COMPUTER MODELING FOR PROPAGATION OF ULTRASONIC WAVES IN SOLID POLYCRYSTALLINE STRUCTURES

A THESIS SUBMITTED TO
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ORCAN KOLANKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF METALLURGICAL AND MATERIALS ENGINEERING

DECEMBER 2002

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Tayfur ÖZTÜRK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Naci SEVİNÇ
Head of Department

We certify that we have read this thesis and in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Metallurgical and Materials Engineering.

Assoc. Prof. Dr. C. Hakan GÜR
Supervisor

Prof. Dr. A. Bülent DOYUM
Co-Supervisor

Examining Committee Members:

Prof. Dr. Alpay ANKARA

Prof. Dr. A.Bülent DOYUM

Prof. Dr. Bilgehan ÖGEL

Assoc. Prof. Dr. Kadri AYDINOL

Assoc. Prof. Dr. C. Hakan GÜR

ABSTRACT

COMPUTER MODELING FOR
PROPAGATION OF ULTRASONIC WAVES
IN SOLID POLYCRYSTALLINE STRUCTURES

KOLANKAYA, Orcan
M.Sc., Metallurgical and Materials Engineering Department
Supervisor: Assoc. Prof. Dr. C. Hakan GÜR
Co-Supervisor: Prof. Dr. A. Bülent Doyum

December 2002, 69 pages

Ultrasonic testing in industrial applications must perform ever more accurate and reliable characterization and sizing of the defects in the components having complex shapes. Computer simulation that has been increasingly used at the various stages of ultrasonic engineering is an effective way of determining or designing the appropriate transducer for a given specific application. By this way, the performances of methods in complex configurations may also be demonstrated at a lower cost than experimental approaches. The aim of the thesis is to establish a model to simulate the propagation of ultrasonic waves in polycrystalline materials.

In order to develop a model, scattering behavior of the polycrystalline materials should be generalized, to be able to apply this model to all sorts of polycrystalline media whose single-crystal anisotropy is not large. It is also valid for time-harmonic elastic waves with all ratios of grain size to wavelength. The proposed model is based on the studies of Stanke and Kino.

In the programming stage, Visual Basic 6.0© was used. The program mainly consists of two parameter groups. First group considers material variables such as elastic constants and density. The second group takes into account wave variables, i.e., frequency or wavelength. The computer program was designed to give an animated simulation for the wave propagation and to obtain numerical results for the total attenuation due to scattering. For the sake of simplicity, the losses due to beam divergence and absorption were not taken into consideration.

Improvements on this study may bring out various practical results such that the components having complex shapes can be exposed to a simulated ultrasonic inspection by locating defects at critical sections. Another practical application can be the ultrasonic inspection of composite materials with layered structures.

Keywords: Wave Propagation, Ultrasonic Scattering, Ultrasonic Wave Model, Computer Simulation.

ÖZ

KATI POLİKRİSTAL YAPILARDA ULTRASONİK DALGA HAREKETLERİNİN BİLGİSAYARDA MODELLENMESİ

KOLANKAYA, Orcan
M.Sc., Metallurgical and Materials Engineering Department
Tez Yöneticisi: Doç. Dr. C. Hakan GÜR
Yardımcı Tez Yöneticisi: Prof. Dr. A. Bülent Doyum

Aralık 2002, 69 Sayfa

Endüstriyel ultrasonik test uygulamalarında, karmaşık yapılara sahip parçalar için hataların karakterizasyonu ve boyutlarının saptanması çok daha güvenilir ve doğru olmalıdır. Ultrasonik uygulamaların çeşitli aşamalarında yaygınlaşarak artan bilgisayar simülasyonları, uygun prob seçimi veya tasarımlarında çok verimli yöntem oluştururlar. Bu şekilde, karmaşık yapılarda uygulanacak metotlar deneysel yöntemlerden çok daha az maliyetle denene bilinmektedir. Bu tezin amacı, ultrasonik dalgaların polikristal malzemeler içerisindeki hareketlerini modellemektir.

Polikristal malzemelerdeki saçılma davranışları genellendiği taktirde geliştirilen model her tür polikristal yapıya uygulanabilir ve her tane boyutu/frekans oranı için geçerli olur. Bu tezdeki model, Stanke ve Kino tarafından gerçekleştirilen çalışmalar referans alınarak oluşturulmuştur.

Programlama aşamasında Visual Basic 6.0© programlama dili kullanılmıştır. Program iki ana paramètre grubundan oluşmaktadır. Birinci grup yoğunluk ve elastik sabit gibi malzeme değişkenlerini, ikinci grup ise frekans ve

kayıplarını hesaplayarak, animatik bir dalga simülasyonu sergileyecek şekilde tasarlanmıştır. Modeli basitleştirmek amacıyla, ses demeti açılımı ve sönümlenmeden kaynaklanan kayıplar kapsam dışı bırakılmıştır.

Bu konudaki gelişmeler, kompleks yapılarda gerçekleştirilen ultrasonik testler için pratik sonuçlar doğurabilir. Bu yöntemle kritik bölgelerde bulunabilecek hatalar önceden modellenebilir. Bir diğer önemli uygulama da katmanlı yapılara sahip kompozit malzemeler üzerinde olacaktır.

Anahtar sözcükler : Dalga hareketi, Ultrasonik Saçılma, Dalga Modeli, Bilgisayar Benzetimi.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

1.1 Elastic Wave Propagation

The propagation of the waves through random media occurs in many forms such as electromagnetic w ave propagation through atmosphere, ultrasonic wave propagation through polycrystalline metals or human tissue, and seismic wave propagation through earth. As these waves propagate, they are altered by scattering due to scatters within the medium. Most of the time propagation becomes very complex because of multiple scattering effect.

Understanding the mechanism of the elastic wave propagation and the multiple scattering is very important for several purposes. If a defect or an object is going to be located in a random media by using the waves, one has to know the effect of randomness on the wave propagation. In addition, wave propagation through a media can contain essential information about the medium that can be used to help the researchers to understand material properties, such as elastic constants and etc,. Developing a general scattering formalism for ultrasonic elastic waves through polycrystalline structures and simulating them for some special materials by the help of computers is the subject of this thesis.

A polycrystalline material is composed of discrete grains, each having regular, crystalline atomic structure. The elastic properties of the grains are anisotropic, and their crystallographic axes are randomly oriented with respect to fixed axes. As a result the elastic constants of the material measured in the laboratory coordinate system are microscopically inhomogeneous, i.e., they vary

as a function of position within the medium. Acoustic wave propagation in such a medium is scattered and consequently has an amplitude attenuation coefficient and a phase velocity which vary with frequency of wave.

In ultrasonic non-destructive testing (NDT) use is made of the physical properties of elastic waves in solids in order to detect defects and material inhomogeneities. However, anisotropic elastic behavior of the grains leads to complicated wave propagation phenomena. In order to ensure the reliability of ultrasonic inspection techniques, these material properties and the influences of microstructure have to be taken into account. In this respect, simulation of ultrasonic inspections gained a considerable importance, where mathematical modeling provides an assisting for analysis. In order to develop a computer simulation for the ultrasonic elastic waves, one has to describe the wave propagation with numerical analysis.

An ultrasonic system used in the inspections can be modeled in three different parts. First part relates to the incident wave and piezoelectric effect, second part to the ultrasonic field propagation within material, and third part to the wave interaction with defects and geometrical limits of the component. Among these, the most important part is the ultrasonic field propagation for which various theories and numerical results were constructed and represented by various scientists in history of acoustics. After modeling the ultrasonic propagation through a single phase, anisotropic polycrystalline medium, the transducers with the equations of piezoelectricity and the interactions of the wave between defects or objects can be considered in the further studies.

When a wave passes through a random medium, the field quantities such as displacement, stress, or acoustic pressure, will have amplitudes and phases which fluctuate in space and time. These quantities may be written as

$$u = u_c + u_d = \langle u \rangle + u_d \tag{1.1}$$

Where the brackets, $\langle\rangle$, denote ensemble averages. The coherent field, u_c, is defined as mean or average field ($u_c = \langle u \rangle$). The incoherent field, u_d, is the fluctuating part of the field which has zero mean ($u_d = \langle 0 \rangle$). The coherent field is

characterized by its complex wave number. The real part of the wave number is the wave speed of the coherent field and the imaginary part is the attenuation. This attenuation here is not due to absorption mechanism, but energy scattered by the main beam.

In order to find the total attenuation suffered by the incident ultrasonic beam, one should know the attenuation coefficient due to scattering. Grain properties, elastic characterization, i.e., anisotropy of the materials, the degree of inhomogeneity of the structure and frequency region of the ultrasonic wave are responsible for the attenuation due to scattering. When, developing a model for the ultrasonic wave propagation, all these mechanisms must be taken into account. Detailed information on models for each of the mechanisms will be explained in further chapters.

In this study, as a starting point, losses other than scattering were not taken into consideration. Losses caused by absorption, beam divergence as well as various structures within the grains and at the grain boundaries such as dislocations, twins, and voids are not considered for a simple primary approximation.

It should be noted that, understanding the propagation phenomena includes complex mathematical approaches. Many of these approaches were constructed by various scientists using ultrasonic material characterization. In the following section these will be summarized briefly in chronological order.

1.2 Theoretical Background

In this study, the previous works and theories are investigated and the appropriate ones are used to construct the computer model for the ultrasonic propagation phenomena

Ultrasonic materials characterization of solid media with random microstructures relies, in the literature, mostly upon the use of the coherent field, through measurements of either wave speed or attenuation. This work began in the late 1940s with Mason and McSkimin, [7]. They examined the coherent propagation of ultrasound in polycrystals and demonstrated that the scattering attenuation scales with the fourth power of frequency in the low-frequency

Rayleigh limit. Many others (Pekeris, [8]; Huntington, [9]) conducted similar research on ultrasonic scattering during this time period.

A number of theories have been proposed to describe the coherent propagation through polycrystalline media.

Mason and McSkimin theorized that in the low-frequency Rayleigh limit the grains would scatter energy in the same way that sound scatters from spheres, as discussed by Rayleigh.

Bhatia then improved upon this theory by assuming the grains were isotropic with elastic properties that vary slightly from the elastic properties of the bulk medium, [10]. A perturbation approach was then used. Finally, Bhatia and Moore developed accurate expressions for the scattered energy due to variations in elastic constants for a general orthorhombic crystallite in the Rayleigh limit, [11]. Their expressions agreed with the attenuation measurements of Mason and McSkimin.

The theoretical work for frequencies outside the Rayleigh limit is more recent. Hirsekorn assumed the grains were individual scatterers which scattered like spheres, [2, 3]. This assumption allowed her to calculate attenuations as well as wave speeds as a function of excitation frequency. Her results showed oscillatory behavior with respect to frequency outside the Rayleigh regime because she assumed only single-sized spheres. This behavior is unrealistic for polycrystalline materials.

Stanke and Kino used stochastic operator theory (Karal and Keller, [5]) to derive accurate expressions for attenuations and wave speeds that were valid over the entire frequency range from the Rayleigh limit to the geometrical optics limit, [1]. Their use of the Keller approximation is valid for materials with weak heterogeneity and eliminated the assumption of sphere-like scattering used by Hirsekorn, [2, 3].

Hirsekorn then used an identical approach applied to multiphase polycrystals, [4]. This work showed the dependence of the complex wave number on the volume fraction of the phases comprising the polycrystal.

Experimental work accompanied much of the above theoretical work. Experiments with coherent ultrasound on polycrystalline specimens are typically performed in a water bath using through transmission (two transducers) or

reflection (one transducer) techniques. If two transducers are used, one transducer sends a known amplitude into the specimen which is received by the other transducer located on the opposite side of the specimen. The change in amplitude of the received signal after passing through the medium is attributed to the exponential decay caused by attenuation. If one transducer is used, it acts as both transmitter and receiver. The wave reflects from the opposite face and returns to the transducer. In this case, the amplitude will be reduced due to two specimen crossings. Coherent ultrasound experiments have many limitations. Absorption and scattering attenuations cannot be distinguished in these types of experiments. The methods are also dependent upon certain geometric constraints. The surfaces must be perfectly parallel and polished so that the transmission and reflection from the faces can be accurately modeled. Diffraction corrections must also be made. The merit of coherent ultrasonic experiments for field component characterization is thus limited.

1.3 Physical Definition of the Problem

The main goal of this thesis is to create a basis for the further studies on simulating ultrasonic propagation in polycrystalline solid structures. Modern structures can be multi-phased, strongly textured, and fully anisotropic and can have discrete scatterers. In this study, several assumptions are made in order to simplify the problem for a primal approximation. These approximations are believed to build up a basis for the further studies. In fact it provides good characterization for the recent problem. Consequently, the propagation problem is divided in to several stages, each of whose physical appearance will be described distinctly in the following sections.

It is assumed that there is no energy deviation from the parallel beam by means of beam divergence, i.e., plane shear or longitudinal waves. Also, attenuation due to absorption is not included in this picture. In order to calculate the attenuation coefficient for a wave, the unified theory of Stanke and Kino for single phase polycrystalline structures will be used, [1]. Their results indicated that some numerical calculations and algebraic manipulations are needed to obtain attenuation coefficients and phase velocities. Grain properties, frequency regions,

elastic constants, and relative anisotropies are the variables affecting the attenuation coefficient here. The assumptions made for mathematical formulations will be explained in the following chapters. However, it must be clarified that the structure is assumed to be untextured, the grains are equiaxed, have cubic symmetry, and have a low anisotropy factor [1].

As a further study, wave source of the ultrasound can be modified for the same theory to get a divergent beam rather than a plane wave. It is obvious that additional theories should be investigated or developed to get accurate results. From this point, transducer interactions, refraction diffraction and reflection loss principles of waves and absorption mechanisms can be investigated for additional improvements.

CHAPTER 2

GENERAL PRINCIPLES OF PROPAGATION

2.1 Elastic Constants and Notations

In order to understand wave propagation one has to gain some knowledge about elastic constants and their tensor notations. Tensor notation can provide a very concise way of writing general identities in physics such as in Hooke's Law,

$$\sigma_{ij} = C_{ijkl}\varepsilon_{ijkl} \tag{2.1}$$

One can easily notice that σ (stress component), C (elastic constant component), and ε (strain component) have indices like i, j, k, and l.

C_{ijkl} is fourth rank tensor notation of elastic constant. Tensors may have an arbitrary number of indices. The number of these indices defines the rank of the tensor. Zeroth-rank tensors are called scalars and first rank tensors are called vectors which are all familiar terms in physics, [12].

Tensor equations actually represent a number of equations whose number is defined by 3^k, where k is the number of the free indices. In the Hooke's Law above, on the left hand side of the equation, there exist two free indices, i, and j, so the number of equations is nine.

If in some tensor expression a certain index occurs twice (dummy index), this expression like in the right hand side of the Hooke's Law is summed with respect to that index for all admissible values. This is called Einstein summation convention.

In the forthcoming discussions of tensor equations, Einstein summation convention will be used.

As mentioned before, theoretical approach of this modeling considers the grain structures have cubic class of symmetry. For cubic class of symmetry with anisotropy only three independent elastic constants remain, C_{11},C_{12}, and C_{44}. These constants are the contracted notations of stiffness which are in fact fourth rank tensor components, [12].

When there is a full isotropy with cubic class of symmetry only two constants remains, C_{11}, and C_{12} which are also defined by Lame Constants.

In polycrystalline materials, single crystals make up the polycrystals and they can be at all possible directions. The stiffness of the aggregate is the space averages of the stiffness of the crystallites which is referred as Voight averages, [12].

In following chapters, the elastic constants will be referred as unweighted Voight averages, C_{IJ}^{o},of the constants of the constituent crystallites and the relation between the Voight averages and the Lame's constants will be defined.

2.2 Wave Equation

Ultrasonic propagation constants of materials are determined by the use of waves having low amplitudes, for which there is a linear relationship between the applied stress and the resultant strain. For ease of correlation of the propagation constants of a given substance with its other physical constants, measurements are made as far as possible with plane waves. These originate from a which vibrates with simple harmonic motion.

Where the source vibrates in the direction of the wave motion, *longitudinal waves* are propagated. These waves give rise to alternate compressions and often called as *compression waves*. Where the motion of the source is at right angles to the direction of wave motion, *transverse waves* are propagated. For a medium in bulk these give rise to alternating shear stresses and the term *shear wave* is often used.

The general form of the wave equation for a simple harmonic motion in any propagation direction can be represented as

$$\frac{\partial^2 u}{\partial t^2} = \vartheta^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \tag{2.2}$$

where, u is the particle displacement and υ is the plane wave velocity, [14]. For a wave whose average propagation is parallel to z-axis this equation becomes

$$\frac{\partial^2 u}{\partial t^2} = \vartheta^2 \left(\frac{\partial^2 u}{\partial z^2} \right) \tag{2.3}$$

The propagation properties of the unperturbed wave which travels in the unperturbed medium are

$$\vartheta_o = \left[\frac{C_{IJ}^o}{\rho} \right]^{1/2} \tag{2.4}$$

$$k_o = \frac{\omega}{\vartheta_o} \tag{2.5}$$

$$\lambda_o = \frac{2\pi\vartheta_o}{\omega} \tag{2.6}$$

where IJ is 11 for longitudinal waves and 44 for the shear waves. In equation (2.4), ϑ_o is velocity, ρ is material density respectively. In Equation (2.5), k_o is propagation constant (sometimes called as wave number), and ω is the radian frequency defined as $2\pi f$. In equation (2.6) λ_o denotes the wavelength.

The main goal in this study is to find a general solution having the form of a plane wave propagating in the z direction

$$u(\mathbf{r}) = Ue^{i(\omega t - kz)} \tag{2.7}$$

where **r** is any point in a laboratory coordinate system with it z axis parallel to the average direction of propagation, [1]. It has an expected propagation constant of the form

$$k = \beta - i\alpha \tag{2.8}$$

α is the attenuation coefficient, and phase velocity is given by the equation

$$\vartheta_p = \frac{\omega}{\beta} \tag{2.9}$$

where β is known as wave number.

CHAPTER 3

MODELS OF UNIFIED THEORY

Unified theory that is going to be used as the basis for the computer modeling is composed of four discrete models, [1]. These are models for material characterization, grain geometry, inhomogeneity, and frequency regions. Mathematical method (Keller approximation) to solve this physical problem is explained in detail in the last section of this chapter.

3.1 Material Characterization Model

Modeling of propagation of elastic waves in polycrystalline media, is a complex case of general class of problems dealing with wave propagation in random, inhomogeneous media. The anisotropy of the grains complicates the mathematical formalism immensely, but the basic scattering concepts which must be used to obtain solutions for the expected value of propagation constant are the same as for the simpler cases. As a mathematical characterization of material properties, an *anisotropic* crystal characterization model is used.

In a solid there are generally three wave modes possible in any direction, with properties depend on that direction. The anisotropic characterization accounts for these effects with a great increase in mathematical complexity. It is an approximation since it ignores various structures within the grains and at the grain boundaries which can affect the propagation of waves, e.g., dislocations, twins and voids.

3.2 Grain Geometry Models

Three models for the geometry of the grains which have been used to solve the scattering problem are (1) a collection of *individual particles*, (2) a regular *array of particles*, (3) a *stochastic process*. First two models tend to employ shapes for the particles which are simple and uniform, which can lead to coherent artifacts that would not arise with irregularly shaped grains.

The *stochastic process* model is the most integrated since it treats the medium as a whole in terms of geometric statistics of grains, which directly reflect the macroscopic properties of the sample which they compose. This model is chosen for the recent problem.

It is believed to be convenient to introduce a few concepts at this point. Let a "free path" to be line segment passing through one grain with its end points on grain boundaries and having a length d. To specify the granular geometry, the density function $p_d(y)$ is used, defined such that $p_d(y)dy$ is the probability that a randomly placed point will fall on a free path of length y. In the general formulation, we allow this function to vary depending on the direction of free paths with respect to laboratory coordinate system, but in calculations, it is assumed that it does not, i.e., the grains are equiaxed. The associated "geometric correlation" function

$$W(r) = \int_r^\infty \left(1 - \frac{r}{y}\right) p_d(y) dy \tag{3.1}$$

is the probability that points separated by a distance r are in the same grain, [1]. The effective linear dimension of the grains is given by the mean free path length, [1]

$$\bar{d} = \int_0^\infty y p_d(y) dy \tag{3.2}$$

and the effective average volume of the grains is given by

$$V_{g,eff} = \int_{v} W(r)dv \tag{3.3}$$

where V is an infinite volume, dv is a differential volume, and r is the radial component in a spherical coordinate system, [1].

It is assumed that p_d(y) has the simple but realistic form

$$p_d(y) = ye^{-y/l}/l^2 \tag{3.4}$$

where l will be referred to as the correlation distance. For this choice it is obtained that

$$W(r) = e^{-r/l} \tag{3.5}$$

$$\bar{d} = 2l \tag{3.6}$$

$$V_{g,eff} = 8\pi l^3 \tag{3.7}$$

3.3 Degree of Inhomogeneity

The inhomogeneity in a single phase, polycrystalline material is caused by and is proportional to the anisotropy of the grains. If the grains are weakly anisotropic, then the elastic properties of the medium have only weak variations from grain to grain, and the amount of interaction between the fields in the grains is small. In this case the relative inhomogeneity of the propagation constant, $\in$, is much less than unity and can be expressed in terms of effective elastic constants as

$$\in^2 = \frac{1}{4}\left\{\left\langle\left[C_{IJ}^e(r) - C_{IJ}^o\right]^2\right\rangle / \left(C_{IJ}^o\right)^2\right\} \tag{3.8}$$

The fundamental assumption of the theoretical formulation is that ϵ is small. Further calculations assume in addition that the grains have cubic symmetry and are untextured, i.e., all Euler rotations are equally likely. For this case the inhomogeneities are

$$\epsilon_l^2 = (4/525)\left[\nu^2/(C_{11}^o)^2\right] \tag{3.9}$$

where, ϵ_l is the value for l waves, and

$$\epsilon_s^2 = (3/700)\left[\nu^2/(C_{44}^o)^2\right] \tag{3.10}$$

where, ϵ_s is the value for s waves and ν is the invariant anisotropy factor for cubic class crystals defined as

$$\nu = C_{11} - C_{12} - 2C_{44} \tag{3.11}$$

3.4 Frequency Regions

The ratio between the size of grains and the acoustic wavelength strongly influences the amount and type of scattering that occurs. The product

$$x_o = k_o \bar{d} = 2\pi d/\lambda_o \tag{3.12}$$

is a measure of this ratio which is proportional to the frequency for a given sample and will be referred to as the "normalized frequency" There are three frequency regions where the attenuation and phase velocity are approximately equal to limiting cases that correspond to simple physical models of scattering mechanism: (1) the *Rayleigh* region where $x_o \ll 1$, (2) the *stochastic* region where $1 < x_o < 1/\epsilon$, (3) the *geometric* region where $1/\epsilon < x_o$. The calculations are valid for all values of normalized frequency. The assumptions used in most of the literature limit the frequency range over which the calculations are valid. The starting point for many

of these theories is the Born approximation, which calculates the mean wave in terms of scattered fields generated by the unperturbed wave and not valid for all frequency ranges.

3.5 Keller Approximation

Keller, [17] developed a general formulation for determining propagation constants in slightly inhomogeneous media using the *stochastic process* model and a general operator notation which allows his result to be applied to various physical problems, for a particular inhomogeneous medium, labeled ξ, of an ensemble of possible media, Ξ. The stochastic wave equation be written in the form

$$L_{\xi}(\mathbf{r})u_{\xi}(\mathbf{r}) = 0 \tag{3.13}$$

where $u_{\xi}(\mathbf{r})$ is the actual field in a medium ξ and $L_{\xi}(\mathbf{r})$ is a linear operator which describes that medium. The set of functions $L_{\xi}(\mathbf{r})$ and the probability density function p_{ξ}, which is the probability of choosing any particular medium, form a stochastic process.

If there is an operator L_o which is independent of r and ξ, invertible

$$L_{\xi}(\mathbf{r}) = L_o - \in L_{1,\xi}(\mathbf{r}) - \in^2 L_{2,\xi}(\mathbf{r}) - O(\in^3) \tag{3.14}$$

where $\in$ is much less than unity, then a general perturbation solution for $u(\mathbf{r})$ can be obtained or equivalently for propagation constant, k.

The hypothetical unperturbed medium is described by L_o, and the unperturbed solution u_o is a plane wave.

$$u_o(\mathbf{r}) = Ue^{i(wt-k_o z)} \tag{3.15}$$

which satisfies the wave equation

$$L_o\, u_o(\mathbf{r}) = 0 \tag{3.16}$$

The inverse of L_o is

$$L_o^{-1} u_o(\mathbf{r}) = \int_V G(\mathbf{r}-\mathbf{r}')u(\mathbf{r}')dv' \tag{3.17}$$

where $G(\mathbf{r}-\mathbf{r}')$ is the Green's Function, [1]

By use of Eq. (3.13) and (3.14) and inverse operator, u_ξ can be expressed as

$$u_\xi = u_o + \in (L_o)^{-1} L_{1,\xi} u_\xi + \in^2 (L_o)^{-1} L_{2,\xi} u_\xi + O(\in^3) \tag{3.18}$$

where $\mathbf{r}$ dependence is not shown. All of the terms in the right hand side, except the first term, represent perturbations to the field caused by scattered waves generated by the actual field. Repeated iteration, followed by averaging has done by Stanke and Kino, [1], which yields the Born series for the mean plane wave

$$u = u_o + \in (L_o)^{-1}\langle L_1\rangle u_o + \in^2 (L_o)^{-1}\langle L_1 (L_o)^{-1} L_1\rangle u_o + \in^2 (L_o)^{-1}\langle L_2\rangle u_o + O(\in^3) \tag{3.19}$$

Solving Eq. (3.19) for u_o and applying L_o to both sides yields and truncating the results to accuracy $\in^2$

$$L_o u = \in \langle L_1\rangle u + \in^2 \left[\langle L_1 (L_o)^{-1} L_1\rangle - \langle L_1\rangle (L_o)^{-1}\langle L_1\rangle + L_2\right] u \tag{3.20}$$

This equation accounts for some degree of multiple scattering since the scattered waves are generated by the mean plane wave, which is itself affected by scattering. The expected field is sufficiently better as an approximation to the actual field that this form is valid for all frequencies if the fundamental assumption that $\in \ll 1$ is fulfilled.

3.5.1 Application of Anisotropic Characterization

In order to apply the anisotropic characterization to the Keller approximation, Einstein convention for summations and the comma notation will be used.

The stochastic wave equation is

$$\left[C^{\xi}{}_{ijkl}(\mathbf{r})u^{\xi}{}_{k,l}(\mathbf{r})\right]_{,j}+\rho w^2 u^{\xi}{}_{i}(\mathrm{r})=0 \tag{3.21}$$

where $C^{\xi}{}_{ijkl}(\mathbf{r})$ is the local elastic tensor and $u^{\xi}(\mathbf{r})$ is the displacement vector. A perturbation of the local elastic tensor from the unperturbed is defined as

$$\in \Delta^{\xi}{}_{ijkl}(\mathbf{r}) = C^{\xi}{}_{ijkl}(\mathbf{r}) - C^{o}{}_{ijkl} \tag{3.22}$$

The operators needed to apply are

$$L_o u \Rightarrow \left[C_{ijkl}{}^{0} u_{k,lj}(\mathbf{r}) + \rho\omega^2 u_i(\mathbf{r})\right] \tag{3.23}$$

$$L_{1,\xi}(\mathrm{r})u(\mathrm{r}) \Rightarrow \left[\in \Delta^{\xi}{}_{ijkl}(\mathbf{r})u_i(\mathbf{r})\right]_{,j} \tag{3.24}$$

$$L_{2,\xi}(\mathbf{r})u(\mathbf{r}) \Rightarrow 0 \tag{3.25}$$

The Green's function is given as

$$G_{ik}(r) = (1/r)\left[n_r n_k g(r) + \delta_{ik} h(r)\right] \tag{3.26}$$

where δ is Dirac delta function

$$g(r) = (4\pi\rho\omega^2 r^2)^{-1}\left[3(1+ikr) - k^2 r^2\right]e^{-ikr}\Big|_l^s \tag{3.27}$$

$$h(r) = -(4\pi\rho\omega^2 r^2)^{-1}\left[(i+ikr) - k^2 r^2\right]e^{-ikr}\Big|_l^s - (k_s)^2 r^2 e - ik_s\gamma \tag{3.28}$$

By assuming the media are statistically homogeneous and with the use of conditional probabilities Stanke and Kino, [1] has found second order Keller approximation to be

$$0 = \left[\begin{array}{l}\left[\left(C^o_{ijkl} + \in\left\langle\Delta^\xi_{ijkl}\right\rangle\right)\left\langle u_k(\mathbf{r})\right\rangle_{,l}\right]_{,j} + \rho\omega^2\left\langle u_i(\mathbf{r})\right\rangle - \in^2\left[\left\langle\Delta^\xi_{ijkl}\Delta_{mnpq}\right\rangle - \left\langle\Delta^\xi_{ijkl}\right\rangle\left\langle\Delta_{mnpq}\right\rangle\right] \\ \times\left(\int G_{km,l}(\mathbf{r}-\mathbf{r}')\left[W(\mathbf{r}-\mathbf{r}')\left\langle u_p(\mathbf{r}')\right\rangle_{,q'}\right]_{,n'} dv\right)\end{array}\right] \tag{3.29}$$

The displacement field is the product of amplitude A, a polarization unit vector $\hat{u}_i$, and propagation factor containing the unit vector, which specifies the propagation direction $\hat{z}$.

$$u_i(\mathbf{r}) = A\hat{u}_i e^{ik\mathbf{r}\cdot\hat{z}} \tag{3.30}$$

Both of the derivatives, $\partial/\partial x_j$ and $\partial/\partial x_l$, in the first term on the right-hand side of Eq. (3.29) can be evaluated to yield a factor of $(-ik\hat{z})^2$. Since this term and the following vary only as $e^{ik\mathbf{r}\cdot\hat{z}}$, the same must be true of the last term, so that its derivatives, $\partial/\partial x_j$ and $\partial/\partial x_l$, can also be evaluated to yield

$$0 = \left[\begin{array}{l}\left(C^o_{ijkl} + \in\left\langle\Delta_{ijkl}\right\rangle\right)\hat{u}_k\hat{z}_j\hat{z}_l - \rho(\omega^2/k^2)\hat{u}_i - \in^2\left(\left\langle\Delta_{ijkl}\Delta_{mnpq}\right\rangle - \left\langle\Delta_{ijkl}\right\rangle\left\langle\Delta_{mnpq}\right\rangle\right) \\ \times\int G_{km,l}(\mathbf{r}-\mathbf{r}')\left[W(\mathbf{r}-\mathbf{r}')e^{ik(\mathbf{r}-\mathbf{r}')\cdot\hat{z}}\right]_{,n'}\times dv'\hat{u}_p\hat{z}_j\hat{z}_q\end{array}\right] \tag{3.31}$$

Green's theorem for the elastic case has the form

$$\int_V G_{ij,k'}(\mathbf{r}-\mathbf{r}')u_j(\mathbf{r}')dv = \int_S G_{ij}(\mathbf{r}-\mathbf{r}')u_j(\mathbf{r}')\hat{n}_{k'}ds - \int_V G_{ij}(\mathbf{r}-\mathbf{r}')u_{j,k}(\mathbf{r}')dv \tag{3.32}$$

where S is the surface of V, ds is the differential surface, and $\hat{n}_i$ is the outward unit normal vector to S. Since the autocorrelation function $W(\mathbf{r}-\mathbf{r}')$ has a finite extent and V represents an infinite volume, the surface integral can be neglected. Since Eq. (3.32) dependent only on $(\mathbf{r}-\mathbf{r}')$ it will be convenient to change this variable to $\mathbf{r}$. After the application of the Green's theorem and changing the variable the final form is

$$0 = \left[\begin{array}{l} \left(C^o_{ijkl} + \in \langle \Delta_{ijkl} \rangle\right)\hat{u}_k \hat{z}_j \hat{z}_l - \rho(\omega^2/k^2)\hat{u}_i - \in^2 \left(\langle \Delta_{ijkl}\Delta_{mnpq} \rangle - \langle \Delta_{ijkl} \rangle \langle \Delta_{mnpq} \rangle \right) \\ \times \int_V G_{km}(\mathbf{r}) \left[W(\mathbf{r}) e^{ik(\mathbf{r})\cdot\hat{z}} \right]_{\ln'} \times dv \; \hat{u}_p \hat{z}_j \hat{z}_q \end{array} \right] \tag{3.33}$$

This is an equation for the expected propagation constant k, accurate to $\in^2$, and applicable for arbitrary symmetry in media with texture and preferred grain elongation.

3.5.2 Restricted Cases for Calculations

In order to get a general solution for the anisotropic characterization several numbers of modifications and assumptions were made on Eq. (3.33): (1) the media are untextured, (2) the grains are equiaxed, and (3) the grains have cubic symmetry.

The first assumption requires that $\langle \Delta_{ijkl} \rangle = 0$, since it represents the macroscopic average anisotropy in textured media, and brings out a form

$$0 = \left[\begin{array}{l} C^o_{ijkl}\hat{u}_k \hat{z}_j \hat{z}_l - \rho(\omega^2/k^2)\hat{u}_i - \in^2 \langle \Delta_{ijkl}\Delta_{mnpq} \rangle \\ \times \int_V G_{km}(\mathbf{r}) \left[W(\mathbf{r}) e^{ik(r)\cdot\hat{z}} \right]_{\ln'} \times dv \; \hat{u}_p \hat{z}_j \hat{z}_q \end{array} \right] \tag{3.34}$$

The second assumption allows W $(\mathbf{r})$ to be replaced by W(r), which greatly simplifies derivatives $\partial/\partial x_l$ and $\partial/\partial x_n$, which appear in the Green's integral.

The third assumption leads the Voight averages of elastic constants to be

$$C_{11}^{o} = \lambda + 2\mu - \frac{3}{5}\upsilon$$
$$C_{12}^{o} = \lambda + \frac{1}{5}\upsilon \qquad (3.35)$$
$$C_{44}^{o} = \mu + \frac{1}{5}\upsilon$$

where λ, μ are the Lame constants for the isotropic case, and υ is the anisotropy factor.

With the propagation direction $\hat{z}_i$ pointing in the three directions, the form of the two-point averages for longitudinal waves is

$$\epsilon \left\langle \Delta_{33kl} \Delta_{mn33} \right\rangle = \epsilon^2 \left\langle \Delta^2 \right\rangle_{klmn}^{l} \qquad (3.36)$$

and for shear waves, with the particle motion in the *l* direction

$$\epsilon \left\langle \Delta_{13kl} \Delta_{mn13} \right\rangle = \epsilon^2 \left\langle \Delta^2 \right\rangle_{klmn}^{s} \qquad (3.37)$$

Stanke and Kino have evaluated the averages and used the symmetry relations in the elastic tensor to find other averages, which appear in Eq. (3.34), [1]. They also carried out the calculations analytically as far as possible using the algebraic manipulation MACSYMA, [15]. They assumed the geometric autocorrelation function W(r) has the form as given in Eq. (3.5)

These modifications include mathematically complex steps, which will not be represented here. However, after evaluation of the sums, derivatives, and integrals, and some cosmetic changes, the solution of the wave equation for *longitudinal waves* becomes, [1]

$$(x_l)^2-(x_{ol})^2=-\frac{1}{525}\frac{\nu^2}{(C_{11}^o)^2}\frac{(x_l)^2}{(x_{ol})^2}\left[\begin{array}{l}\arctan\left(\frac{x_l}{2+ix_{ol}}\right)\left(\begin{array}{l}x_l+\frac{1}{x_l}\left[12+15(x_{ol})^2\right]+\frac{1}{(x_l)^3}\left[48+72(x_{ol})^2+15(x_{ol})^4\right]\\+\frac{1}{(x_l)^5}\left[64+48(x_{ol})^2+12(x_{ol})^4+(x_{ol})^6\right]\end{array}\right)\\-\arctan\left(\frac{x_l}{2+ix_{os}}\right)\left(\begin{array}{l}x_l+\frac{1}{x_l}\left[12-9(x_{os})^2\right]+\frac{1}{(x_l)^3}\left[48-24(x_{os})^2-9(x_{os})^4\right]\\+\frac{1}{(x_l)^5}\left[64+48(x_{os})^2+12(x_{os})^4+(x_{os})^6\right]\end{array}\right)\\-\frac{16(x_{ol})^2\left[4+4ix_{ol}+(x_l)^2\right]}{4+4ix_{ol}+(x_l)^2-(x_{ol})^2}-\frac{20(x_{os})^2\left[4+4ix_{os}+(x_l)^2\right]}{4+4ix_{os}+(x_l)^2-(x_{os})^2}+i(x_{ol}-x_{os})\\+\frac{2}{(x_l)^2}\left(4i(x_{ol}-x_{os})-8\left[2(x_{ol})^2+(x_{os})^2\right]+\frac{i}{3}\left[23(x_{ol})^3+13(x_{os})^3\right]\right)+\\\frac{1}{(x_l)^4}\left\{\begin{array}{l}16i(x_{ol}-x_{os})-16\left[(x_{ol})^2-(x_{os})^2\right]+8i\left[(x_{ol})^3-(x_{os})^3\right]-2\left[(x_{ol})^4-(x_{os})^4\right]\\+i\left[(x_{ol})^5-(x_{os})^5\right]\end{array}\right\}\end{array}\right] \tag{3.38}$$

and for shear waves

$$(x_s)^2-(x_{os})^2=-\frac{1}{1050}\frac{\nu^2}{(C_{44}^o)^2}\frac{(x_s)^2}{(x_{os})^2}\left[\begin{array}{l}\arctan\left(\frac{x_s}{2+ix_{ol}}\right)\left(\begin{array}{l}-x_s-\frac{1}{x_s}\left[12-9(x_{ol})^2\right]-\frac{1}{(x_s)^3}\left[48-24(x_{ol})^2+9(x_{ol})^4\right]\\-\frac{1}{(x_s)^5}\left[64+48(x_{ol})^2+12(x_{ol})^4+(x_{ol})^6\right]\end{array}\right)\\+\arctan\left(\frac{x_s}{1+ix_{os}}\right)\left(\begin{array}{l}x_s+\frac{1}{x_s}\left[12+9(x_{os})^2\right]+\frac{1}{(x_s)^3}\left[48+48(x_{os})^2+9(x_{os})^4\right]\\+\frac{1}{(x_s)^5}\left[64+48(x_{os})^2+12(x_{os})^4+(x_{os})^6\right]\end{array}\right)\\-\frac{20(x_{ol})^2\left[4+4ix_{ol}+(x_s)^2\right]}{4+4ix_{ol}+(x_s)^2-(x_{ol})^2}-\frac{28(x_{os})^2\left[4+4ix_{os}+(x_s)^2\right]}{4+4ix_{os}+(x_s)^2-(x_{os})^2}-i(x_{ol}-x_{os})\\-\frac{1}{(x_s)^2}\left(8i(x_{ol}-x_{os})+4\left[4(x_{ol})^2+5(x_{os})^2\right]-\frac{i}{3}\left[13(x_{ol})^3+14(x_{os})^3\right]\right)-\\\frac{1}{(x_s)^4}\left\{\begin{array}{l}16i(x_{ol}-x_{os})-16\frac{\left[(x_{ol})^2-(x_{os})^2\right]}{2}+8i\left[(x_{ol})^3-(x_{os})^3\right]-2\left[(x_{ol})^4-(x_{os})^4\right]\\+i\left[(x_{ol})^5-(x_{os})^5\right]\end{array}\right\}\end{array}\right] \tag{3.39}$$

CHAPTER 4

MODELING PARAMETERS and INPUT DATA

4.1 General

As stated in Eq. (2.8) propagation constant is a complex variable. The wave numbers and the scattering coefficients of the waves in the polycrystal are the real and imaginary parts, respectively, of the complex propagation constants.

The general solution of the wave equation has been stated in the Eq. (2.7). In a more general form

$$u(r) = Ue^{i(\omega t-(\beta-i\alpha)z)} \tag{4.1}$$

and this can also be represented as

$$u(r) = Ue^{-z\alpha}e^{(\omega t-z\beta)} \tag{4.2}$$

where z is the distance in propagation direction, α is the scattering coefficient, ω is the radian frequency, t is time, and β is the wave number.

For the wave amplitude this expression gets the form

$$u(r) = Ue^{-(z+\Delta z)\alpha}\cos\left[\omega(t+\Delta t)-(z+\Delta z)\frac{\omega}{\vartheta_p}\right] \tag{4.3}$$

where U is the initial amplitude.

In order to model this plane wave with distance and time

- α, scattering coefficient,
- ω ,radian frequency (or equivalently frequency f),
- ϑ_p phase velocity (or equivalently wave number β),
- Time step size,
- Distance step size, are the main parameters.

4.2 Scattering Coefficient and Phase Velocity

In order to obtain both scattering coefficient and phase velocity one is to solve Eq. (3.38) or Eq.(3.39) for longitudinal or shear waves, respectively. The solution of both equations is done with software prepared by writing codes including iterative algorithms for two equations. The codes of the software are shown in the Appendix Part A.

Solution of the Eq. (3.38) will give a result for x_l (longitudinal waves) , and solution of Eq. (3.39) will give a result for x_s (shear waves), which are

$$x_l = \beta_l \bar{d} - \alpha_l \bar{d} i \tag{4.4}$$

$$x_s = \beta_s \bar{d} - \alpha_s \bar{d} i \tag{4.5}$$

where $\bar{d}$ is the average grain size. To obtain phase velocity and scattering coefficient these results must be divided into $\bar{d}$. Consequently an input data for the software program will be average grain size, $\bar{d}$. When software is executed, any grain size can be input in millimeters. The model of the software has no limitations for the grain size. Various numerical results of the case studies for various sizes are shown in the Chapter 5.

In order to solve Eq. (3.38) or Eq. (3.39) additional input data is needed for software. These are wave frequency f, and material constants which are

C_{11}^o ,and C_{44}^o, elastic constants ,

ρ_o, density, and

υ, anisotropy factor.

4.2.1 Method for Iterative Solution

Since Eq. (3.38) and Eq. (3.39) cannot be solved analytically, Newton-Raphson method is used to solve these equations. Eq. (3.38) and (3.39) are equations with complex variables, x_l and x_s respectively. It is known that Newton-Raphson method is valid for non-linear equations with complex variables, [13].

Non-linear equation can be defined as

$$f(x) = 0 \tag{4.6}$$

whose roots cannot be determined analytically.

Newton-Raphson method can be stated mathematically as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.7}$$

where n is a integer starting from 1. One has to choose a starting root, x_1 to iterate a new root, x_2. This iteration is repeated until a stopping criterion is reached.

In this point, it is seen that there should be starting root and a stopping criterion. Choosing the starting root is important because if a wrong root is chosen the method will not converge or converge to another root which is not the desired root. Since there is no convergence problem for these equations, roots are taken for longitudinal waves as

$$x_1 = x_{ol} - i \tag{4.8}$$

and for shear waves as

$$x_1 = x_{os} - i \tag{4.9}$$

where x_o is stated as in Eq. (3.12)

Using any iterative scheme on a computer to estimate roots of an equation requires some condition to be satisfied so that the algorithm stops. One way is to

stop after a certain number of iterations. Stopping after a certain number of iterations prevents algorithm from aimlessly looping. After some experiments made on the model this number is chosen as 100. This number is sufficient for the absolute value of two successive rounded iterates agree up to 10 decimal places.

4.2.2 Material Constants

As stated before, there are four material constants to be used in the modeling of ultrasonic wave propagation

- C_{11}^{o},and C_{44}^{o}, elastic constants
- ρ_o, density
- υ, anisotropy factor

Computer program indicates the units of these moduli and constants which are to be input for correct results.

4.3 Ultrasonic Wave Frequency

Ultrasonic wave frequency can be any frequency since the main formalism detailed in the previous chapters is valid for all values of frequency. The main thing influencing the amount of scattering is the ratio of the size of grain to the wave length, which is named as normalized frequency. Eq. (3.38) and, Eq. (3.39) is valid for all values of normalized frequency, which ranges from 10^{-2} to 10^{3}. In Chapter 5, numerical results will be presented with different acoustic frequency values within this range. Input must be done in MHz.

4.4 Time Step Size

It can be seen from Eq. (5.3) that changing the time step size will influence the modeling. Decreasing the time step size will result in more exact animations. However hardware used in the modeling has some memory limitations. Moreover lowering the time step will lead to excessive time consuming. Considering these, the modeling is done for the time step 0.01 seconds.

4.5 Mesh Model

In the computer program, mesh size is left as an input which can be decided by the user. Any size can be used for this model. The number of elements is 100 in x and z directions. Mesh model is given in Fig 4.1.

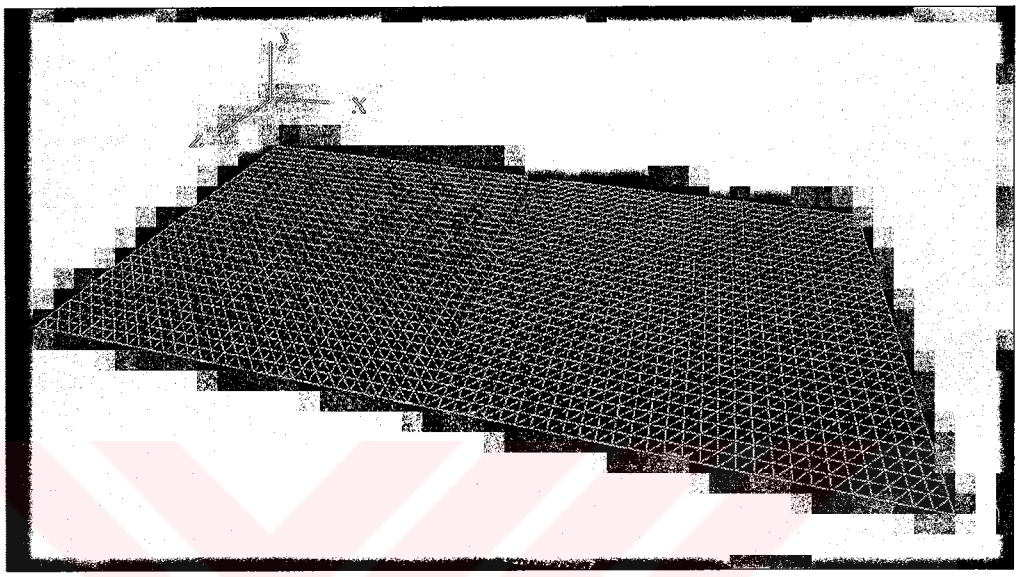

Figure 4.1 Mesh Model

Mesh model can be rotated around each axis by changing the camera positions in the simulation window. Simulation window examples can be found in case studies in Chapter 5.

In y direction, amplitude of the wave is visualized. Initial amplitude can also be any value. By default, initial amplitude value is chosen as 1. The unit of this number is twip, which is used as graphics unit in Visual Basic 6.0© Programming language, [16]. Microsoft Directx 7.0 libraries were used as graphical interfaces.

Because the wave source is a point source, due to symmetry of axes, -z direction is not shown in the simulated animation.

CHAPTER 5

NUMERICAL RESULTS and DISCUSSION

5.1 Verification of the Proposed Model

Among the parameters of the wave propagation model, scattering coefficient and phase velocity are the most important parameters instructing the whole wave propagation model. So, results of numerical algorithms performed by the written codes were compared with the results published in literature.

Codes written in Visual Basic programming language, (Appendix A, B) execute Newton-Raphson method to generate complex propagation constants, Eq. (2.8). Computer program needs material constants, acoustic frequency, and grain size to output a real, and an imaginary part. From the real part of the complex result phase velocity is obtained by use of Eq.(2.9). Scattering coefficient is the imaginary part.

Numerical methods are applied for longitudinal and shear waves of aluminum and iron polycrystals. Analyses were done for the frequency dependence of phase velocity and scattering coefficients as well as grain size dependence of scattering coefficients.

It has been seen that numerical analysis for the shear waves disagrees with the ones in the literature, [1]. The disagreement of numerical results may have resulted from possible publishing mistakes in the formalism, Eq. (3.39), of unified theory for the transverse waves. A serious attempt was done to make contact with writers resulting with no response. For that reason, computer modeling and simulations could have only been done for the longitudinal waves.

On the other hand, results for longitudinal waves are identically same with those of Stanke and Kino, [1]. The graphical results are given below.

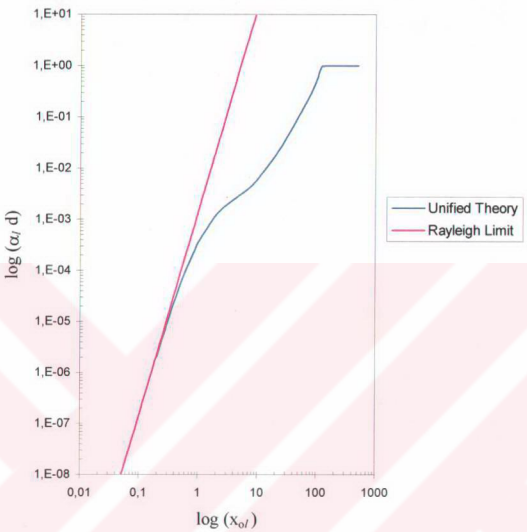

Figure 5.1 Scattering coefficient versus frequency for longitudinal waves in polycrystalline aluminum.

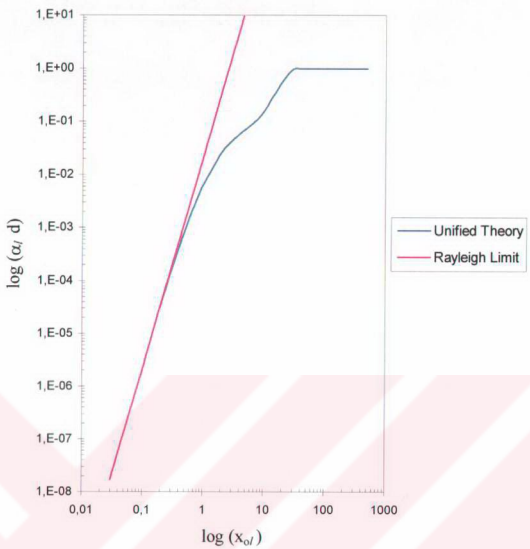

Figure 5.2 Scattering coefficient versus frequency for longitudinal waves in polycrystalline iron.

Normalized scattering coefficient versus normalized frequency plots for Al and Fe are shown in Figure 5.1 and Figure 5.2 with logarithmic scale. These plots are independent of the choice of average grain size, $\bar{d}$, and are agree with those obtained in the studies of Stanke and Kino [1]. Results show that scattering curves coincides with Rayleigh limits at low frequencies, makes humps between Rayleigh and stochastic transitions and independent of frequency at geometric regions (high frequency).

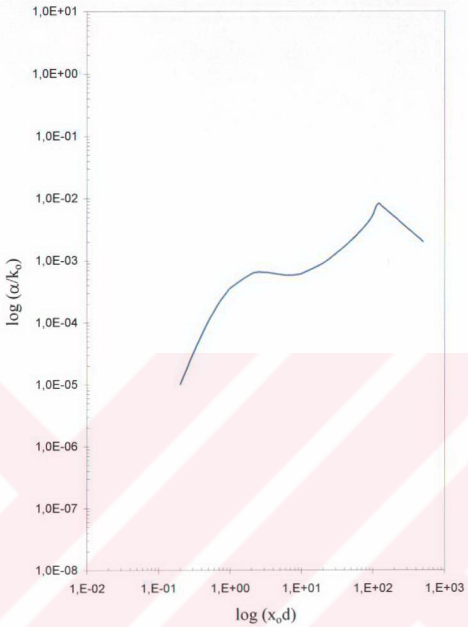

Figure 5.3 Scattering coefficient versus grain size for longitudinal waves in polycrystalline aluminum.

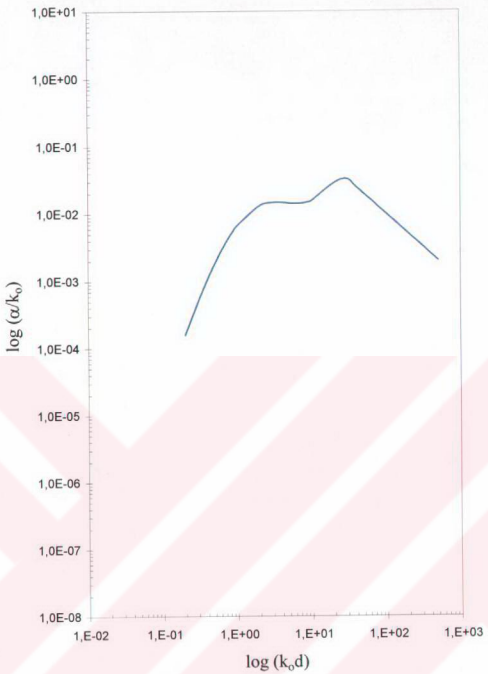

Figure 5.4 Scattering coefficient versus grain size for longitudinal waves in polycrystalline iron.

Normalized scattering coefficient versus normalized grain size plots for Al and Fe are shown in Figure 5.3 and Figure 5.4 with logarithmic scale. These plots are agree with those obtained in the studies of Stanke and Kino [1]. In the Rayleigh region scattering coefficient is proportional to d^3, in stochastic region to d, and in geometric region to $1/d$. For both of plots the maxima occurs at $\bar{d} = 1/\in k_o$.

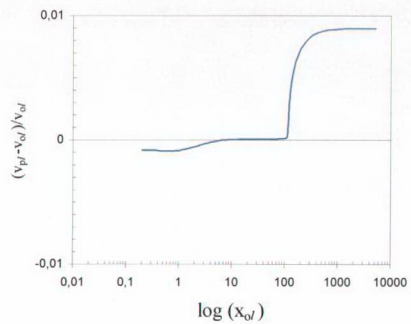

Figure 5.5 Phase Velocity versus frequency for longitudinal waves in polycrystalline aluminum.

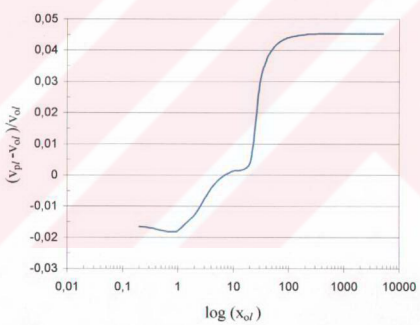

Figure 5.6 Phase Velocity versus frequency for longitudinal waves in polycrystalline iron.

Normalized variation of phase velocity versus normalized frequency plots for aluminum and iron polycrystals are shown in Figure 5.5 and 5.6. Plots are identical with those of Stanke and Kino [1].

5.2 Case Studies

The computer program was executed for aluminum, iron and Fe-30%Ni, which have cubic structures in order to investigate the scattering properties of the longitudinal waves. Among these structures, aluminum can be considered as low anisotropy example while iron and Fe-30%Ni as high-anisotropy example. Moduli, densities and anisotropy factors are tabulated in Table 5.1.

Table 5.1 Material Constants

Matrix	$C_{11}^o \times 10^9$ (N/m^2)	$C_{44}^o \times 10^9$ (N/m^2)	ρ_o (g/cm^3)	$\upsilon \times 10^9$ (N/m^2)
Fe [2]	273.61	82.00	7.86	-136
Al [2]	107.85	26.45	2.70	-11
Fe-30%Ni [6]	177	105	8.22	-155

To compare wave propagation and attenuation the program was executed for various frequencies and grain sizes using different element sizes (100 elements make up the mesh model, see Chapter 4). For visualizing the wave propagation in reasonable sections, the element size was chosen larger than the wavelength. This resulted in the formation of wave envelopes instead of sinusoidal waves.

Wave amplitudes are colored according to their magnitude. As the amplitude attenuates, the color of the wave fades from red to blue, which is very helpful in observing the scattering behavior of the structure. Wave was produced from a point source, so it has the maximum amplitude at this point.

5.2.1 Aluminum

The element size was chosen as 20 mm. Scattering coefficient is found to be 3.170×10^{-1} m^{-1} at a frequency of 5 MHz and an average grain size of 0.08 mm. Figure 5.8 shows the scattering behavior of a longitudinal ultrasonic wave in aluminum. Due to the element size, the wave is propagating in a square cross-section consisting of 100 elements on each side, i.e., with a side length of 2

meters. The input window of the developed program is given in Figure 5.7 for the specific example of aluminum.

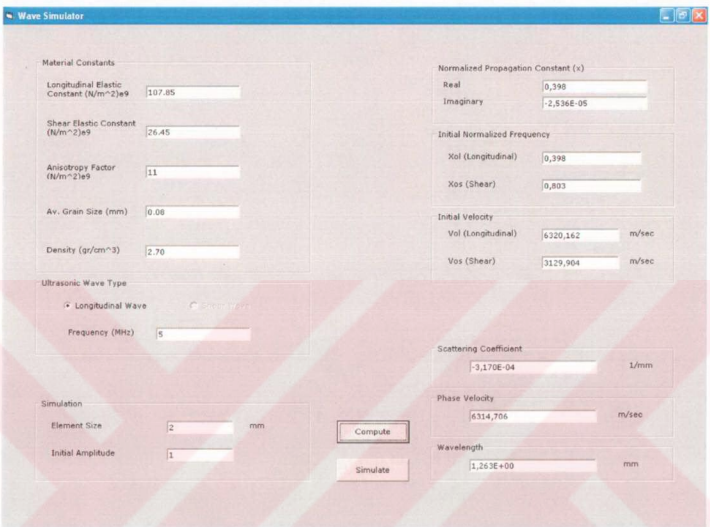

Figure 5.7 Execution of the computer program for aluminum

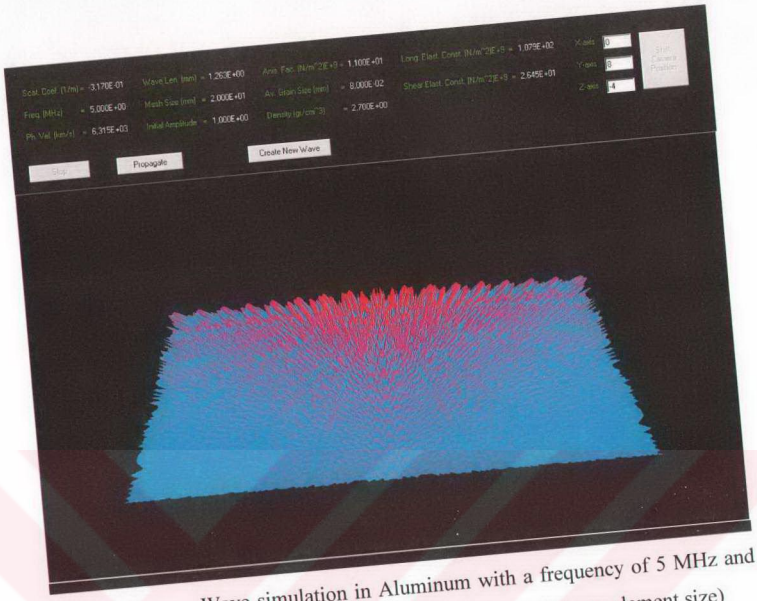

Figure 5.8 Wave simulation in Aluminum with a frequency of 5 MHz and with an average grain size of 0.08 mm. (20 mm element size)

When the average grain size was lowered from 0.08 mm to 0.02 mm, wave scattering behavior dramatically changed. However, in this case, the element size should have to be modified in order to follow the wave attenuation clearly. Figure 5.9 shows the wave simulation for the same wave frequency (5 MHz), but with an element size of 1000 mm. Scattering coefficient was calculated as $6{,}958\times10^{-3}$ m^{-1}, which is much lower (by a factor of approximately 45) than the one with grain size of 0.08 mm. This dramatic change shows how the grain size influences the scattering behavior of the ultrasonic waves in aluminum.

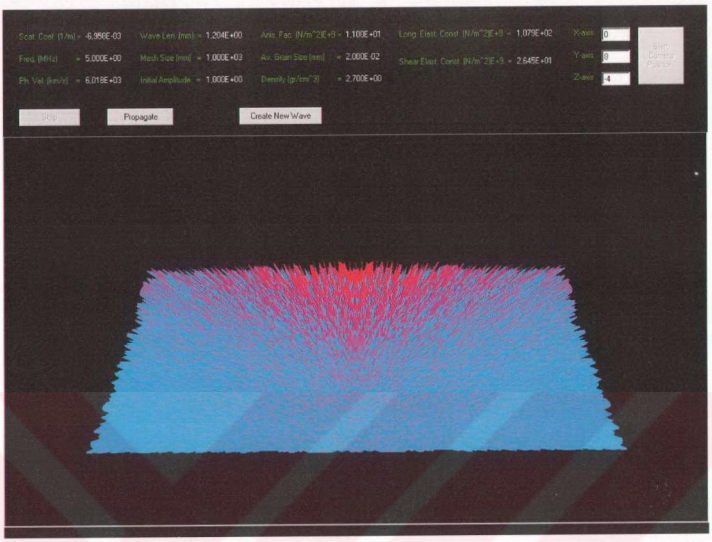

Figure 5.9 Wave simulation in Aluminum with a frequency of 5 MHz and with an average grain size of 0.02 mm. (1000 mm element size)

In Figure 5.10 and 5.11, frequency dependence of scattering is observed. Increasing the frequency from 5 MHz to 10 MHz, increased the scattering coefficient to a value of 2.577 m^{-1}.

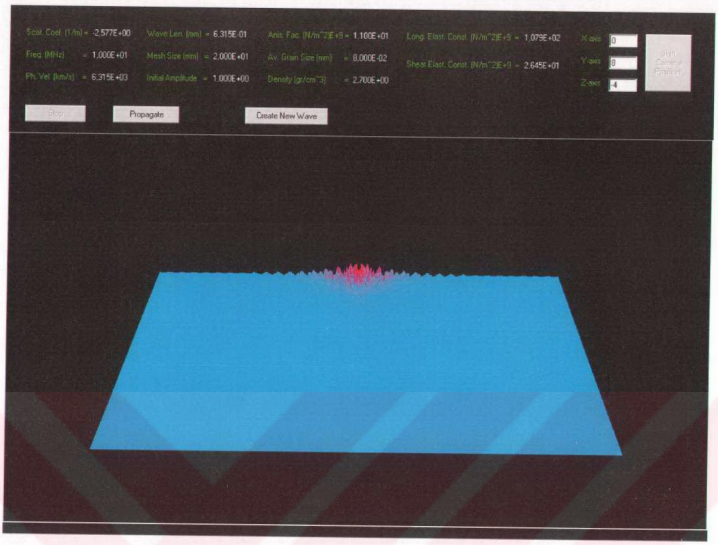

Figure 5.10 Wave simulation in Aluminum with a frequency of 10 MHz and with an average grain size of 0.08 mm. (20 mm element size)

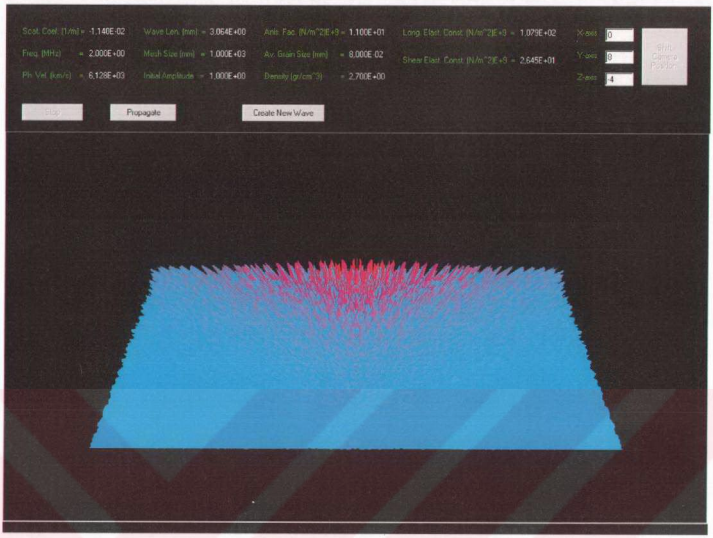

Figure 5.11 Wave simulation in Aluminum with a frequency of 2 MHz and with an average grain size of 0.08 mm. (1000 mm element size)

Decreasing frequency to 2 MHz, affected the scattering as dramatically as decreasing the average grain size. In this situation, scattering coefficient was found to be 1.140×10^{-2} m^{-1}. Figure 5.11 resembles Figure 5.9 and they both have an element size of 1000 mm.

5.2.2 Iron

Owing to its high anisotropy, higher scattering coefficient values were expected for the iron cubic crystal structure. Figure 5.12 clearly demonstrates what was expected.

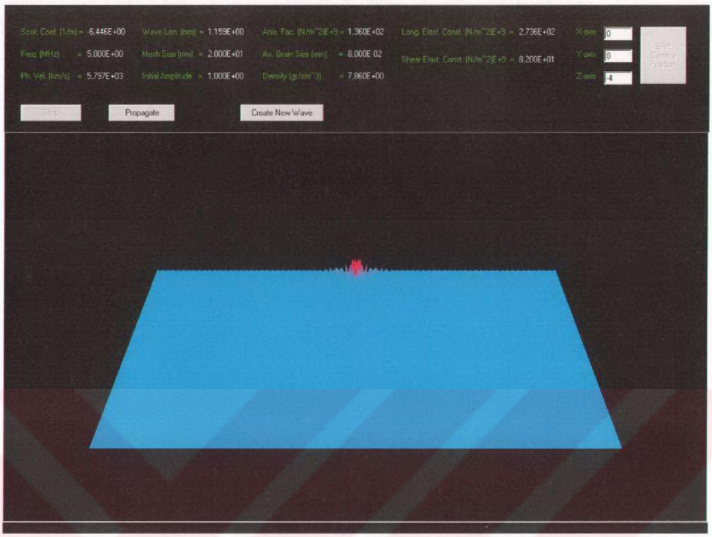

Figure 5.12 Wave simulation in iron with a frequency of 5 MHz and with an average grain size of 0.08 mm. (20 mm element size)

The difference between Figure 5.8 and Figure 5.12 is evident in that longitudinal waves in iron much rapidly attenuate when compared to those in aluminum. In the case of iron, the scattering coefficient was computed as 6.446 m^{-1}, approximately 20 times as much as that of aluminum.

When the grain size was decreased to 0.02 mm, similar dramatic decrease in scattering coefficient was observed (by a factor of approx. 50) as in the case of aluminum. For this grain size, the scattering coefficient was found to be $1{,}335\times10^{-1}$ m^{-1}. It must be underlined that wave simulation in Figure 5.13 has an element size of 500 mm.

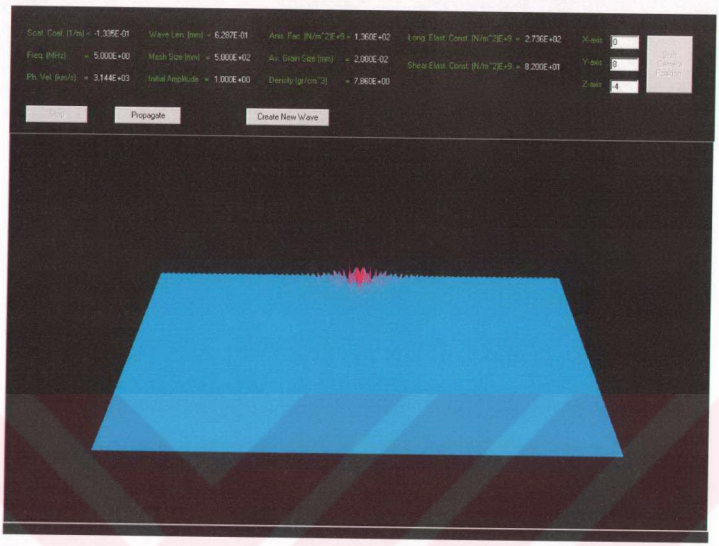

Figure 5.13 Wave simulation in iron with a frequency of 5 MHz and with an average grain size of 0.02 mm. (500 mm element size)

The Figure 5.14 represents the case for 10 MHz frequency. Since the element size was chosen (0.25 mm) smaller than the wavelength, one can see the sinusoidal waves clearly. Duplicating the frequency resulted in a scattering coefficient of $5{,}398\times10^{1}$ m^{-1}, which is a considerably high value especially, compared to previous results.

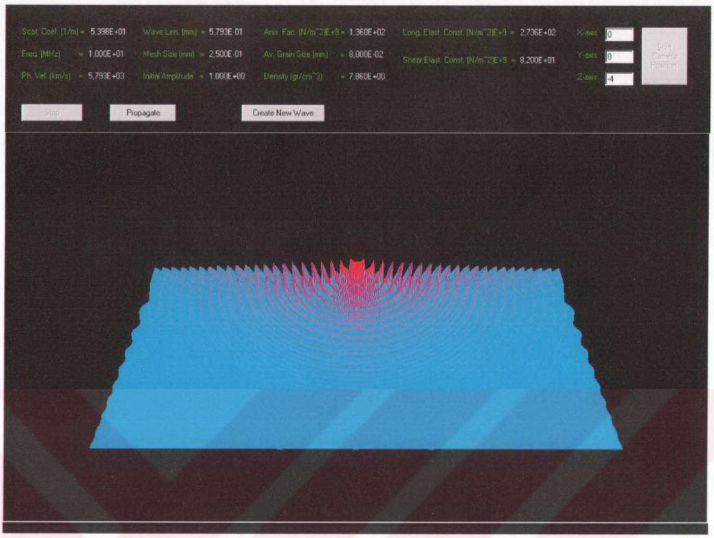

Figure 5.14 Wave simulation in iron with a frequency of 10 MHz and with an average grain size of 0.08 mm. (0.25 mm element size)

Decreasing the frequency from 5 MHz to 2 MHz visually indicates the high frequency dependency of scattering coefficient in iron (Figure 5.15). The scattering coefficient was calculated as $2{,}187\times10^{-1}$ m^{-1} for the element size of 100 mm.

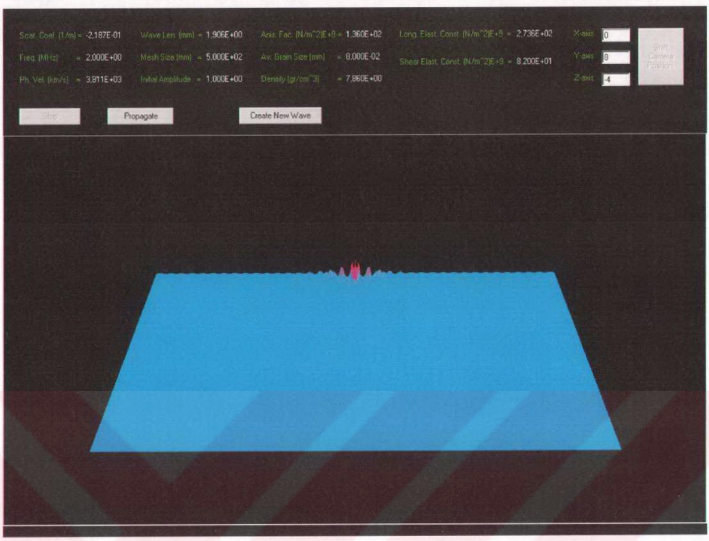

Figure 5.15 Wave simulation in iron with a frequency of 2 MHz and with an average grain size of 0.08 mm. (500 mm element size)

Between Figure 5.15 and Figure 5.13, there is such a resemblance as in between Figure 5.11 and Figure 5.9, which are for aluminum. The resemblance occurred since they have comparable scattering coefficients. In fact, for both of the couple, the ratio of the scattering coefficients is 1.64.

5.2.3 Fe-30%Ni

The scattering coefficient was found to be 1.142×10^{1} m^{-1} for 5 MHz frequency and 0.08 mm average grain size (Figure 5.16). This value is high by a factor of 1.8 when compared to that of iron. But the simulation views are likely.

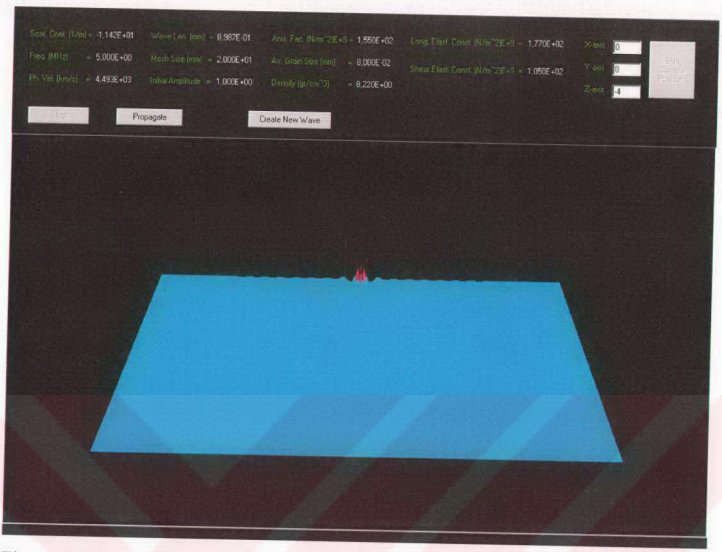

Figure 5.16 Wave simulation in Fe-30%Ni with a frequency of 5 MHz and with an average grain size of 0.08 mm. (20 mm element size)

In Figure 5.17, the simulation of the wave with same frequency is demonstrated in a structure with an average grain size of 0.02 mm.

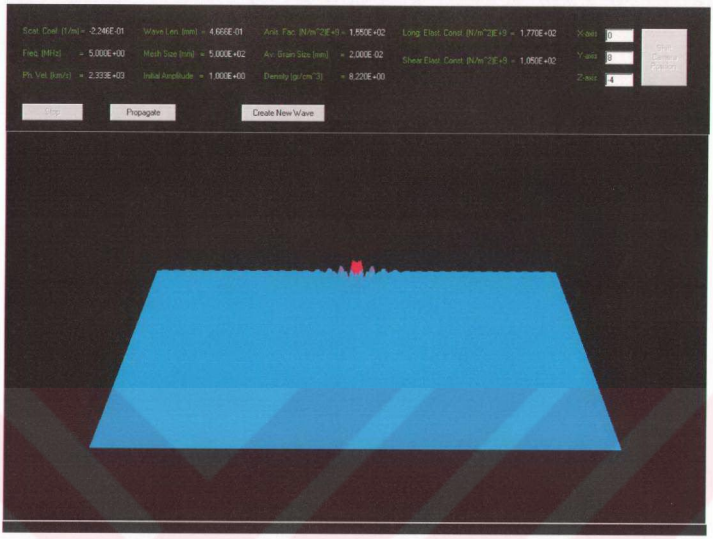

Figure 5.17 Wave simulation in Fe-30%Ni with a frequency of 5 MHz and with an average grain size of 0.02 mm. (500 mm element size)

The slight difference between Figure 5.17 and Figure 5.13 occurred owing to a higher scattering coefficient of Fe-30%Ni (by a factor of approx. 1.8). Scattering coefficient in this case was calculated as 2.246×10^{-1} m^{-1}.

When the frequency was raised to 10 MHz, scattering was calculated as 1.013×10^{2} m^{-1} in a structure with grain size of 0.08 mm. The simulation is shown in Figure 5.18. Again a higher attenuation is observed when compared to Figure 5.14.

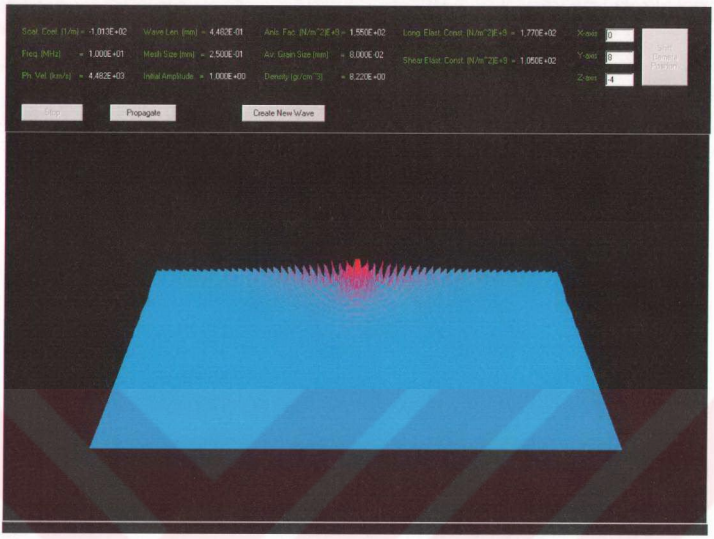

Figure 5.18 Wave simulation in Fe-30%Ni with a frequency of 10 MHz and with an average grain size of 0.08 mm. (0.25 mm element size)

Figure 5.19 shows the simulation when frequency was lowered to 2 MHz. Average grain size was kept 0.08 mm. Element size was equal to the one in Figure 5.17. Scattering coefficient was calculated as 3.717×10^{-1} m^{-1}.

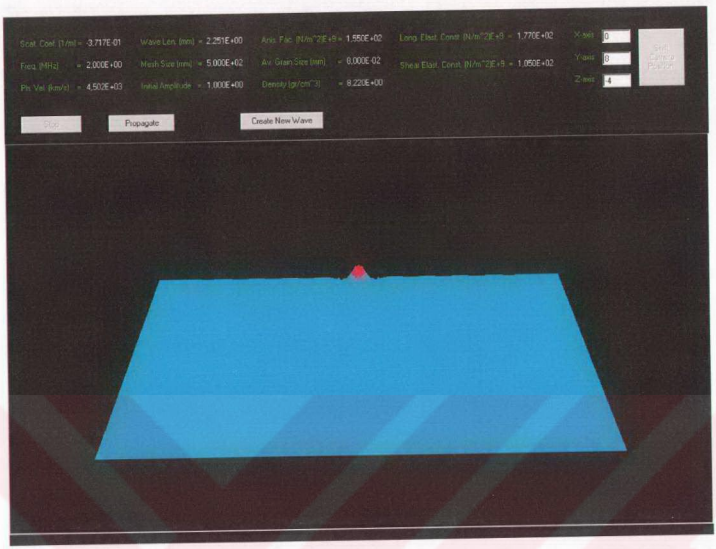

Figure 5.19 Wave simulation of Fe-30%Ni with a frequency of 2 MHz and with an average grain size of 0.08 mm. (500 mm element size)

Numerical results of scattering coefficients for three case studies are tabulated below.

Table 5.2 Scattering Coefficient Results for Aluminum

Aluminum	Frequency(MHz)	Average Grain Size (mm)	Element Size (mm)	Scat. Coeff.(1/m)
Fig. 5.8	5	0.08	20	3.170×10^{-1}
Fig. 5.9	5	0.02	1000	6.958×10^{-3}
Fig. 5.10	10	0.08	20	2.577
Fig. 5.11	2	0.08	1000	1.140×10^{-2}

Table 5.3 Scattering Coefficient Results for Iron

Iron(Fe)	Frequency(MHz)	Average Grain Size (mm)	Element Size (mm)	Scat. Coeff.(1/m)
Fig. 5.12	5	0.08	20	6.446
Fig. 5.13	5	0.02	500	1.335×10^{-1}
Fig. 5.14	10	0.08	0.25	5.398×10^{1}
Fig. 5.15	2-	0.08	500	2.187×10^{-1}

Table 5.4 Scattering Coefficient Results for Fe-30%Ni

Fe-30%Ni	Frequency(MHz)	Average Grain Size (mm)	Element Size (mm)	Scat. Coeff.(1/m)
Fig. 5.16	5	0.08	20	1.142×10^{1}
Fig. 5.17	5	0.02	500	2.246×10^{-1}
Fig. 5.18	10	0.08	0.25	1.013×10^{2}
Fig. 5.19	2	0.08	500	3.717×10^{-1}

CHAPTER 6

CONCLUSION

A model for simulation of ultrasonic wave propagation based on the unified theory of Stanke and Kino, [1] was developed using Visual Basic© programming language. This model is able to monitor elastic plane waves of untextured single phase polycrystals with little anisotropy in two dimensional matrix. Scattering coefficients and phase velocities of the same elastic waves can instantaneously be calculated for a given acoustic frequency, grain size and set of material constants.

Main parameters constructing the model were highlighted as acoustic frequency, grain size, and material constants (density, anisotropy factor, elastic constant). Numerical recipes built with these parameters in the programming stage were verified by the numerical results of literature.

It was made possible with this model that, effects of variation in frequency, grain size or both can be analyzed and monitored very easily.

It is obvious that information about attenuation has many advantages in industrial ultrasonic evaluation. Practically, materials in industry do not appear to be in such properties. Scattering mechanisms of multiphase polycrystals with textured body or with high anisotropy is remained for the further studies. Moreover, absorption suffered by elastic waves was left out of context. Nevertheless it is believed that this study constructs a basis for attenuation and wave modeling researches.

REFERENCES

[1] Stanke F. E., Kino G.S., A Unified Theory for Elastic Wave Propagation in Polycrystalline Materials, J. Acoust. Soc. Am., 75 (1984), 665-681.

[2] Hirsekorn S., The Scattering of Ultrasonic Waves by Polycrystals, J. Acoust. Soc. Am., 72 (1982), 1021-1031.

[3] Hirsekorn S., The Scattering of Ultrasonic Waves by Polycrystals. II. Shear Waves, J. Acoust. Soc. Am., 73 (1983), 1160-1163.

[4] Hirsekorn S., The Scattering of Ultrasonic Waves by Multiphase Polycrystals, J. Acoust. Soc. Am., 83 (1988), 1231-1242.

[5] Karal F. C., Keller J. B., Elastic, Electromagnetic, and Other Waves in Random Medium, J. Math. Phys., 5 (1964), 537-547.

[6] Papadakis E. P., Revised Grain-Scattering Formulas and Tables, J.Acoust. Soc. Am., 37 (1965), 703-710.

[7] Mason W. P., McSkimin H. J., Attenuation and Scattering of High Frequency Sound Waves in Metals and Glasses, J.Acoust. Soc. Am., 19 (1947), 464-473

[8] Pekeris C. L., Scattering of Ultrasound in Polycrystalline Materials, J. Appl. Phys., 19 (1948), 940-946.

[9] Huntington H. B., On Ultrasonic Scattering by Polycrystals, J. Acoust. Soc. Am., 22 (1950), 362-364.

[10] Bhatia A. B., Scattering of High Frequency Sound Waves in Polycrystalline Materials, J. Acoust. Soc. Am., 31 (1959), 16-23.

[11] Bhatia A. B., Moore R. A., Scattering of High Frequency Sound Waves in Polycrystalline Materials, 31 (1959), 1140-1141.

[12] Gould P. L., Introduction to Linear Elasticity,1985.

[13] Chafra S. C., Numerical Methods for Engineering, 1988.

[14] Blitz J., Fundamentals of Ultrasonics, 1963.

[15] MACSYMA Reference Manual (The Mathlab Group, Laboratory for Computer Science, MIT, 1977.

[16] Uysal M., Visual Basic 6.0 ile Yazılım Geliştirme,1999

[17] Keller J. B., Stochastic Equations and Wave Propagation in Random Media, in Proceeding oh the 16th Symposium on Applied Mathematics, American Mathematical Society, New York, 1964, 145-179

APPENDIX A

VISUAL BASIC FORM CODE LIST

Newton (newtonmtd.frm1)

```
Option Base 1
Const PI As Double = 3.14159265358979
Const cone As Double = 2.71828281828

Private Sub Command1_Click()
On Error GoTo errhandler


w = Val(newton.Text5.Text)       'Anisotropy Factor'
Text5.DataField = w

c11 = Val(newton.Text3.Text)     'Long Elastic constant'
Text3.DataField = c11

c44 = Val(newton.Text4.Text)     'Shear Elastic Constant'
Text4.DataField = c44

f = Val(newton.Text7.Text)       'Frequency'
Text7.DataField = f

d = Val(newton.Text6.Text)       'Average grain size'
Text6.DataField = d

dens = Val(newton.Text8.Text)    'Density'
Text8.DataField = dens

meshsize = Val(newton.Text15.Text) 'mesh size
newton.Text15.DataField = meshsize

amplitude = Val(newton.Text16.Text) 'initial amplitude
newton.Text16.DataField = amplitude

Vol = (c11 / dens) ^ 0.5         'initial longitudinal velocity (km/sec)
Vos = (c44 / dens) ^ 0.5         'initial shear velocity  (km/sec)
n = 100

a = 2 * PI * f / Vol * d      'Initial Long. Normalized Frequency x(ol)'
b = a * Vol / Vos            'Initial Shear Normalized Frequency x(os)'

If Option1.Value = True And a > 0.2 Then
Call Longwave(w, c11, a, b, f, d)
```

```
ElseIf Option1.Value = True And a < 0.2 Then
Call RayLong(w, c11, a, Vol, Vos, d, f)

ElseIf Option2.Value = True And b > 2 Then
Call shearwave(w, c11, a, b, f, d)
ElseIf Option2.Value = True And b < 2 Then
Call shearwave(w, c11, a, b, f, d)


Else
MsgBox "Select Wave Type", vbCritical, "No Wave Specified"
End If

newton.Text9.Text = Format(a, "0.000")            'xol
newton.Text9.DataField = a

newton.Text10.Text = Format(b, "0.000")           'xos
newton.Text10.DataField = b

newton.Text11.Text = Format(Vol * 1000, "0.000")      'm/sec
newton.Text11.DataField = Vol * 1000

newton.Text12.Text = Format(Vos * 1000, "0.000")      'm/sec
newton.Text12.DataField = Vos * 1000
Command2.Enabled = True

newton.Text17.Text = Format(newton.Text14.DataField / f / 1000, "0.000E+00") 'wavelength (output)
newton.Text17.DataField = newton.Text14.DataField / f / 1000

errhandler:

If Err.Number = 11 Then
MsgBox "Division by Zero", vbCritical, "Check Input Data"
ElseIf Err.Number = 6 Then
MsgBox "Owerflow", vbCritical, "Check Input Data!"

Exit Sub
End If
End Sub

Private Sub Command2_Click()
Command2.Enabled = False

newton.Hide
Load Form2

End Sub

Private Sub Form_Load()
newton.Width = Screen.Width
newton.Height = Screen.Height
newton.Left = 0
newton.Top = 0


newton.Text3.Text = "273.61" 'c11      For Steel
newton.Text4.Text = "82"    'c44
newton.Text5.Text = "136"   'anisotropy factor
newton.Text6.Text = "0.08"  'grain size
newton.Text7.Text = "5"     'frquency
newton.Text8.Text = "7.86"  'density

newton.Option1.Value = True
newton.Text15.Text = "2"    'mesh size
newton.Text16.Text = "1"    'initial amp
```

```
End Sub

Private Sub Form_Unload(Cancel As Integer)
Endit
End

End Sub
' RenderIt - renders the scene
```

Form 2 (form2.frm)

```
' RenderIt - renders the scene

Sub RenderIt()


Do While EndNow = False
   ' increase the counter
   Counter = Counter + 1

   ' clear the viewport
   D3Ddevice.Clear 1, Viewport(), D3DCLEAR_TARGET, vbBlack, 0, 0

   D3Ddevice.BeginScene

   Createwaves SIZE, Counter
   D3Ddevice.DrawPrimitive D3DPT_TRIANGLELIST, D3DFVF_LVERTEX, Vertices(0), cont,
D3DDP_DEFAULT
   Createwaves -SIZE, Counter
   D3Ddevice.DrawPrimitive D3DPT_TRIANGLELIST, D3DFVF_LVERTEX, Vertices(0), cont,
D3DDP_DEFAULT

   D3Ddevice.EndScene

   ' rotate the matrix

   DX.RotateZMatrix matSpin2, a
   DX.RotateXMatrix matSpin3, PI
   DX.RotateYMatrix matSpin, b
   ' set the new world transform matrix

   D3Ddevice.SetTransform D3DTRANSFORMSTATE_WORLD, matSpin3


   ' copy the backbuffer to the screen
   DX.GetWindowRect hWnd, DestRect
   Primary.Blt DestRect, Backbuffer, SrcRect, DDBLT_WAIT

   DoEvents

Loop

End Sub

Private Sub Command1_Click()
EndNow = True
Form2.Command1.Enabled = False
Form2.Command2.Enabled = True
Form2.Command3.Enabled = True
Form2.Command4.Enabled = False

newton.Text1.Text = ""
newton.Text2.Text = ""
newton.Text13.Text = ""
```

```
newton.Text14.Text = ""
newton.Text17.Text = ""
newton.Text9.Text = ""
newton.Text10.Text = ""
newton.Text11.Text = ""
newton.Text12.Text = ""
End Sub

Private Sub Command2_Click()
Unload Form2
EndNow = False
Load newton
newton.Show
End Sub

Private Sub Command3_Click()
Form2.Command1.Enabled = True
Form2.Command3.Enabled = False
Form2.Command2.Enabled = False
Form2.Command4.Enabled = True

EndNow = False
RenderIt

End Sub

Private Sub Command4_Click()
s1 = Val(Form2.Text1.Text)
s2 = Val(Form2.Text2.Text)
s3 = Val(Form2.Text3.Text)
Call Direct3DInit
End Sub

Private Sub Form_Load()
Show

Form2.Command3.Enabled = False
Form2.Command2.Enabled = False
Form2.Width = Screen.Width
Form2.Height = Screen.Height
Form2.Left = 0
Form2.Top = 0

Form2.Label4.Caption = Format(Val(Str(newton.Text13.DataField)) * 1000, "0.000E+00")
Form2.Label5.Caption = Format(Val(Str(newton.Text7.DataField)), "0.000E+00")
Form2.Label6.Caption = Format(Val(Str(newton.Text14.DataField)), "0.000E+00")

Form2.Label13.Caption = Format(Val(Str(newton.Text17.DataField)), "0.000E+00")
Form2.Label14.Caption = Format(Val(Str(newton.Text15.DataField)), "0.000E+00")
Form2.Label15.Caption = Format(Val(Str(newton.Text16.DataField)), "0.000E+00")

Form2.Label19.Caption = Format(Val(Str(newton.Text5.DataField)), "0.000E+00")
Form2.Label20.Caption = Format(Val(Str(newton.Text6.DataField)), "0.000E+00")
Form2.Label21.Caption = Format(Val(Str(newton.Text8.DataField)), "0.000E+00")

Form2.Label24.Caption = Format(Val(Str(newton.Text3.DataField)), "0.000E+00")
Form2.Label25.Caption = Format(Val(Str(newton.Text4.DataField)), "0.000E+00")

Form2.Label10.Left = 3000
Form2.Label11.Left = 3000
Form2.Label12.Left = 3000

Form2.Label10.Width = 1400
Form2.Label11.Width = 1400
Form2.Label12.Width = 1400
```

```
Form2.Label13.Left = 4440
Form2.Label14.Left = 4440
Form2.Label15.Left = 4440

Form2.Label13.Width = 1095
Form2.Label14.Width = 1095
Form2.Label15.Width = 1095

Form2.Label16.Left = 5640
Form2.Label17.Left = 5640
Form2.Label18.Left = 5640

Form2.Label16.Width = 1800
Form2.Label17.Width = 1800
Form2.Label18.Width = 1800

Form2.Label19.Left = 7480
Form2.Label20.Left = 7480
Form2.Label21.Left = 7480

Form2.Label19.Width = 1095
Form2.Label20.Width = 1095
Form2.Label21.Width = 1095


Form2.Text1.Text = 0
Form2.Text2.Text = 8
Form2.Text3.Text = -4

Form2.Command4.Left = Screen.Width - 1500

Form2.Text1.Left = Screen.Width - 2300
Form2.Text2.Left = Screen.Width - 2300
Form2.Text3.Left = Screen.Width - 2300

Form2.Label7.Left = Form2.Text1.Left - 600
Form2.Label8.Left = Form2.Text1.Left - 600
Form2.Label9.Left = Form2.Text1.Left - 600

Form2.Label22.Left = Form2.Label7.Left - 3800
Form2.Label23.Left = Form2.Label7.Left - 3800

Form2.Label22.Width = 2500
Form2.Label23.Width = 2500

Form2.Label24.Left = Form2.Label7.Left - 3800 + 2540
Form2.Label25.Left = Form2.Label7.Left - 3800 + 2540
Form2.Label24.Width = 1095
Form2.Label25.Width = 1095

s1 = 0
s2 = 8
s3 = -4
Call DirectDrawInit
Call Direct3DInit

' call rendering loop
RenderIt

End Sub


Private Sub Createwaves(Raio As Single, var As Long)
Dim X As Double, Y As Double, z As Double
```

```
Dim temp(0 To 3) As Double

Dim dist As Double
Dim time As Double
Dim disx As Double ' for attenuation in x direction
Dim freq As Double
Dim velocity As Double
Dim attenuation As Double
Dim att1 As Double
Dim mesh As Double
Dim initamp As Double

cont = 0

freq = Val(newton.Text7.Text)   'MHz '.text bec. its on the left-side (not with comma)
velocity = newton.Text14.DataField / 1000  'km/sec (phase velocity)
att1 = newton.Text13.DataField '1/mm
attenuation = att1 * -1      '1/mm
mesh = newton.Text15.DataField
initamp = newton.Text16.DataField

numberofvert = 6 * ((9 * Raio) / DETAIL) ^ 2 'number of vertices needed

Erase Vertices ' clear array
ReDim Vertices(0 To numberofvert)
disx = 0

For X = 0 To 5 Step DETAIL

   dist = disx 'for rayleigh waves make it zero

   For z = 0 To 5 Step DETAIL

   temp(0) = (X ^ 2) + (z ^ 2)
   temp(1) = ((X + DETAIL) ^ 2 + (z) ^ 2)
   temp(2) = ((X) ^ 2 + (z + DETAIL) ^ 2)
   temp(3) = ((X + DETAIL) ^ 2 + (z + DETAIL) ^ 2)

   If temp(0) > 0 And temp(1) > 0 And temp(2) > 0 And temp(3) > 0 Then


      Y = Sqr(temp(0))
      Vertices(cont).X = Raio * X 'for raio=-size wave propagates trough -x direction
      Vertices(cont).z = z
      Vertices(cont).Y = amp(initamp, attenuation, dist, freq, Y, velocity, var, mesh)
      Vertices(cont).Color = DX.CreateColorRGB(Abs(Vertices(cont).Y), 0.5 - Abs(Vertices(cont).Y), 1 -
Abs(Vertices(cont).Y))
      cont = cont + 1
      dist = dist + mesh



      Y = Sqr(temp(1))
      Vertices(cont).X = Raio * X + Raio * DETAIL
      Vertices(cont).z = z
      Vertices(cont).Y = amp(initamp, attenuation, dist, freq, Y, velocity, var, mesh)
      Vertices(cont).Color = DX.CreateColorRGB(Abs(Vertices(cont).Y), 0.5 - Abs(Vertices(cont).Y), 1 -
Abs(Vertices(cont).Y))
      cont = cont + 1
      dist = dist + mesh


      Y = Sqr(temp(2))
      Vertices(cont).X = Raio * X
```

```
      Vertices(cont).z = z + DETAIL
      Vertices(cont).Y = amp(initamp, attenuation, dist, freq, Y, velocity, var, mesh)
      Vertices(cont).Color = DX.CreateColorRGB(Abs(Vertices(cont).Y), 0.5 - Abs(Vertices(cont).Y), 1 -
Abs(Vertices(cont).Y))
      cont = cont + 1
      dist = dist - mesh


      Y = Sqr(temp(1))
      Vertices(cont).X = Raio * X + Raio * DETAIL
      Vertices(cont).z = z
      Vertices(cont).Y = amp(initamp, attenuation, dist, freq, Y, velocity, var, mesh)
      Vertices(cont).Color = DX.CreateColorRGB(Abs(Vertices(cont).Y), 0.5 - Abs(Vertices(cont).Y), 1 -
Abs(Vertices(cont).Y))
      cont = cont + 1
      dist = dist + mesh


      Y = Sqr(temp(2))
      Vertices(cont).X = Raio * X
      Vertices(cont).z = z + DETAIL
      Vertices(cont).Y = amp(initamp, attenuation, dist, freq, Y, velocity, var, mesh)
      Vertices(cont).Color = DX.CreateColorRGB(Abs(Vertices(cont).Y), 0.5 - Abs(Vertices(cont).Y), 1 -
Abs(Vertices(cont).Y))
      cont = cont + 1
      dist = dist + mesh


      Y = Sqr(temp(3))
      Vertices(cont).X = Raio * X + Raio * DETAIL
      Vertices(cont).z = z + DETAIL
      Vertices(cont).Y = amp(initamp, attenuation, dist, freq, Y, velocity, var, mesh)
      Vertices(cont).Color = DX.CreateColorRGB(Abs(Vertices(cont).Y), 0.5 - Abs(Vertices(cont).Y), 1 -
Abs(Vertices(cont).Y))
      cont = cont + 1
      dist = dist - mesh

    End If
    Next
    disx = disx + mesh

Next

End Sub

' The Form_KeyDown event - Also stops the main loop

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeyEscape Then EndNow = True

End Sub

Private Sub Form_Resize()

    picbox.Width = Me.ScaleWidth
    picbox.Height = Me.ScaleHeight - 3000

End Sub

Private Sub Form_Unload(Cancel As Integer)

Load newton
EndNow = False
newton.Show
End Sub
```

APPENDIX B

VISUAL BASIC MODULE CODE LIST

Module 1 (D3DInit.bas)

```
'Direct3DInit - Initializes Direct3D objects

Function Direct3DInit() As Long


Set D3D = DDRAW.GetDirect3D
' create the direct3d object

' create the rendering device - we are using software emulation only
Set D3Ddevice = D3D.CreateDevice("IID_IDirect3DRGBDevice", Backbuffer)

' set the viewport rectangle.
VPdesc.lWidth = DestRect.Right - DestRect.Left
VPdesc.lHeight = DestRect.Bottom - DestRect.Top
VPdesc.minz = 0
VPdesc.maxz = 1
D3Ddevice.SetViewport VPdesc

' cache the viewport rectangle for later use
With Viewport(0)
   .X1 = 0: .Y1 = 0
   .X2 = VPdesc.lWidth
   .Y2 = VPdesc.lHeight
End With



' disable culling
D3Ddevice.SetRenderState D3DRENDERSTATE_CULLMODE, D3DCULL_NONE
'wireframe mode - try using D3DFILL_SOLID too
D3Ddevice.SetRenderState D3DRENDERSTATE_FILLMODE, D3DFILL_SOLID

'flat shading,gouraud
D3Ddevice.SetRenderState D3DRENDERSTATE_SHADEMODE, D3DSHADE_GOURAUD

' set the material
Material.Ambient.r = 1: Material.Ambient.g = 1: Material.Ambient.b = 1
D3Ddevice.SetMaterial Material

' the world matrix - all polygons in world space are transformed by this matrix
DX.IdentityMatrix matWorld ' used to initiate the matrix
```

```
D3Ddevice.SetTransform D3DTRANSFORMSTATE_WORLD, matWorld



' the view matrix - basically the camera position is at -5
' (although it's really just making the whole world at +5)
DX.IdentityMatrix matView ' used to initiate the matrix
DX.ViewMatrix matView, MakeVector(s1, s2, s3), MakeVector(0, 0, 0), MakeVector(0, 1, 0), 0
D3Ddevice.SetTransform D3DTRANSFORMSTATE_VIEW, matView

' the projection matrix - decides how the 3D scene is projected onto the 2D surface
DX.IdentityMatrix matProj ' used to initiate the matrix
DX.ProjectionMatrix matProj, 1, 1000, 3.14 / 2
D3Ddevice.SetTransform D3DTRANSFORMSTATE_PROJECTION, matProj
  D3Ddevice.SetRenderState D3DRENDERSTATE_LIGHTING, False
   D3Ddevice.SetRenderState D3DRENDERSTATE_SHADEMODE, D3DSHADE_FLAT

'lights


End Function
```

Module 2 (DDInit.bas)

```
' DirectDrawInit - Initializes DirectDraw objects
Function DirectDrawInit() As Long

Set DDRAW = DX.DirectDrawCreate("") ' create the directdraw object

DDRAW.SetCooperativeLevel Form2.hWnd, DDSCL_NORMAL ' set the cooperative level, we only need
normal

' set the properties of the primary surface
SurfDesc.lFlags = DDSD_CAPS
SurfDesc.ddsCaps.lCaps = DDSCAPS_PRIMARYSURFACE

Set Primary = DDRAW.CreateSurface(SurfDesc) ' create the primary surface

' set up the backbuffer surface (which will be where we render the 3D view)
SurfDesc.lFlags = DDSD_HEIGHT Or DDSD_WIDTH Or DDSD_CAPS
SurfDesc.ddsCaps.lCaps = DDSCAPS_OFFSCREENPLAIN Or DDSCAPS_3DDEVICE

' use the size of the form to determine the size of the render target
' and viewport rectangle
DX.GetWindowRect Form2.hWnd, DestRect ' set the dimensions of the surface description

SurfDesc.lWidth = DestRect.Right - DestRect.Left
SurfDesc.lHeight = DestRect.Bottom - DestRect.Top

Set Backbuffer = DDRAW.CreateSurface(SurfDesc) ' create the backbuffer surface

' cache the size of the render target for later use
With SrcRect
      .Left = 0: .Top = 0
      .Bottom = SurfDesc.lHeight
      .Right = SurfDesc.lWidth
End With

' create a DirectDrawClipper and attach it to the primary surface.
Set Clipper = DDRAW.CreateClipper(0)
Clipper.SetHWnd Form2.picbox.hWnd
```

```
Primary.SetClipper Clipper

End Function
```

Module 3 (End3D.bas)

```
Public Sub Endit()
   ' Clean up DirectX

   Set D3Ddevice = Nothing
   Set D3D = Nothing
   Set Backbuffer = Nothing
   Set Primary = Nothing
   Set DDRAW = Nothing
   Set DX = Nothing

   End
End Sub
```

Module 4 (LongAlg.bas)

```
Const PI As Double = 3.14159265358979
Public X() As Double
Public Y() As Double

Public Sub Longwave(w, c11, a, b, f, d)
Dim i As Integer
Dim n As Integer

n = 100
ReDim X(n) As Double
ReDim Y(n) As Double
X(1) = a
Y(1) = -0.01
i = 0

Do
i = i + 1
v1 = (2 * X(i) + Y(i) * a) / (4 + a ^ 2)
v2 = (2 * Y(i) - X(i) * a) / (4 + a ^ 2)
v3 = (1 - v1 ^ 2 - v2 ^ 2) / (v1 ^ 2 + (v2 - 1) ^ 2)
v4 = (-2 * v1) / (v1 ^ 2 + (v2 - 1) ^ 2)
v5 = -1 / 2 * Atan2(v4, v3)
v6 = Ln(v3 ^ 2 + v4 ^ 2) / 4
v7 = X(i) ^ 6 - 15 * X(i) ^ 4 * Y(i) ^ 2 + 15 * X(i) ^ 2 * Y(i) ^ 4 - Y(i) ^ 6
v8 = 6 * X(i) ^ 5 * Y(i) - 20 * X(i) ^ 3 * Y(i) ^ 3 + 6 * X(i) * Y(i) ^ 5
v9 = X(i) ^ 5 - 10 * X(i) ^ 3 * Y(i) ^ 2 + 5 * X(i) * Y(i) ^ 4
v10 = 5 * X(i) ^ 4 * Y(i) - 10 * X(i) ^ 2 * Y(i) ^ 3 + Y(i) ^ 5
v11 = X(i) ^ 4 - 6 * X(i) ^ 2 * Y(i) ^ 2 + Y(i) ^ 4
v12 = 4 * X(i) ^ 3 * Y(i) - 4 * X(i) * Y(i) ^ 3
v13 = X(i) ^ 2 - Y(i) ^ 2
v14 = 2 * X(i) * Y(i)
v15 = 12 + 15 * a ^ 2
v16 = 48 + 72 * a ^ 2 + 15 * a ^ 4
v17 = 64 + 48 * a ^ 2 + 12 * a ^ 4 + a ^ 6
v18 = v7 + v11 * v15 + v13 * v16 + v17
```

```
v19 = v8 + v12 * v15 + v14 * v16
v20 = (v18 * v9 + v10 * v19) / (v9 ^ 2 + v10 ^ 2)
v21 = (v9 * v19 - v10 * v18) / (v9 ^ 2 + v10 ^ 2)
v22 = v5 * v20 - v6 * v21
v23 = v5 * v21 + v6 * v20

v24 = (2 * X(i) + Y(i) * b) / (4 + b ^ 2)
v25 = (2 * Y(i) - X(i) * b) / (4 + b ^ 2)
v26 = (1 - v24 ^ 2 - v25 ^ 2) / (v24 ^ 2 + (v25 - 1) ^ 2)
v27 = (-2 * v24) / (v24 ^ 2 + (v25 - 1) ^ 2)
v28 = -1 / 2 * Atan2(v27, v26)
v29 = Ln(v26 ^ 2 + v27 ^ 2) / 4
v30 = 12 - 9 * b ^ 2
v31 = 48 - 24 * b ^ 2 - 9 * b ^ 4
v32 = 64 + 48 * b ^ 2 + 12 * b ^ 4 + b ^ 6
v33 = v7 + v30 * v11 + v13 * v31 + v32
v34 = v8 + v12 * v30 + v14 * v31
v35 = (v33 * v9 + v34 * v10) / (v9 ^ 2 + v10 ^ 2)
v36 = (v34 * v9 - v33 * v10) / (v9 ^ 2 + v10 ^ 2)
v37 = v28 * v35 - v29 * v36
v38 = v28 * v36 + v29 * v35

v39 = -64 * a ^ 2 - 16 * a ^ 2 * (X(i) ^ 2 - Y(i) ^ 2)
v40 = -64 * a ^ 3 - 32 * a ^ 2 * X(i) * Y(i)
v41 = 4 + X(i) ^ 2 - Y(i) ^ 2 - a ^ 2
v42 = 2 * X(i) * Y(i) + 4 * a
v43 = (v39 * v41 + v40 * v42) / (v41 ^ 2 + v42 ^ 2)
v44 = (v40 * v41 - v39 * v42) / (v41 ^ 2 + v42 ^ 2)

v45 = -80 * b ^ 2 - 20 * b ^ 2 * (X(i) ^ 2 - Y(i) ^ 2)
v46 = -80 * b ^ 3 - 40 * b ^ 2 * X(i) * Y(i)
v47 = 4 + X(i) ^ 2 - Y(i) ^ 2 - b ^ 2
v48 = 4 * b + 2 * X(i) * Y(i)
v49 = (v45 * v47 + v46 * v48) / (v47 ^ 2 + v48 ^ 2)
v50 = (v46 * v47 - v45 * v48) / (v47 ^ 2 + v48 ^ 2)

v51 = -16 * (2 * a ^ 2 + b ^ 2)
v52 = 8 * (a - b) + 2 / 3 * (23 * a ^ 3 + 13 * b ^ 3)
v53 = X(i) ^ 2 - Y(i) ^ 2
v54 = 2 * X(i) * Y(i)
v55 = (v51 * v53 + v52 * v54) / (v53 ^ 2 + v54 ^ 2)
v56 = (v52 * v53 - v51 * v54) / (v53 ^ 2 + v54 ^ 2)

v57 = -16 * (a ^ 2 - b ^ 2) - 2 * (a ^ 4 - b ^ 4)
v58 = 8 * (a ^ 3 - b ^ 3) + (a ^ 5 - b ^ 5) + 16 * (a - b)
v59 = (v57 * v11 + v58 * v12) / (v11 ^ 2 + v12 ^ 2)
v60 = (v58 * v11 - v57 * v12) / (v11 ^ 2 + v12 ^ 2)

'                    h(x)                    '
v61 = v22 - v37 + v43 + v49 + v55 + v59
v62 = v23 - v38 + v44 + v50 + v56 + v60 + a - b



v63 = 4 - a ^ 2 + (X(i) ^ 2 - Y(i) ^ 2)
v64 = 4 * a + 2 * X(i) * Y(i)
v65 = (2 * v63 + a * v64) / (v63 ^ 2 + v64 ^ 2)
v66 = (a * v63 - 2 * v64) / (v63 ^ 2 + v64 ^ 2)
v67 = v65 * v20 - v66 * v21
v68 = v65 * v21 + v66 * v20

v69 = v7 - v11 * v15 - 3 * v16 * v13 - 5 * v17
v70 = v8 - v12 * v15 - 3 * v14 * v16
v71 = v69 * v5 - v70 * v6
v72 = v69 * v6 + v70 * v5
```

```
v73 = (v71 * v7 + v72 * v8) / (v7 ^ 2 + v8 ^ 2)
v74 = (v72 * v7 - v71 * v8) / (v7 ^ 2 + v8 ^ 2)

v75 = 4 - b ^ 2 + (X(i) ^ 2 - Y(i) ^ 2)
v76 = 4 * b + 2 * X(i) * Y(i)
v77 = (2 * v75 + b * v76) / (v75 ^ 2 + v76 ^ 2)
v78 = (b * v75 - 2 * v76) / (v75 ^ 2 + v76 ^ 2)
v79 = v77 * v35 - v78 * v36
v80 = v77 * v36 + v78 * v35

v81 = v7 - v30 * v11 - 3 * v13 * v31 - 5 * v32
v82 = v8 - v30 * v12 - 3 * v14 * v31
v83 = (v81 * v7 + v82 * v8) / (v7 ^ 2 + v8 ^ 2)
v84 = (v82 * v7 - v81 * v8) / (v7 ^ 2 + v8 ^ 2)
v85 = v28 * v83 - v29 * v84
v86 = v28 * v84 + v29 * v83

v87 = 32 * a ^ 4 * X(i)
v88 = 32 * a ^ 4 * Y(i)
v89 = v41 ^ 2 - v42 ^ 2
v90 = 2 * v41 * v42
v91 = (v88 * v90 + v87 * v89) / (v89 ^ 2 + v90 ^ 2)
v92 = (v88 * v89 - v87 * v90) / (v89 ^ 2 + v90 ^ 2)

v93 = 40 * b ^ 4 * X(i)
v94 = 40 * b ^ 4 * Y(i)
v95 = v47 ^ 2 - v48 ^ 2
v96 = 2 * v47 * v48
v97 = (v93 * v95 + v94 * v96) / (v95 ^ 2 + v96 ^ 2)
v98 = (v94 * v95 - v93 * v96) / (v95 ^ 2 + v96 ^ 2)

v99 = X(i) ^ 3 - 3 * X(i) * Y(i) ^ 2
v100 = 3 * X(i) ^ 2 * Y(i) - Y(i) ^ 3
v101 = -2 * v51
v102 = -2 * v52
v103 = (v101 * v99 + v102 * v100) / (v99 ^ 2 + v100 ^ 2)
v104 = (v102 * v99 - v101 * v100) / (v99 ^ 2 + v100 ^ 2)

v105 = -4 * v57
v106 = -4 * v58
v107 = (v105 * v9 + v106 * v10) / (v9 ^ 2 + v10 ^ 2)
v108 = (v9 * v106 - v105 * v10) / (v9 ^ 2 + v10 ^ 2)


'                    h'(x)                    '
v109 = v67 + v73 - v79 - v85 + v91 + v97 + v103 + v107
v110 = v68 + v74 - v80 - v86 + v92 + v98 + v104 + v108



v111 = 1 / 525 * w ^ 2 / c11 ^ 2 / a ^ 2

'                    f (X)                    '
v112 = v111 * v13 * v61 - v111 * v14 * v62 + v13 - a ^ 2
v113 = v111 * v13 * v62 + v111 * v14 * v61 + v14


'                    f'(x)                    '
v114 = 2 * v111 * v61 * X(i) - 2 * v111 * v62 * Y(i) + v111 * v13 * v109 - v111 * v14 * v110 + 2 * X(i)
v115 = 2 * v111 * v62 * X(i) + 2 * v111 * v61 * Y(i) + v111 * v13 * v110 + v111 * v14 * v109 + 2 * Y(i)


v116 = (v112 * v114 + v113 * v115) / (v114 ^ 2 + v115 ^ 2)
v117 = (v113 * v114 - v112 * v115) / (v114 ^ 2 + v115 ^ 2)
```

```
X(i + 1) = X(i) - v116
Y(i + 1) = Y(i) - v117
Loop While i < n - 1

newton.Text1.Text = Format(X(i + 1), "0.000")              'Real Part of Wave Number (unitless)
newton.Text1.DataField = X(i + 1)

newton.Text2.Text = Format(Y(i + 1), "0.000E+00")          'Imaginary part of wave number(unitless)
newton.Text2.DataField = Y(i + 1)

newton.Text13.Text = Format(Y(i + 1) / d, "0.000E+00")         'scattering coeff 1/mm (output)
newton.Text13.DataField = Y(i + 1) / d

newton.Text14.Text = Format(2 * PI * f / (X(i + 1) / d) * 1000, "0.000")       'phase velocity (m/sec) (output)
newton.Text14.DataField = 2 * PI * f / (X(i + 1) / d) * 1000

End Sub
Public Function Atan2(ByVal Y As Double, ByVal X As Double) As Double
  If X = 0 And Y = 0 Then
    Atan2 = 0
  ElseIf X < 0 And Y > 0 Then
    Atan2 = Atn(Y / X) + PI
  ElseIf X < 0 And Y < 0 Then
    Atan2 = Atn(Y / X) - PI
  ElseIf X = 0 And Y > 0 Then
    Atan2 = PI / 2
  ElseIf X = 0 And Y < 0 Then
    Atan2 = -PI / 2
  Else
    Atan2 = Atn(Y / X)
  End If
End Function
```

Module 5 (ShearAlg.bas)

```
Const PI As Double = 3.14159265358979
Public X() As Double
Public Y() As Double

Public Sub shearwave(w, c44, a, b, f, d)
Dim i As Integer
Dim n As Integer

n = 100
ReDim X(n) As Double
ReDim Y(n) As Double
X(1) = b
Y(1) = -0.01
i = 0

Do
i = i + 1
v1 = (2 * X(i) + Y(i) * a) / (4 + a ^ 2)
v2 = (2 * Y(i) - X(i) * a) / (4 + a ^ 2)
v3 = (1 - v1 ^ 2 - v2 ^ 2) / (v1 ^ 2 + (v2 - 1) ^ 2)
v4 = (-2 * v1) / (v1 ^ 2 + (v2 - 1) ^ 2)
v5 = -1 / 2 * Atan2(v4, v3)
v6 = Ln(v3 ^ 2 + v4 ^ 2) / 4
v7 = X(i) ^ 6 - 15 * X(i) ^ 4 * Y(i) ^ 2 + 15 * X(i) ^ 2 * Y(i) ^ 4 - Y(i) ^ 6
v8 = 6 * X(i) ^ 5 * Y(i) - 20 * X(i) ^ 3 * Y(i) ^ 3 + 6 * X(i) * Y(i) ^ 5
v9 = X(i) ^ 5 - 10 * X(i) ^ 3 * Y(i) ^ 2 + 5 * X(i) * Y(i) ^ 4
v10 = 5 * X(i) ^ 4 * Y(i) - 10 * X(i) ^ 2 * Y(i) ^ 3 + Y(i) ^ 5
```

```
v11 = X(i) ^ 4 - 6 * X(i) ^ 2 * Y(i) ^ 2 + Y(i) ^ 4
v12 = 4 * X(i) ^ 3 * Y(i) - 4 * X(i) * Y(i) ^ 3
v13 = X(i) ^ 2 - Y(i) ^ 2
v14 = 2 * X(i) * Y(i)
v15 = 12 - 9 * a ^ 2
v16 = 48 - 24 * a ^ 2 + 9 * a ^ 4
v17 = 64 + 48 * a ^ 2 + 12 * a ^ 4 + a ^ 6
v18 = v7 + v11 * v15 + v13 * v16 + v17
v19 = v8 + v12 * v15 + v14 * v16
v20 = (v18 * v9 + v10 * v19) / (v9 ^ 2 + v10 ^ 2)
v21 = (v9 * v19 - v10 * v18) / (v9 ^ 2 + v10 ^ 2)
v22 = v5 * v20 - v6 * v21
v23 = v5 * v21 + v6 * v20

v24 = (X(i) + Y(i) * b) / (1 + b ^ 2)
v25 = (Y(i) - X(i) * b) / (1 + b ^ 2)
v26 = (1 - v24 ^ 2 - v25 ^ 2) / (v24 ^ 2 + (v25 - 1) ^ 2)
v27 = (-2 * v24) / (v24 ^ 2 + (v25 - 1) ^ 2)
v28 = -1 / 2 * Atan2(v27, v26)
v29 = Ln(v26 ^ 2 + v27 ^ 2) / 4
v30 = 12 + 9 * b ^ 2
v31 = 48 + 48 * b ^ 2 + 9 * b ^ 4
v32 = 64 + 48 * b ^ 2 + 12 * b ^ 4 + b ^ 6
v33 = v7 + v30 * v11 + v13 * v31 + v32
v34 = v8 + v12 * v30 + v14 * v31
v35 = (v33 * v9 + v34 * v10) / (v9 ^ 2 + v10 ^ 2)
v36 = (v34 * v9 - v33 * v10) / (v9 ^ 2 + v10 ^ 2)
v37 = v28 * v35 - v29 * v36
v38 = v28 * v36 + v29 * v35

v39 = -64 * a ^ 2 - 16 * a ^ 2 * (X(i) ^ 2 - Y(i) ^ 2)
v40 = -64 * a ^ 3 - 32 * a ^ 2 * X(i) * Y(i)
v41 = 4 + X(i) ^ 2 - Y(i) ^ 2 - a ^ 2
v42 = 2 * X(i) * Y(i) + 4 * a
v43 = (v39 * v41 + v40 * v42) / (v41 ^ 2 + v42 ^ 2)
v44 = (v40 * v41 - v39 * v42) / (v41 ^ 2 + v42 ^ 2)

v45 = -80 * b ^ 2 - 20 * b ^ 2 * (X(i) ^ 2 - Y(i) ^ 2)
v46 = -80 * b ^ 3 - 40 * b ^ 2 * X(i) * Y(i)
v47 = 4 + X(i) ^ 2 - Y(i) ^ 2 - b ^ 2
v48 = 4 * b + 2 * X(i) * Y(i)
v49 = (v45 * v47 + v46 * v48) / (v47 ^ 2 + v48 ^ 2)
v50 = (v46 * v47 - v45 * v48) / (v47 ^ 2 + v48 ^ 2)

v51 = 4 * (4 * a ^ 2 + 5 * b ^ 2)
v52 = 8 * (a - b) - 1 / 3 * (13 * a ^ 3 + 14 * b ^ 3)
v53 = X(i) ^ 2 - Y(i) ^ 2
v54 = 2 * X(i) * Y(i)
v55 = (v51 * v53 + v52 * v54) / (v53 ^ 2 + v54 ^ 2)
v56 = (v52 * v53 - v51 * v54) / (v53 ^ 2 + v54 ^ 2)

v57 = -8 * (a ^ 2 - b ^ 2) - 2 * (a ^ 4 - b ^ 4)
v58 = 8 * (a ^ 3 - b ^ 3) + (a ^ 5 - b ^ 5) + 16 * (a - b)
v59 = (v57 * v11 + v58 * v12) / (v11 ^ 2 + v12 ^ 2)
v60 = (v58 * v11 - v57 * v12) / (v11 ^ 2 + v12 ^ 2)

'                    h(x)                    '
v61 = -v22 + v37 + 5 / 4 * v43 + 7 / 5 * v49 - v55 - v59
v62 = -v23 + v38 + 5 / 4 * v44 + 7 / 5 * v50 - v56 - v60 - a + b



v63 = 4 - a ^ 2 + (X(i) ^ 2 - Y(i) ^ 2)
v64 = 4 * a + 2 * X(i) * Y(i)
v65 = (2 * v63 + a * v64) / (v63 ^ 2 + v64 ^ 2)
```

```
v66 = (a * v63 - 2 * v64) / (v63 ^ 2 + v64 ^ 2)
v67 = v65 * v20 - v66 * v21
v68 = v65 * v21 + v66 * v20

v69 = v7 - v11 * v15 - 3 * v16 * v13 - 5 * v17
v70 = v8 - v12 * v15 - 3 * v14 * v16
v71 = v69 * v5 - v70 * v6
v72 = v69 * v6 + v70 * v5
v73 = (v71 * v7 + v72 * v8) / (v7 ^ 2 + v8 ^ 2)
v74 = (v72 * v7 - v71 * v8) / (v7 ^ 2 + v8 ^ 2)

v75 = 1 - b ^ 2 + (X(i) ^ 2 - Y(i) ^ 2)
v76 = 2 * b + 2 * X(i) * Y(i)
v77 = (v75 + b * v76) / (v75 ^ 2 + v76 ^ 2)
v78 = (b * v75 - v76) / (v75 ^ 2 + v76 ^ 2)
v79 = v77 * v35 - v78 * v36
v80 = v77 * v36 + v78 * v35

v81 = v7 - v30 * v11 - 3 * v13 * v31 - 5 * v32
v82 = v8 - v30 * v12 - 3 * v14 * v31
v83 = (v81 * v7 + v82 * v8) / (v7 ^ 2 + v8 ^ 2)
v84 = (v82 * v7 - v81 * v8) / (v7 ^ 2 + v8 ^ 2)
v85 = v28 * v83 - v29 * v84
v86 = v28 * v84 + v29 * v83

v87 = 40 * a ^ 4 * X(i)
v88 = 40 * a ^ 4 * Y(i)
v89 = v41 ^ 2 - v42 ^ 2
v90 = 2 * v41 * v42
v91 = (v88 * v90 + v87 * v89) / (v89 ^ 2 + v90 ^ 2)
v92 = (v88 * v89 - v87 * v90) / (v89 ^ 2 + v90 ^ 2)

v93 = 56 * b ^ 4 * X(i)
v94 = 56 * b ^ 4 * Y(i)
v95 = v47 ^ 2 - v48 ^ 2
v96 = 2 * v47 * v48
v97 = (v93 * v95 + v94 * v96) / (v95 ^ 2 + v96 ^ 2)
v98 = (v94 * v95 - v93 * v96) / (v95 ^ 2 + v96 ^ 2)

v99 = X(i) ^ 3 - 3 * X(i) * Y(i) ^ 2
v100 = 3 * X(i) ^ 2 * Y(i) - Y(i) ^ 3
v101 = -2 * v51
v102 = -2 * v52
v103 = (v101 * v99 + v102 * v100) / (v99 ^ 2 + v100 ^ 2)
v104 = (v102 * v99 - v101 * v100) / (v99 ^ 2 + v100 ^ 2)

v105 = -4 * v57
v106 = -4 * v58
v107 = (v105 * v9 + v106 * v10) / (v9 ^ 2 + v10 ^ 2)
v108 = (v9 * v106 - v105 * v10) / (v9 ^ 2 + v10 ^ 2)


  '                    h'(x)                    '
v109 = -v67 - v73 + v79 + v85 + v91 + v97 - v103 - v107
v110 = -v68 - v74 + v80 + v86 + v92 + v98 - v104 - v108



v111 = 1 / 1050 * w ^ 2 / c44 ^ 2 / b ^ 2

'                    f (X)                    '
v112 = v111 * v13 * v61 - v111 * v14 * v62 + v13 - b ^ 2
v113 = v111 * v13 * v62 + v111 * v14 * v61 + v14
```

```
'                    f'(x)                    '
v114 = 2 * v111 * v61 * X(i) - 2 * v111 * v62 * Y(i) + v111 * v13 * v109 - v111 * v14 * v110 + 2 * X(i)
v115 = 2 * v111 * v62 * X(i) + 2 * v111 * v61 * Y(i) + v111 * v13 * v110 + v111 * v14 * v109 + 2 * Y(i)


v116 = (v112 * v114 + v113 * v115) / (v114 ^ 2 + v115 ^ 2)
v117 = (v113 * v114 - v112 * v115) / (v114 ^ 2 + v115 ^ 2)

X(i + 1) = X(i) - v116
Y(i + 1) = Y(i) - v117
Loop While i < n - 1

newton.Text1.Text = X(i + 1)
newton.Text1.DataField = X(i + 1)

newton.Text2.Text = Y(i + 1)
newton.Text2.DataField = Y(i + 1)

newton.Text13.Text = Y(i + 1) / d
newton.Text13.DataField = Y(i + 1) / d

newton.Text14.Text = 2 * PI * f / (X(i + 1) / d)
newton.Text14.DataField = 2 * PI * f / (X(i + 1) / d)

End Sub
Public Function Atan2(ByVal Y As Double, ByVal X As Double) As Double
  If X = 0 And Y = 0 Then
    Atan2 = 0
  ElseIf X < 0 And Y > 0 Then
    Atan2 = Atn(Y / X) + PI
  ElseIf X < 0 And Y < 0 Then
    Atan2 = Atn(Y / X) - PI
  ElseIf X = 0 And Y > 0 Then
    Atan2 = PI / 2
  ElseIf X = 0 And Y < 0 Then
    Atan2 = -PI / 2
  Else
    Atan2 = Atn(Y / X)
  End If
End Function
```

Module 6 (RayleighLong.bas)

```
Public Sub RayLong(w, c11, a, Vol, Vos, d, f)
Dim real As Double
Dim img As Double

real = a + 2 / 375 * w ^ 2 / c11 ^ 2 * Vol * (2 + 3 * Vol ^ 2 / Vos ^ 2)
img = -2 / 375 * w ^ 2 / c11 ^ 2 * a ^ 4 / 4 * (2 + 3 * Vol ^ 5 / Vos ^ 5)

newton.Text1.Text = Format(real, "0.000")
newton.Text1.DataField = real

newton.Text2.Text = Format(img, "0.000E+00")
newton.Text2.DataField = img

newton.Text13.Text = Format(img / d, "0.000E+00")  'scattering coefficient (1/mm)
newton.Text13.DataField = img / d

newton.Text14.Text = Format(2 * PI * f / (real / d) * 1000, "0.000")      'phase velocity m/sec
newton.Text14.DataField = 2 * PI * f / (real / d) * 1000
```

```
End Sub
```

Module 7 (Modulemain.bas)

```
Public Sub main()
newton.Command2.Enabled = False
newton.Show

End Sub
```

Module 8 (NaturalLog.bas)

```
Const cone As Double = 2.71828281828
Public Function Ln(z)
Ln = Log(z) / Log(cone)
End Function
```

Module 9 (Wave.bas)

```
' matrices we will need to describe transformations

Public matWorld As D3DMATRIX ' world matrix
Public matView  As D3DMATRIX ' View matrix
Public matProj As D3DMATRIX 'projection matrix

Public matSpin As D3DMATRIX ' this will be used to rotate stuff
Public matSpin2 As D3DMATRIX
Public matSpin3 As D3DMATRIX

Public s1 As Single
Public s2 As Single
Public s3 As Single

Public Vertices() As D3DLVERTEX

Public cont As Double ' used to count vertices
Public Counter As Long ' used to rotate the world matrix
Public Wave As Integer ' used to make a wave


Public EndNow As Boolean ' this tells the program when to end

Public Const DETAIL = 0.05 ' if you decrease  this, there will be more triangles
Public Const SIZE = 1

''''''''''''''''''''''''''''''''''''

Public DX As New DirectX7 ' the main DirectX object

Public DDRAW As DirectDraw7 ' the DirectDraw object

Public SurfDesc As DDSURFACEDESC2 ' used for surface description
Public Primary As DirectDrawSurface7 ' the primary surface - represents the screen
Public Backbuffer As DirectDrawSurface7 ' the backbuffer surface - used for drawing on

Public Clipper As DirectDrawClipper ' the clip - to contain drawing in just the forms window

Public DestRect As RECT ' the source and destination rectangles
Public SrcRect As RECT
```

```
Public D3D As Direct3D7 ' the main Direct3D object
Public D3Ddevice As Direct3DDevice7 ' the rendering device

Public Viewport(0) As D3DRECT ' the viewport rectangle for the rendering device
Public VPdesc As D3DVIEWPORT7 ' the viewport description

Public Material As D3DMATERIAL7 ' the material for our polygon surface

Public Const PI As Double = 3.14159265358979

Public Type tLight
  Light As D3DLIGHT7
  Tag As String
End Type

Public Light() As tLight

Function MakeLight(LightType As CONST_D3DLIGHTTYPE, Ambient As D3DCOLORVALUE, Diffuse
As D3DCOLORVALUE, Specular As D3DCOLORVALUE, Pos As D3DVECTOR, Dir As D3DVECTOR,
Optional attenuation0 As Single = 0, Optional attenuation1 As Single = 0, Optional attenuation2 As Single =
0, Optional FallOff As Single = 0, Optional phi As Single = 0, Optional theta As Single = 0, Optional Range
As Single = 0) As D3DLIGHT7
With MakeLight
 .Ambient = Ambient
 .attenuation0 = attenuation0
 .attenuation1 = attenuation1
 .attenuation2 = attenuation2
 .Diffuse = Diffuse
 .direction = Dir
 .dltType = LightType
 .FallOff = FallOff
 .phi = phi
 .position = Pos
 .Range = Range
 .Specular = Specular
 .theta = theta
End With
End Function
Function MakeD3DCOLORVALUE(a As Single, r As Single, g As Single, b As Single) As
D3DCOLORVALUE
With MakeD3DCOLORVALUE
 .a = a
 .r = r
 .g = g
 .b = b
End With
End Function
Sub AddLight(NewLight As D3DLIGHT7, Tag As String, Enabled As Boolean)
On Error Resume Next
Dim UBL As Long
UBL = UBound(Light) + 1
ReDim Preserve Light(0 To UBL)
Light(UBL).Light = NewLight
Light(UBL).Tag = Tag

D3Ddevice.SetLight UBL, Light(UBL).Light
D3Ddevice.LightEnable UBL, Enabled
End Sub

Function MakeVector(X As Single, Y As Single, z As Single) As D3DVECTOR
' copy x, y and z into the return value
With MakeVector
  .X = X
  .Y = Y
```

```
    .z = z
End With
End Function

Function amp(inamp As Double, alfa As Double, d As Double, f As Double, Y As Double, v As Double, t As
Long, m As Double) As Single

amp = inamp * Exp(-alfa * d) * Sin(2 * PI * f * (Y * (m / 0.1) / (v) - t / (100)))

'alfa : scattering coefficient
'd: dist
' f: frequency
'y: displacement

'v: phase velocity
't: time

End Function
```