
      s2=h3-h2;
      if(s1<s2)
        {
          float zl, z2;
          d_sq=4*dst*dst+(h1-h3)*(h1-h3);
          r_sq=R*R-0.25*d_sq;
          x_sq=(h1-h3)*(h1-h3)*r_sq/d_sq;
          x=sqrt(x_sq);
          if(h1>=h3)
            {
              z1=sqrt(R*R- (x+dst)*(x+dst));
              z2=sqrt(R*R-x*x);
              e=(int) (z2-z1+0.5);
              if((h1-h2)>e)
                {
                  h2=h1-e;
                  t2=0;
```

```
                    lift=FALSE;
                  }
              }
            else
              {
                z1=sqrt(R*R-(x+dst)*(x+dst));
                z2=sqrt(R*R-x*x);
                e=(int)(z2-z1+0.5);
                if((h3-h2)>e)
                  {
                    h2=h3-e;
                    t2=0;
                    lift=FALSE;
    return:
}

ElmntSrfc::ElmntSrfc()
  : prj 0
{
  w=1;
  l=1;
  srfc=new SrfcNdType[w] [MAX-SZ] ;
}

void ElmntSrfc::Locate(float wd, float lng, float Z)
{
  extern GlbSrfcNdType (*surface) [MAX-SZ];
  extern int xsize, ysize;
  int j, k;
  int X, Y;
  prj.Locate(wd, lng);
  prj.Size(w, 1) ;
  srfc=new SrfcNdType[w] [MAX-SZ];
  for(k=0; k<l; k++)
    for(j=0; j<w; j++)
      {
        prj.SetOffNd(srfc[j] [k].x, srfc[j] [k].y, j, k);
        srfc[j] [k].z=(int)(Z+0.5);
        srfc[j] [k].touch=0;
        X=srfc[j] [k].x;
        Y=srfc[j] [k].y;
        if((X<xsize&&X>=0)&&(Y<ysize&&Y>=0))
          I
            if(surface[X] [Y] .height==0)
              srfc[j] [k].z=srfc[j][k] .z+surface[X][Y].height;
            else
              srfc[j] [k].z=srfc[j][k] .z+surface[X][Y].height-1;
            srfc[j] [kl .touch=surface[X][Y].id;
          }
```

```
        }
    prj.Cleanup();
    return;
}

void ElmntSrfc::Smooth(float R1, float Rt, float vh, float p)
{
    int j, k;
    float a;
    Boolean L, 11, 12, 13, 14;

    L=FALSE;
    while(L==FALSE)
      {
        11=TRUE;
        12=TRUE;
        13=TRUE;
        14=TRUE;
        for(k=0; k<1; k++)
          for(j=0; j<w-2; j++)
            {
              a=0.5*vh*PlaneDst(srfc[j] [k] .x,srfc[j][k].y,
                                srfc [ j+2][k].x, srfc [ j+2][k].y) ;

              ConcaveAdjust(srfc[j] [k].z, srfc[j+1][k].z,
                            srfc[j+2][k].z, srfc[j][k].touch,
                            srfc[j+1] [k] .touch, srfc[j+2][k].touch,
                            a, Rt*vh, 11);
            }
        for(k=0; k<1; k++)
          for(j=w-1; j>=2; j--)
            {
              a=0.5*vh*PlaneDst(srfc[j] [k].x, srfc[j][k].y,
                                srfc[j-2] [k].x,srfc[j-2][k].y);

              ConcaveAdjust(srfc[j] [k] .z,srfc[j-1][k].z,
                            srfc[j-2][k].z, srfc[j][k].touch,
                            srfc[j-l][k].touch, srfc[j-2] [kl.touch,
                            a, Rt*vh, 12);
            }
        for(k=0; k<1; k++)
          for(j=0; j<w-2; j++)
            {
              a=0.5*vh*PlaneDst(srfc[j] [k].x, srfc[j][k].y,
                                srfc[j+2] [k].x,srfc[j+2][k].y);

              ConvexAdjust(srfc[j] [k].z, srfc[j+1][k].z,
                           srfc[j+2][k].z, srfc[j] [k].touch,
                           srfc[j+1] [k] .touch, srfc[j+2][k].touch,
                           a, Rt*vh, 13);
            }
        for(k=0; k<1; k++)
          for(j=w-1; j>=2; j--)
```

```
        {
          a=0.5*vh*PlaneDst(srfc[j][k].x, srfc[j][k].y,
                            srfc[j-21 [k].x, srfc[j-2][k].y);

          ConvexAdjust(srfc[j] [k].z, srfc[j-1][k].z,
                       srfc[j-2][k].z, srfc[j][k].touch,
                       srfc[j-l][k].touch,   srfc[j-2][k].touch,
                       a, Rt*vh, 14);
        }
    L=l1&&l2&&l3&&l4;
  }
L=FALSE;
while(L==FALSE)
  {
    l1=TRUE;
    l2=TRUE;
    l3=TRUE;
    l4=TRUE;
    for(j=0; j<w; j++)
      for(k=0; k<l-2; k++)
        {
          a=0.5*vh*PlaneDst(srfc[j] [k].x, srfc[j][k].y,
                            srfc[j][k+2].x, srfc[j][k+2].y);

          ConcaveAdjust(srfc[j][k].z, srfc[j][k+1].z,
                        srfc[j][k+2].z, srfc[j][k].touch,
                        srfc[j][k+l].touch,  srfc[j] [k+2].touch,
                        a, Rl*vh, l1);
        }
    for(j=0; j<w; j++)
      for(k=l-1; k>=2; k--)
        {
          a=0.5*vh*PlaneDst(srfc[j] [k].x, srfc[j][k].y,
                            srfc[j][k-2].x, srfc[j][k-2].y);

          ConcaveAdjust(srfc[j] [k].z, srfc[j][k-1].z,
                        srfc[j][k-2].z, srfc[j][k].touch,
                        srfc[j][k-1].touch, srfc[j] [k-2].touch,
                        a, Rl*vh, 12);
        }
    for(j=0; j<w; j++)
      for(k=0; k<l-2; k++)
        {
          a=0.5*vh*PlaneDst(srfc[j] [k].x, srfc[j][k].y,
                            srfc[j][k+2].x, srfc[j][k+2].y);

          ConvexAdjust(srfc[j][k].z, srfc[j] [k+1].z,
                       srfc[j][k+2].z, srfc[j][k].touch,
                       srfc[j][k+1].touch, srfc[j][k+2].touch,
                       a, Rl*vh, 13);
        }
    for(j=0; j<w; j++)
      for(k=l-1; k>=2; k--)
```

```
            {
              a=0.5*vh*PlaneDst(srfc[j][k].x, srfc[j][k].y,
                                srfc[j][k-2].x, srfc[j][k-2].y);

              ConvexAdjust(srfc[j][k].z, srfc[j][k-1].z,
                           srfc[j][k-2].z, srfc[j][k].touch,
                           srfc[j][k-1].touch, srfc[j][k-2].touch,
                           a, R1*vh, 14);
            }
        L=l1&&l2&&l3&&l4;
      }
   return;
}

void ElmntSrfc::Size(int &bd_w, int &bd_l)
{
   bd_w=w;
   bd_l=l;
   return;
}

void ElmntSrfc::SetOffNd(int &x, int &y, int &z, int &touch, int j, int
k)
{
   x=srfc[j][k].x;
   y=srfc[j][k].y;
   z=srfc[j][k].z;
   touch=srfc[j][k].touch;
   return;
}

void ElmntSrfc::Cleanup()
{
   delete[] srfc;
   return;
}
```

A-11 cyl_op.h - Definition of 3D Body of Cylindrical Element (BDE)

```
#ifndef  CYL_OP_H
#define  CYL_OP_H
#include  "global-var.h"
#include  "srfc_op.h"
#include <fstream.h>

struct CylNdType
{
   int x;
   int y;
   int z;
   int  touch;
   int os;
```

```
   int is;
};

class CylBd
{
   public:
      CylBd();
      void Locate(int, float, float, float, float, float, float, float,
                  float);
      void TrimEdge(int, int, int, int);
      void Store(ofstream&, float);
      void Cleanup();
   private:
      int cylID;
      CylNdType (*cyl)[MAX_SZ];
      int w;     // width
      int 1;     // length
      int hthk; // half thickness
      ElmntSrfc srfc;
};
#endif
```

A-12 cyl_op.C - Implementation of 3D Body of Cylindrical Element (BDE)

```
#include <fstream.h>
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include  "cylbd_op.h"
#include <X11/Intrinsic.h>
#include "../MyLib/transform.h"

#ifndef  M-PI
#define   M-PI   3.14159265358979323846
#endif

void HalfCrossSection(int w, int t, int wall, float h[][2])
{
   int i;
   int p, q, u, v;
   p=int(t/2+0.5);
   q=w-p;
   u=wall;
   v=w-wall;
   for(i=0; i<=u; i++)
      {
        h[i][0]=sqrt(p*p- (p-i)*(p-i));
        h[i][1]=0;
      }
   for(i=u+1; i<=p; i++)
      {
```

```
        h[i] [0]=sqrt(p*p- (p-i)*(p-i));
        h[i] [1]=sqrt((p-u)*(p-u)·(p-i)*(p-i));
      }
   for(i=p+1; i<q; i++)
      {
        h[i][0]=0.5*t;
        h[i][1]=0.5*t-wall;
      }
   for(i=q; i<v; i++)
      {
        h[i][0]=sqrt((w-q)*(w-q)-(i-q+1)*(i-q+1));
        h[i][1]=sqrt((v-q)*(v-q)-(i-q+1)*(i-q+1));
      }
   for(i=v; i<w; i++)
      {
        h[i][0]=sqrt((w-q)*(w-q)-(i-q+1)*(i-q+1));
        h[i][1]=0;
      }
   return;
}

CylBd::CylBd()
   : srfc()
{
   cylID=0;
   w=1;
   l=1;
   hthk=l;
   cyl=new CylNdType[w] [MAX-SZ];
}

void CylBd::Locate(int id, float width, float length, float thick,
                   float w-ratio, float vh, float flx_l, float flx_t,
                   float p)
{
   extern GlbSrfcNdType (*surface)[MAX_SZ1;
   extern int xsize, ysize;
   int j, k;
   int X, Y;
   int thk, wthk;
   float (*ht)[2];       // cross section outline
   cylID=id;
   thk=(int)(thick+0.5);
   hthk=thk/2;
   srfc.Locate(width, length, hthk*vh /*, surface*/);
   srfc.Smooth(flx_l, flx_t, vh, p);
   srfc.Size(w, 1);
   wthk=(int)(w_ratio*w+0.5);
   cyl=new CylNdType [w][MAX_SZ];
   ht=new float [w][2];
   HalfCrossSection(w, thk, wthk, ht);
   for(k=0; k<1; k++)
     for(j=0; j<w; j++)
```

```
        {
          srfc.SetOffNd(cyl[j] [k].x, cyl[j] [k].y, cyl[j] [k].z,
                        cyl[j] [k].touch, j, k);
          cyl[j][k].os=(int) (ht[j][0]+0.5);
          cyl[j] [k] .is=(int) (ht[j][1]+0.5);
        }
   for(k=0; k<l; k++)
     for(j=0; j<w; j++)
        {
          X=cyl[j][k].X;
          Y=cyl [j][k].y;
          if((X<xsize&&X>=0)&&(Y<ysize&&Y>=0))
             {
               suface[X] [Y].height=cyl[j][k] .z+(int) (cyl[j][k] .os*vh);
               surface[X] [Y] .num++;
               surface[X] [Y] .id=cylID;
             }
        }
   delete [] ht;
   srfc.Cleanup();
   return:
}

void CylBd::TrimEdge(int left,  int  right, int back,  int  front)
{
   int j, k;
   for(j=0; j<w; j++)
      {
        for(k=1; k<l; k++)
           {
             if(cyl[j] [k].x>right-1 || cyl[j] [k].x<left ||
               cyl[j] [k].y>front-1    cyl[j] [k].y<back)
               {
                 cyl[j][k].x =cyl[j][k-1].x;
                 cyl [j][k].y =cyl[j][k-1].y;
                 cyl[j] [k].z =cyl[j] [k-1].z;
                 cyl[j] [k].os=cyl[j] [k-1].os;
                 cyl[j] [k].is=cyl[j] [k-1].is;


        for(k=l-2;  k>=0;  k--)
           {
             if(cyl[j] [k].x>right-1    cyl[j] [k].x<left
               cyl[j] [k].y>front-1    cyl[j] [k].y<back)
               {
                 cyl [j][k].x =cyl[j][k+1].x;
                 cyl [j][k].y =cyl[j][k+1].y;
                 cyl[j] [k].z =cyl[j] [k+1].z;
                 cyl [j][k].os=cyl[j][k+1].os;
                 cyl[j] [k].is=cyl[j] [k+1].is;
               }
           }
      }
```

```
   for(k=0; k<1; k++)
      {
         for(j=1; j<w; j++)
            {
               if(cyl [j][k].x>right-1 || cyl[j] [k] .x<left ||
                  cyl[j] [k].y>front-1    cyl[j] [k].y<back)
                  {
                     cyl[j][k].x =cyl[j -1][k].x;
                     cyl [j][k].y =cyl[j -11 [k].y;
                     cyl [j][k].z =cyl[j -1][k].z;
                     cyl[j][k].os=cyl[j -11 [k].os;
                     cyl[j][k].is=cyl[j -1][k] .is;
                  ,
            }
         for(j=w-2;  j>=0; j--)
            {
               if(cyl[j] [k] .x>right-1    cyl[j] [k] .x<left
                  cyl[j] [k].y>front-1   cyl[j] [k] .y<back)
                  {
                     cyl[j] [k].x =cyl[j+1][k] .x;
                     cyl[j] [k].y =cyl [j+1][k].y;
                     cyl[j] [k].z =cyl[j+1][k] .z;
                     cyl[j][k].os=cyl[j+1][k].os;
                     cyl[j][k].is=cyl[j+1][k].is;


      }
   return:


void CylBd::Store(ofstream &oF, float vh)
{
   int i, j, k;
   // output cylinder volume
   if(w>1&&l>1)

        oF<<"ZONE T="<<"\"CYLINDER "<<cylID<<"\""<<endl;
        oF<<"I="<<2*w<<", "<<"J="<<l
          <<", "<<"K="<<hthk+1<<","<<"F=POINT"<<endl;
        oF<<"DT=(SINGLE SINGLE SINGLE SINGLE SINGLE)"<<endl;
        for(i=0; i<=hthk; i++)
          for(k=0; k<l; k++)
             {
                // output top half
                for(j=0; j<w; j++)
                   I
                     int h;
                     if(i<cyl[j][k].is)
                       h=cyl[j][k].is;
                     else if(i>cyl[j][k].os)
                       h=cyl[j][k].os;
                     else
                       h=i;
```

```
                    oF<<cyl[j][k].x<<' '<<cyl[j][k].y
                      <<' '<<(int)(cyl[j][k].z/vh)+h
                      <<' '<<cylID<<' '<<"0"<<endl;
                  }
               // output bottom half
               for(j=w-1; j>=0; j--)
                  {
                    int h;
                    if(i<cyl[j][k].is)
                      h=cyl[j][k].is;
                    else if(i>cyl[j][k].os)
                      h=cyl[j][k].os;
                    else
                      h=i;
                    oF<<cyl[j][k].x<<' '<<cyl[j][k].y
                      <<' '<<(int)(cyl[j][k].z/vh)-h
                      <<' '<<cylID<<' ';
                    if(i==hthk)
                      oF<<cyl[j][k].touch<<endl;
                    else
                    oF<<"0"<<endl;
                  }
            }
     }
  return;
}

void CylBd::Cleanup()
{
  delete []cyl;
  return:
}
```

A-13 cbd_op.h - Definition of 3D Body of Cuboid Element (BDE)

```
#ifndef CBD_OP_H
#define  CBD_OP_H
#include  "global_var.h"
#include  "srfc_op.h"
#include <fstream.h>

struct CbdNdType
{
  int x;
  int y;
  int z;
  int  touch;
};

class CbdBd
{
```

```
   public:
      CbdBd();
      void Locate(int, float, float, float, float, float, f loat, float);
      void TrimEdge(int, int, int, int);
      void Store(ofstream&, float);
      void Cleanup();
   private:
      int cbdID;
      CbdNdType (*cbd) [MAX-SZI;
      int w;     // width
      int 1;     // length
      int thk;   // thickness
      ElmntSrfc srfc;
};
#endif
```

A-14 cbd_op.C ▪ Implementation of 3D Body of Cuboid Element (BDE)

```
#include <fstream.h>
#include <stdlib.h>
#include <math.h>
#include  "cbd_op.h"
#include <X11/Intrinsic.h>
#include "../MyLib/transform.h"

#ifndef  M-PI
#define   M-PI   3.14159265358979323846
#endif

CbdBd::CbdBd()
   : srfc()
{
   cbdID=0;
   w=1;
   l=1;
   thk=l;
   cbd=new  CbdNdType[w] [MAX-SZI;
}

void  CbdBd::Locate(int  id,  float  width,  float  length,  float  thick,
                    float  vh,  float  El,  float  Et,  float  p)
{
   extern  GlbSrfcNdType  (*surface)  [MAX_SZ];
   extern  int  xsize,  ysize;
   int j, k;
   int X, Y;
   cbdID=id;
   srfc.Locate(width,   length,   0);
   srfc.Smooth(flx_l, flx_t, vh,  p);
   srfc.Size(w, 1);
   thk=(int)(thick+0.5);
```

```
   cbd=new CbdNdType[w][MAX_SZ];
   for(k=0; k<l; k++)
      for(j=0; j<w; j++)
         srfc.SetOffNd(cbd[j][k].x, cbd[j][k].y, cbd[j][k].z,
                       cbd[j][k].touch, j, k);
   for(k=0; k<l; k++)
      for(j=0; j<w; j++)
         {
            X=cbd[j][k].x;
            Y=cbd[j][k].y;
            if((X<xsize&&X>=0)&&(Y<ysize&&Y>=0))
               {
                  surface[X][Y].height=cbd[j][k].z+(int)(thk*vh);
                  surface[X][Y].num++;
                  surface[X][Y].id=cbdID;
               }
         }
   srfc.Cleanup();
   return:
}

void CbdBd::TrimEdge(int left, int right, int back, int front)
{
   int j, k;
   for(j=0; j<w; j++)
      {
         for(k=1; k<l; k++)
            {
               if(cbd[j][k].x>right-1 || cbd[j][k].x<left ||
                  cbd[j][k].y>front-1   cbd[j][k].y<back)
                  {
                     cbd[j][k].x =cbd[j][k-1].x;
                     cbd[j][k].y =cbd[j][k-1].y;
                     cbd[j][k].z =cbd[j][k-l].z;


         for(k=l-2; k>=0; k--)
            {
               if(cbd[j][k].x>right-1    cbd[j][k].x<left
                  cbd[j][k].y>front-1    cbd[j][k].y<back)
                  {
                     cbd[j][kl.x =cbd[j][k+1].x;
                     cbd[j][k].y =cbd[j][k+1].y;
                     cbd[j][k].z =cbd[j][k+1].z;
                  }
            }
      }
   for(k=0; k<l; k++)
      {
         for(j=1; j<w; j++)
            {
               if(cbd[j][k].x>right-1    cbd[j][k].x<left
                  cbd[j][k].y>front-1  | cbd[j][k].y<back)
```

```
                {
                  cbd[j][k].x =cbd[j-11[k].x;
                  cbd[j][kl .y =cbd[j-1][k].y;
                  cbd[j][k].z =cbd[j-1][k].z;
                }
            }
          for(j=w-2; j>=0; j--)
            {
              if(cbd[j][kl .x>right-1 || cbd[j][k].x<left ||
                  cbd[j][k].y>front-1    cbd[j][k].y<back)
                {
                  cbd[j][k].x =cbd[j+1][k].x;
                  cbd[j][k].y =cbd[j+1][k].y;
                  cbd[j][k].z =cbd[j+1][k].z;
                }


    return;
}

void CbdBd::Store(ofstream &oF, float vh)
{
    int i, j, k;
    if(w>1&&l>1)
      {
        oF<<"ZONE T="<<"\"CUBOID "<<cbdID<<"\""<<endl;
        oF<<"I="<<w<<", "<<"J="<<l<<", "<<"K="<<thk+1
          <<", "<<"F=POINT"<<endl;
        oF<<"DT=(SINGLE SINGLE SINGLE SINGLE SINGLE)"<<endl;
        for(i=0; i<=thk; i++)
          for(k=0; k<l; k++)
            for(j=0; j<w; j++)
              {
                oF<<cbd[j][k].x<<' '<<cbd[j][kl .y
                  <<' '<<(int)(cbd[j][k].z/vh)+i
                  <<' '<<cbdID<<' ';
                if(i==0)
                  oF<<cbd[j][k].touch<<endl;
                else
                  oF<<"0"<<endl;
              }
      }
    return;
}

void CbdBd::Cleanup()
{
    delete[] cbd;
    return;
}
```

BIOGRAPHY OF THE AUTHOR

Huaijun Wang was born in Nanjing, China, on October 3 1, 1964. He was raised there and graduated from Nanlin High School in 1982, and then attended Nanjing Forestry University. His major at the university was wood science and technology. In 1986, he graduated with a degree of Bachelor of Science in Engineering.

From August 1986 to December 1994, Huaijun was employed by The Research Institute of Wood Industry, Chinese Academy of Forestry in Beijing, China.

In 1995, Huaijun entered the Department of Forest Management, University of Maine as a graduate student. He graduated with a degree of Master of Science in Forestry in December, 1996.

Huaijun is a candidate for the Doctor of Philosophy degree in Forest Resources from The University of Maine in December, 2000.