

**Visualization of Wave Propagation
in Elastic Solids Using a
Mass-Spring Lattice Model**

by

Shiva Ayyadurai

**S.B. Electrical Engineering
Massachusetts Institute of Technology
(1987)**

**SUBMITTED TO THE MEDIA ARTS &
SCIENCES SECTION
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF**

MASTER OF SCIENCE IN VISUAL STUDIES

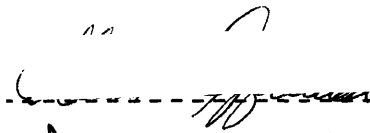
at the

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February, 1990**

© Shiva Ayyadurai, 1989. All rights reserved

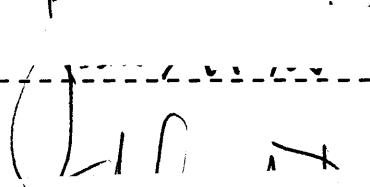
**The author hereby grants to M.I.T. permission to reproduce and to
distribute copies of this thesis document in whole or in part.**

Signature of Author



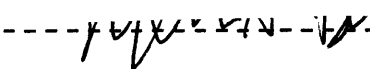
Department of Mechanical Engineering
December, 1989

Certified by



Professor James H. Williams, Jr.
Department of Mechanical Engineering
Thesis Supervisor

Accepted by



Professor Stephen A. Benton, Chairman
Department Graduate Committee
on Graduate Students

Rotch

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

FEB 27 1990



Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
Ph: 617.253.2800
Email: docs@mit.edu
<http://libraries.mit.edu/docs>

DISCLAIMER NOTICE

The accompanying media item for this thesis is available in the MIT Libraries or Institute Archives.

Thank you.



Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
Ph: 617.253.2800
Email: docs@mit.edu
<http://libraries.mit.edu/docs>

DISCLAIMER OF QUALITY

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available. If you are dissatisfied with this product and find it unusable, please contact Document Services as soon as possible.

Thank you.

Page 123 is missing from the original document.

Visualization of Wave Propagation in Elastic Solids Using a Mass-Spring Lattice Model

by

Shiva Ayyadurai

Submitted to the Media Arts & Sciences Section
on December 27, 1989 in partial fulfillment of the
requirements for the Degree of

Master of Science in Visual Studies

Abstract

A technique for modeling and visualizing wave propagation in elastic solids is presented. The model, referred to as the mass-spring lattice model (MSLM), offers three major features: 1) simplicity, 2) usage of a relatively small amount of computer memory, and 3) requirement of a relatively short amount of computing time. In this thesis, the MSLM is developed for two-dimensions; however, its extension to three-dimensions is straightforward and will be pursued in future research.

Following mathematical formulation of the model, four computer graphics techniques are developed for visualizing data generated from the MSLM. The four techniques are termed color contouring, vector-fields, two-dimensional mesh deformation and three-dimensional surface deformation. The techniques of two-dimensional mesh deformation and three-dimensional surface deformation are two new and unique techniques which are a product of this thesis. Each technique is implemented in software and can accept any type of data including displacement, velocity, stress or strain.

Displacement data is generated using the MSLM for three materials: zinc, uni-directional graphite fiber-reinforced epoxy and beryl. The four computer graphics techniques are then applied to these three sets of data to produce twelve video animations which depict wave propagation in the three materials. Production of the video animations involves the configuration of specialized hardware and the development of new software for automatically directing the production of the entire animation. A video tape is attached containing these animations.

Thesis Supervisor: James H. Williams, Jr.
Title: Professor of Mechanical Engineering

Acknowledgments

I wish to dedicate this thesis to my mother, father, sister and my new-born nephew Shivaji.

I wish to thank the following people for their friendship, encouragement and support which was instrumental to my completing this work:

Prof. James H. Williams, Jr., my thesis advisor, his friendship, guidance, support, and understanding will always be remembered.

Prof. David Zeltzer, Prof. Stephen Benton and Prof. David Gossard for their guidance and support in completing this work.

My brother and colleague Fred Foreman and my sister Nadene Foreman for their technical and personal support.

My colleagues Norman Fortenberry, Dave Wooton, Ray Nagem and Han Song Seng.

Reed Sturtevant and Bob Phenix for their friendship, trust, and continued support of my work at Lotus Development Corporation and at M.I.T.

Table of Contents

Abstract	2
Acknowledgments.....	3
Table of Contents.....	4
Chapter 1: Formulation of Mass-Spring Lattice Model	6
Abstract	7
Introduction	8
Mathematical Formulation	10
<i>Analysis of F_x</i>	12
<i>Analysis of F_z</i>	25
Equations of Motion.....	28
Boundary Formulae for Mass-Spring Lattice Model	32
Conclusions and Results.....	34
<i>Isotropic Case</i>	37
<i>Transversely Isotropic Case</i>	38
References	41
Figures.....	42
Chapter 2: Computer Graphics Techniques for Visualizing Wave Propagation	49
Abstract	50
Introduction	51
Configuration of Mesh	55
Color Contouring.....	58
<i>Mapping Colors to Nodal Displacement Data</i>	58
<i>Smooth Shading of Mesh Elements to Create Color Contour</i>	61
Vector Fields	64
Two-Dimensional Mesh Deformation.....	66
Three-Dimensional Surface Deformation	68
<i>Calculating Three-Dimensional Curves</i>	69
<i>Steps in Creating the Three-Dimensional Surface</i>	76

Conclusions	78
References	79
Figures	80
Appendix A: Program for Generating Mesh	95
Appendix B: Program for Calculating Maximum and Minimum Displacement.....	98
Appendix C: Program for Assigning RGB Values to a Node.....	100
Appendix D: Program for Smooth Shading Mesh Element	102
Appendix E: Program for Rendering Vector at Each Node	104
Appendix F: Program for Two-Dimensional Mesh Deformation	110
Appendix G: Program for Calculating Y Component for a Node.....	116
Appendix H: Program for Generating Cubic Spline Surface	118
Chapter 3: Computer Graphics Animations of Wave Propagation in Elastic Solids	123
Abstract	124
Introduction	125
Video	127
References	128
Figures.....	129

Chapter 1:

*Formulation of Mass-Spring
Lattice Model*

Abstract

A mass-spring lattice model in a two-dimensional framework is presented. The model makes use of point masses and simple extensional and rotational springs. Springs and masses are connected in a lattice structure as an idealized representation of the material to be studied. The equations of motion in two-dimensions are developed following an analysis of forces in both x - and z -directions. The mass-spring lattice model can serve to analyze and to predict wave propagation in both isotropic and anisotropic media.

Introduction

Ultrasonic testing is a widely used method for nondestructive evaluation. A knowledge of wave propagation is of enormous benefit for the interpretation of experimental results obtained from ultrasonic testing. Many theories have been developed and applied to the theoretical study of ultrasonic nondestructive testing.

In applying elastic wave theory, for example, computer simulations should be executed and compared with experiments in order to verify the validity of the theory. These simulations can be executed by several schemes which are classified in part according to how the medium is modeled. In this report, the two-dimensional mass-spring lattice model (MSLM) for analyzing elastic wave propagation in both isotropic and anisotropic media is developed. This model can be used to simulate and to visualize wave propagation in these media.

Conventional models for performing such simulations have been shown to possess prohibitive aspects for the purposes of nondestructive evaluation. FDM (*Finite Difference Methods*) require complex boundary conditions to be satisfied [1]. FEM (*Finite Element Methods*) generally require a great deal of computing time and computer memory for executing simulations. Matsuzawa, in order to execute simulations more efficiently, developed a "mass-point system model" [2]; however, it proved to provide no cogent physical significance. Takahashi [3], in order to give this mass-point model more physical significance, modified Matsuzawa's model by developing a potential function for the medium of interest. However, Takahashi's model addressed only static problems, so it was not adequate for wave propagation problems.

Sato [4], using a different approach, developed a mass-point system model by connecting different mass-points with pure extensional springs. Because he included only

simple extensional springs without considering the rotational effects of the diagonal springs, his model works only when $\lambda = \mu$, where λ and μ are Lamé's constants. His model, therefore, is not useful in ultrasonic testing, where $\lambda = \mu$ does not apply in general. Since Sato's model included only the extensional effects of the springs, Harumi [1] modified the model by also including the rotational effects in the diagonal springs in addition to the extensional effects. Harumi's purpose was to develop a model which would emulate media with arbitrary combinations of λ and μ . The MSLM presented here is based on the Harumi model; however, some portions of his model have been modified to give the model more physical meaning. In summary, the MSLM offers three main advantages: 1) it is simple in its approach, 2) it requires relatively little computer memory and 3) it requires relatively little computing time.

Mathematical Formulation

In the MSLM, an elastic continuum is modeled by lumped masses which are connected by elastic springs. The two-dimensional framework of the MSLM is depicted in Fig. 1, where the circles denote the lumped masses and the lines denote the elastic springs having the corresponding spring constants, k_1 through k_8 . In Fig. 1, i and j are the indices of the masses along the x - and z -directions, respectively. The horizontal and vertical spacings between adjacent masses are h . The model assumes that 1) the grid spacing, h , is much less than the shortest wavelength of the elastic wave propagating in the material and 2) the magnitude of the displacement of the masses is smaller than the grid spacing, h .

As shown in Fig. 1, the central mass (i,j) is connected by eight springs to eight adjacent masses. In this model, two types of springs are used: pure extensional and hybrid springs. In making use of these two different types of springs, the goal is to create a more general model which can emulate elastic media having arbitrary elastic constants, as mentioned above in the introduction. In Fig. 1, the horizontal and vertical springs of the model are pure extensional springs which cause forces only in the the x - or z -directions, respectively. However, the diagonal springs in Fig. 1 are hybrid springs. They are a combination of a pure extensional spring and a rotational spring. Fig. 2 illustrates the hybrid diagonal spring as a combination of an extensional spring and a rotational spring.

In the mathematical formulation, the central mass (i,j) is used as the reference. The equations of motion for the central mass are derived by summing all the spring forces acting on the central mass. It is assumed that a perturbing wave causes displacements of each mass in the x - and z - directions. The spring forces on the central mass are formulated, given displacements of each mass caused by this perturbing wave. The spring forces of the diagonal springs are formulated as two components: 1) the force as a result of extension or

compression of the extensional spring and 2) the force as a result of rotation of the rotational spring. The forces of the vertical and horizontal springs, however, involve only the extension or compression of the spring. The approach is to formulate expressions for the net spring forces acting on the central mass in the x - and z -directions, F_x and F_z , respectively. Once F_x and F_z are formulated, Newton's momentum principle is applied to derive the equations of motion for the central mass.

Analysis of F_x

F_x denotes the x -component of the total spring force on the central mass. There are three components which contribute to F_x : 1) the extension and compression of the two horizontal springs, 2) the extension and compression of the four hybrid diagonal springs and 3) the rotation of the four hybrid diagonal springs. Therefore,

$$F_x = F_{xh} + F_{xd} + F_{xr} \quad (1)$$

where F_{xh} denotes the force in the x -direction due to the extension and compression of the horizontal springs, F_{xd} denotes the force in the x -direction due to the extension and compression of the diagonal spring, and F_{xr} denotes the force in the x -direction due to the rotation of the diagonal spring.

F_{xh} : Contribution to F_x from Horizontal Springs

There are two horizontal springs k_1 and k_5 , as evident from Fig. 1. A perturbing wave causes masses (i,j) and $(i+1,j)$, for example, to experience displacements in the positive x -direction as shown in Fig. 3a. The net displacement of the horizontal spring k_5 , connecting the mass $(i+1,j)$ and the central mass, is

$$\Delta u = (u_{i+1,j} - u_{i,j}) \quad (2)$$

where u denotes the horizontal displacement of the mass indexed by the subscripts.

The force on the central mass, in the positive x -direction, due to k_5 is therefore

$$F = k_5(u_{i+1,j} - u_{i,j}) \quad (3)$$

Similarly, in Fig. 3b, the net displacement of the horizontal spring k_1 , connecting the mass $(i-1,j)$ and the central mass, is

$$\Delta u = (u_{i-1,j} - u_{i,j}) \quad (4)$$

The force on the central mass, in the positive x -direction, due to k_1 is therefore

$$F = k_1(u_{i-1,j} - u_{i,j}) \quad (5)$$

Thus, the total force on the central mass, in the positive x -direction, due to the horizontal springs, from eqns. (3) and (5), is

$$F_{xh} = k_1(u_{i-1,j} - u_{i,j}) + k_5(u_{i+1,j} - u_{i,j}) \quad (6)$$

F_{xd}: Contribution to F_x from Extension of Diagonal Springs

F_{xd} is the sum of the forces on the central mass in the positive x -direction caused by the compression and extension of diagonal springs having spring constants k_2, k_4, k_6 and k_8 . Thus, there are four components which contribute to F_{xd} .

$$F_{xd} = F_{x2} + F_{x4} + F_{x6} + F_{x8} \quad (7)$$

where F_{x2}, F_{x4}, F_{x6} and F_{x8} are the forces in the positive x -direction due to extension and compression of the diagonal springs, k_2, k_4, k_6 and k_8 , respectively, in Fig. 1.

Extension or compression of the diagonal spring involves a net displacement of the central mass in both x - and z -directions, relative to the adjacent masses connected by the diagonal springs. Since it is assumed that the model is a linear system, superposition holds. Therefore, to find F_{x6} , for example, first the masses are displaced in the x -direction and the spring force on the central mass, F_1 as shown in Fig. 4a, is formulated, and then the masses are displaced in the z -direction and the spring force on the central mass, F_2 as shown in Fig. 4b, is formulated. By superposition, the sum of these two forces is taken to arrive at the total force. Once the total force is found, the x -component of this force is calculated to find

the resultant force in the x -direction.

First, F_1 is formulated. The dotted position in Fig. 4a illustrates the net displacement of the central mass in the x -direction relative to the mass $(i+1, j+1)$; the dotted position in Fig. 4b illustrates the net displacement of the central mass in the z -direction relative to the mass $(i+1, j+1)$; and the dotted position in Fig. 4c illustrates the position of the central mass as the superposition of the two net relative displacements. From Fig. 4a, F_1 is found to be

$$F_1 = k_6 \Delta l \quad (8)$$

where Δl denotes the net displacement of the diagonal spring k_6 , with an original length of $h\sqrt{2}$, as shown in Fig. 4a.

The net relative displacement of the central mass in the x -direction is

$$\Delta u = (u_{i+1, j+1} - u_{i, j}) \quad (9)$$

By using the Pythagorean Theorem, the net displacement of the diagonal spring k_6 , Δl , is derived.

$$(\Delta u + h)^2 + h^2 = (\Delta l + \sqrt{2} h)^2 \quad (10)$$

$$\Delta u^2 + 2\Delta u h + h^2 + h^2 = \Delta l^2 + 2\sqrt{2} h \Delta l + 2h^2 \quad (11)$$

$$\Delta u^2 + 2\Delta u h = \Delta l^2 + 2\sqrt{2} h \Delta l \quad (12)$$

Since Δu and Δl are small, eqn. (12) may be approximated, by neglecting second order terms, as

$$2\Delta u h = 2\sqrt{2} h \Delta l \quad (13)$$

$$\Delta l = \frac{\Delta u}{\sqrt{2}} = \frac{\sqrt{2}}{2} \Delta u \quad (14)$$

$$\Delta l = \frac{\sqrt{2}}{2} \Delta u \quad (15)$$

Substituting eqn. (15) into eqn. (8) gives

$$F_1 = k_6 \frac{\sqrt{2}}{2} \Delta u \quad (16)$$

Next, F_2 is formulated, given a net relative displacement of the central mass in the z-direction as shown in Fig. 4b. The force F_2 , when the diagonal spring k_6 is displaced by Δl as shown in Fig. 4b, is

$$F_2 = k_6 \Delta l \quad (17)$$

and, the net relative displacement of the central mass in the z-direction is

$$\Delta w = (w_{i+1,j+1} - w_{i,j}) \quad (18)$$

By using the Pythagorean Theorem, the net displacement of the diagonal spring k_6 , Δl , is calculated as

$$\Delta l = \frac{\sqrt{2}}{2} \Delta w \quad (19)$$

Therefore from eqns. (17) and (19),

$$F_2 = k_6 \frac{\sqrt{2}}{2} \Delta w \quad (20)$$

From Fig. 4c, F_1 and F_2 are superposed to find the total force in the diagonal spring with spring constant k_6 to be

$$F_1 + F_2 = k_6 \frac{\sqrt{2}}{2} (\Delta u + \Delta w) \quad (21)$$

Since only F_{x6} , the horizontal component of the total force in eqn. (21), is desired,

$$F_{x6} = (F_1 + F_2) \frac{\sqrt{2}}{2} \quad (22)$$

Using eqn. (21),

$$F_{x6} = \frac{\sqrt{2}}{2} \left\{ k_6 \frac{\sqrt{2}}{2} (\Delta u + \Delta w) \right\} \quad (23)$$

Substituting for Δu and Δw , using eqns. (9) and (18), respectively, gives

$$F_{x6} = \frac{k_6}{2} \{ (u_{i+1,j+1} - u_{i,j}) + (w_{i+1,j+1} - w_{i,j}) \} \quad (24)$$

For the diagonal springs with spring constants k_2 , k_4 and k_8 , the form of F_{x2} , F_{x4} and F_{x8} is similar to F_{x6} ; however, the direction of F_1 and F_2 in each case varies. Similar analysis for these three diagonal springs leads to the following results.

$$F_{x2} = \frac{k_2}{2} \{ (u_{i-1,j-1} - u_{i,j}) + (w_{i-1,j-1} - w_{i,j}) \} \quad (25)$$

$$F_{x4} = \frac{k_4}{2} \{ (u_{i+1,j-1} - u_{i,j}) - (w_{i+1,j-1} - w_{i,j}) \} \quad (26)$$

$$F_{x8} = \frac{k_8}{2} \{ (u_{i-1,j+1} - u_{i,j}) - (w_{i-1,j+1} - w_{i,j}) \} \quad (27)$$

The net horizontal force acting on the central mass, caused by the extensional effects of all the diagonal springs attached to the central mass, is the sum of the horizontal components of the forces in those diagonal springs, as shown in eqn. (7).

Therefore, eqn. (7), with eqns. (24) through (27), results in

$$F_{xd} = \frac{k_2}{2} \{ (u_{i-1,j-1} - u_{i,j}) + (w_{i-1,j-1} - w_{i,j}) \} \\ + \frac{k_4}{2} \{ (u_{i+1,j-1} - u_{i,j}) - (w_{i+1,j-1} - w_{i,j}) \}$$

$$\begin{aligned} & +\frac{k_6}{2}\{(u_{i+1,j+1}-u_{i,j})+(w_{i+1,j+1}-w_{i,j})\} \\ & +\frac{k_8}{2}\{(u_{i-1,j+1}-u_{i,j})-(w_{i-1,j+1}-w_{i,j})\} \end{aligned} \tag{28}$$

F_x : Contribution to F_x from Rotation of Diagonal Springs

Angular rotations of the hybrid diagonal springs are caused by a relative movement of the central mass in the x - and z -directions. Since there are four hybrid diagonal springs attached to the central mass, four corresponding torques, one from each of the hybrid diagonal springs, are induced due to this relative displacement of the central mass. Each torque corresponds to a force on the central mass. The forces on the central mass, caused by the torque of the hybrid diagonal springs, k_2 , k_4 , k_6 and k_8 , are denoted by F_{x2} , F_{x4} , F_{x6} and F_{x8} , respectively.

The hybrid diagonal spring with spring constant k_6 is examined first; then, this case is used to derive the similar forces on the central mass caused by the other hybrid springs attached to the central mass. It is assumed that the reactive torque, T , produced by the angular rotation of the hybrid diagonal spring is proportional to the angle of rotation (see Fig. 5). Furthermore, the constant of proportionality is denoted by the rotational spring constant, k_r . Therefore,

$$T = k_r \Delta\theta \quad (29)$$

The corresponding force acting on the central mass due to the generated torque in eqn. (29) is

$$F = \frac{T}{L} \quad (30)$$

where L , the length of the spring in Fig. 5, is the moment arm of the corresponding force.

In examining the force caused by the rotation of the hybrid diagonal spring k_6 , superposition is used. From Fig. 6a, T_1 , the torque due to the relative displacement of the central mass in the x -direction, is calculated. Then, from Fig. 6b, T_2 , the torque due to the relative displacement of the central mass in the z -direction, is calculated. The corresponding

forces, F_1 and F_2 , of each torque, T_1 and T_2 , are calculated using eqn. (30). Superposing F_1 and F_2 allows the calculation of the total force on the central mass due to the rotation of the hybrid diagonal spring. The x -component of this total force is calculated to find the total horizontal force on the central mass due to the rotation of the hybrid diagonal spring.

In calculating the total force due to the rotation of the spring, it is assumed that the angle of rotation is sufficiently small. In Fig. 6a, for example, this is depicted.

$$\sin(\Delta\theta) = \frac{\Delta r}{\sqrt{2}h} \quad (31)$$

Since, $\Delta\theta$ is small,

$$\sin(\Delta\theta) = \frac{\Delta r}{\sqrt{2}h} = \Delta\theta \quad (32)$$

Therefore,

$$\Delta r = h\sqrt{2}\Delta\theta \quad (33)$$

In order to calculate $\Delta\theta$ in terms of the grid spacing, h , and the net relative displacement of the central mass in the x -direction, Δu , the Pythagorean Theorem is used.

$$(\Delta u)^2 = (\Delta l)^2 + (\Delta r)^2 \quad (34)$$

Since the angle of rotation is sufficiently small, from eqn. (15),

$$\Delta l = \frac{\sqrt{2}}{2}\Delta u \quad (35)$$

Therefore, eqn. (34), with eqns. (33) and (35), results in

$$(\Delta u)^2 = \frac{(\Delta u)^2}{2} + (h\sqrt{2}\Delta\theta)^2 \quad (36)$$

Thus,

$$\Delta\theta = \frac{\Delta u}{2h} \quad (37)$$

Let $k_6\alpha$ denote the rotational spring constant of the hybrid diagonal spring, k_r , in Fig.

6. Therefore, using eqns. (29) and (37), the torque T_1 is found to be

$$T_1 = \frac{(k_6\alpha\Delta u)}{(2h)} \quad (38)$$

The corresponding force, on the central mass, of this torque T_1 is denoted by F_{T_1} and can be calculated using eqn. (30). Using eqn. (35), the moment arm L in eqn. (30) is

$$L = h\sqrt{2} + \Delta l = h\sqrt{2} + \frac{\sqrt{2}}{2}\Delta u \quad (39)$$

as shown in Fig. 6a.

Using eqns. (30) and (39), the corresponding force F_{T_1} is

$$\begin{aligned} F_{T_1} &= \frac{T_1}{L} \\ &= \frac{k_6\alpha\Delta u}{2h\left(h\sqrt{2} + \frac{\sqrt{2}}{2}\Delta u\right)} \\ &= \frac{k_6\alpha}{2h\sqrt{2}} \frac{\Delta u}{\left(h + \frac{\Delta u}{2}\right)} \\ &= \frac{k_6\alpha}{2\sqrt{2}h^2} \Delta u \end{aligned} \quad (40)$$

where, in the last expression, F_{T_1} is approximated by neglecting second or higher order terms of Δu , since Δu is assumed to be small.

Similarly for Fig. 6b,

$$T_2 = \frac{(k_6 \alpha \Delta w)}{(2h)} \quad (41)$$

The corresponding force, on the central mass, of this torque T_2 is denoted by F_{T_2} and can be calculated using eqn. (30). Using eqn. (19), the moment arm L in eqn. (30) is

$$L = h\sqrt{2} + \Delta l = h\sqrt{2} + \frac{\sqrt{2}}{2} \Delta w \quad (42)$$

as shown in Fig. 6b.

Using eqns. (30) and (42), the corresponding force F_{T_2} is

$$\begin{aligned} F_{T_2} &= \frac{T_2}{L} \\ &= \frac{k_6 \alpha \Delta w}{2h \left(h\sqrt{2} + \frac{\sqrt{2}}{2} \Delta w \right)} \\ &= \frac{k_6 \alpha}{2h\sqrt{2}} \frac{\Delta w}{\left(h + \frac{\Delta w}{2} \right)} \\ &= \frac{k_6 \alpha}{2\sqrt{2} h^2} \Delta w \end{aligned} \quad (43)$$

where, in the last expression, F_{T_2} is approximated by neglecting second or higher order terms of Δw , since Δw is assumed to be small.

The forces F_{T_1} and F_{T_2} are in opposite directions as shown in Fig. 6a and Fig. 6b. Therefore, the total force on the central mass due to the rotation of the hybrid diagonal spring, having rotational spring constant $k_6 \alpha$, from eqns. (40) and (43), is

$$F = F_{T_1} - F_{T_2} = \frac{k_6 \alpha}{2\sqrt{2} h^2} (\Delta u - \Delta w) \quad (44)$$

For convenience, let

$$\beta = \frac{\alpha}{2h^2} \quad (45)$$

Therefore,

$$F = \frac{\sqrt{2}}{2} k_6 \beta (\Delta u - \Delta w) \quad (46)$$

F in eqn. (46) is the total force acting on the central mass caused by the rotation of the hybrid diagonal spring having rotational spring constant $k_6\alpha$; however, at this time, the interest is on the x -component of this force. The x -component of this force for spring k_6 is denoted by F_{x6} . Thus, from eqn. (46),

$$F_{x6} = k_6 \frac{\beta}{2} (\Delta u - \Delta w) \quad (47)$$

where Δu and Δw are the net relative displacements of the central mass in the x - and z -directions, as shown in eqns. (9) and (18), respectively.

Hence,

$$F_{x6} = k_6 \frac{\beta}{2} \{(u_{i+1,j+1} - u_{i,j}) - (w_{i+1,j+1} - w_{i,j})\} \quad (48)$$

The force due to the angular rotation of the other hybrid diagonal springs, having rotational spring constants $k_2\alpha$, $k_4\alpha$ and $k_8\alpha$, is of the same form, as in eqn. (48). However, the direction of T_1 and T_2 in each case varies.

In performing a similar analysis for the other three hybrid diagonal springs,

$$F_{x2} = k_2 \frac{\beta}{2} \{(u_{i-1,j-1} - u_{i,j}) - (w_{i-1,j-1} - w_{i,j})\} \quad (49)$$

$$F_{x4} = k_4 \frac{\beta}{2} \{(u_{i+1,j-1} - u_{i,j}) + (w_{i+1,j-1} - w_{i,j})\} \quad (50)$$

$$F_{x8} = k_8 \frac{\beta}{2} \{(u_{i-1,j+1} - u_{i,j}) + (w_{i-1,j+1} - w_{i,j})\} \quad (51)$$

The total force on the central mass in the x -direction due to the rotation of the four hybrid diagonal springs is denoted by F_x . F_x , by superposition, is

$$F_x = F_{x2} + F_{x4} + F_{x6} + F_{x8} \quad (52)$$

Therefore, using eqns. (48) through (52),

$$\begin{aligned} F_x = & k_2 \frac{\beta}{2} \{(u_{i-1,j-1} - u_{i,j}) - (w_{i-1,j-1} - w_{i,j})\} \\ & + k_4 \frac{\beta}{2} \{(u_{i+1,j-1} - u_{i,j}) + (w_{i+1,j-1} - w_{i,j})\} \\ & + k_6 \frac{\beta}{2} \{(u_{i+1,j+1} - u_{i,j}) - (w_{i+1,j+1} - w_{i,j})\} \\ & + k_8 \frac{\beta}{2} \{(u_{i-1,j+1} - u_{i,j}) + (w_{i-1,j+1} - w_{i,j})\} \end{aligned} \quad (53)$$

Total Force in X-Direction, F_x

The final result for F_x , the total horizontal force on the central mass exerted by the attached springs, can now be given. To summarize, F_x is the total force in the x -direction, caused by the compression and extension of the two horizontal springs, F_{zd} is the total force

in the x -direction, caused by the compression and extension of the four diagonal springs, and F_x is the total force in the x -direction, caused by the rotation of the diagonal springs.

Since F_{xh} , F_{xd} and F_x have been formulated in the previous three sections, an expression for F_x based on eqns. (1), (6), (28) and (53), can be written as

$$\begin{aligned}
F_x = & k_1(u_{i-1,j} - u_{i,j}) + k_5(u_{i+1,j} - u_{i,j}) \\
& + \frac{k_2}{2} \{ (u_{i-1,j-1} - u_{i,j}) + (w_{i-1,j-1} - w_{i,j}) \} \\
& + \frac{k_4}{2} \{ (u_{i+1,j-1} - u_{i,j}) - (w_{i+1,j-1} - w_{i,j}) \} \\
& + \frac{k_6}{2} \{ (u_{i+1,j+1} - u_{i,j}) + (w_{i+1,j+1} - w_{i,j}) \} \\
& + \frac{k_8}{2} \{ (u_{i-1,j+1} - u_{i,j}) - (w_{i-1,j+1} - w_{i,j}) \} \\
& + k_2 \frac{\beta}{2} \{ (u_{i-1,j-1} - u_{i,j}) - (w_{i-1,j-1} - w_{i,j}) \} \\
& + k_4 \frac{\beta}{2} \{ (u_{i+1,j-1} - u_{i,j}) + (w_{i+1,j-1} - w_{i,j}) \} \\
& + k_6 \frac{\beta}{2} \{ (u_{i+1,j+1} - u_{i,j}) - (w_{i+1,j+1} - w_{i,j}) \} \\
& + k_8 \frac{\beta}{2} \{ (u_{i-1,j+1} - u_{i,j}) + (w_{i-1,j+1} - w_{i,j}) \}
\end{aligned} \tag{54}$$

Therefore, eqn. (54) is the complete expression for F_x .

Analysis of F_z

F_z denotes the z-component of the total spring force on the central mass. There are three components which contribute to F_z : 1) the extension and compression of the two vertical springs, 2) the extension and compression of the four hybrid diagonal springs and 3) the rotation of the four hybrid diagonal springs. Therefore,

$$F_z = F_{zv} + F_{zd} + F_{zr} \quad (55)$$

where F_{zv} denotes the force in the z-direction due to the extension and compression of the vertical springs, F_{zd} denotes the force in the z-direction due to the extension and compression of the diagonal springs, and F_{zr} denotes the force in the z-direction due to the rotation of the diagonal springs. In formulating F_z , extensive use of the results from calculating F_x is made.

F_{zv} : Contribution to F_z from Vertical Springs

The results for F_{zv} are analogous to the results for F_{xh} . The force in the z-direction generated by the vertical spring having spring constant k_7 is

$$k_7(w_{i,j+1} - w_{i,j}) \quad (56)$$

The force generated in the z-direction by the vertical spring having spring constant k_3 is

$$k_3(w_{i,j-1} - w_{i,j}) \quad (57)$$

Therefore, the total force in the z-direction, F_{zv} , generated by the two vertical springs is

$$F_{zv} = k_3(w_{i,j-1} - w_{i,j}) + k_7(w_{i,j+1} - w_{i,j}) \quad (58)$$

F_{zd} : Contribution to F_z from Extension of Diagonal Springs

Following the formulation for F_{xd} , an expression for F_{zd} , the extensional contribution of the four diagonal springs, can also be derived. F_{zd} is expressed as

$$F_{zd} = F_{z2} + F_{z4} + F_{z6} + F_{z8} \quad (59)$$

Furthermore,

$$F_{z2} = \frac{k_2}{2} \{(u_{i-1,j-1} - u_{i,j}) + (w_{i-1,j-1} - w_{i,j})\} \quad (60)$$

$$F_{z4} = -\frac{k_4}{2} \{(u_{i+1,j-1} - u_{i,j}) - (w_{i+1,j-1} - w_{i,j})\} \quad (61)$$

$$F_{z6} = \frac{k_6}{2} \{(u_{i+1,j+1} - u_{i,j}) + (w_{i+1,j+1} - w_{i,j})\} \quad (62)$$

$$F_{z8} = -\frac{k_8}{2} \{(u_{i-1,j+1} - u_{i,j}) - (w_{i-1,j+1} - w_{i,j})\} \quad (63)$$

Substituting eqns. (60) through (63) into eqn. (59),

$$\begin{aligned} F_{zd} &= \frac{k_2}{2} \{(u_{i-1,j-1} - u_{i,j}) + (w_{i-1,j-1} - w_{i,j})\} \\ &\quad - \frac{k_4}{2} \{(u_{i+1,j-1} - u_{i,j}) - (w_{i+1,j-1} - w_{i,j})\} \\ &\quad + \frac{k_6}{2} \{(u_{i+1,j+1} - u_{i,j}) + (w_{i+1,j+1} - w_{i,j})\} \\ &\quad - \frac{k_8}{2} \{(u_{i-1,j+1} - u_{i,j}) - (w_{i-1,j+1} - w_{i,j})\} \end{aligned} \quad (64)$$

F_{zr} : Contribution to F_z from Rotation of Diagonal Springs

In formulating F_{zr} , the derivation used in arriving at F_{xr} is used.

$$F_{zr} = F_{zr2} + F_{zr4} + F_{zr6} + F_{zr8} \quad (65)$$

Furthermore,

$$F_{zr2} = -k_2 \frac{\beta}{2} \{(u_{i-1,j-1} - u_{i,j}) - (w_{i-1,j-1} - w_{i,j})\} \quad (66)$$

$$F_{zr4} = k_4 \frac{\beta}{2} \{(u_{i+1,j-1} - u_{i,j}) + (w_{i+1,j-1} - w_{i,j})\} \quad (67)$$

$$F_{zr6} = -k_6 \frac{\beta}{2} \{(u_{i+1,j+1} - u_{i,j}) - (w_{i+1,j+1} - w_{i,j})\} \quad (68)$$

$$F_{zr8} = k_8 \frac{\beta}{2} \{(u_{i-1,j+1} - u_{i,j}) + (w_{i-1,j+1} - w_{i,j})\} \quad (69)$$

Substituting eqns. (66) through (69) into eqn. (65),

$$\begin{aligned} F_{zr} = & -k_2 \frac{\beta}{2} \{(u_{i-1,j-1} - u_{i,j}) - (w_{i-1,j-1} - w_{i,j})\} \\ & + k_4 \frac{\beta}{2} \{(u_{i+1,j-1} - u_{i,j}) + (w_{i+1,j-1} - w_{i,j})\} \\ & - k_6 \frac{\beta}{2} \{(u_{i+1,j+1} - u_{i,j}) - (w_{i+1,j+1} - w_{i,j})\} \\ & + k_8 \frac{\beta}{2} \{(u_{i-1,j+1} - u_{i,j}) + (w_{i-1,j+1} - w_{i,j})\} \end{aligned} \quad (70)$$

Total Force in Z-Direction, F_z

The final result for F_z , the total force in the z-direction on the central mass exerted by the attached springs, can be written by using eqns. (55), (58), (64) and (70). To summarize, F_{zv} is the total force in the z-direction, caused by the compression and extension of the two vertical springs, F_{zd} is the total force in the z-direction, caused by the compression and extension of the four diagonal springs, and F_{zr} is the total force in the z-direction, caused by the rotation of the diagonal springs.

Since F_{zv} , F_{zd} and F_{zr} have been formulated, the final expression for F_z is

$$\begin{aligned}
F_z = & k_3(w_{i,j-1} - w_{i,j}) + k_7(w_{i,j+1} - w_{i,j}) \\
& + \frac{k_2}{2} \{(u_{i-1,j-1} - u_{i,j}) + (w_{i-1,j-1} - w_{i,j})\} \\
& - \frac{k_4}{2} \{(u_{i+1,j-1} - u_{i,j}) - (w_{i+1,j-1} - w_{i,j})\} \\
& + \frac{k_6}{2} \{(u_{i+1,j+1} - u_{i,j}) + (w_{i+1,j+1} - w_{i,j})\} \\
& - \frac{k_8}{2} \{(u_{i-1,j+1} - u_{i,j}) - (w_{i-1,j+1} - w_{i,j})\} \\
& - k_2 \frac{\beta}{2} \{(u_{i-1,j-1} - u_{i,j}) - (w_{i-1,j-1} - w_{i,j})\} \\
& + k_4 \frac{\beta}{2} \{(u_{i+1,j-1} - u_{i,j}) + (w_{i+1,j-1} - w_{i,j})\} \\
& - k_6 \frac{\beta}{2} \{(u_{i+1,j+1} - u_{i,j}) - (w_{i+1,j+1} - w_{i,j})\} \\
& + k_8 \frac{\beta}{2} \{(u_{i-1,j+1} - u_{i,j}) + (w_{i-1,j+1} - w_{i,j})\}
\end{aligned} \tag{71}$$

Equations of Motion

With the final expressions for F_x and F_z , Newton's momentum principle is applied in order to develop the equations of motion for this model. Note, for this model, the mass of each lumped mass can be expressed as

$$m = \rho h^2 \quad (72)$$

where ρ denotes the density of the medium and h is the grid spacing in Fig. 1.

By Newton's momentum principle, the equation of motion in the x -direction for the central mass is

$$m \frac{\partial^2 u}{\partial t^2} = F_x \quad (73)$$

Using eqn. (72),

$$\rho h^2 \frac{\partial^2 u}{\partial t^2} = F_x \quad (74)$$

Substituting eqn. (54) for F_x into eqn. (74) and combining like terms,

$$\begin{aligned} \rho h^2 \frac{\partial^2 u}{\partial t^2} &= k_1(u_{i-1,j} - u_{i,j}) + k_5(u_{i+1,j} - u_{i,j}) \\ &+ \frac{k_2}{2}(1 + \beta)(u_{i-1,j-1} - u_{i,j}) \\ &+ \frac{k_4}{2}(1 + \beta)(u_{i+1,j-1} - u_{i,j}) \\ &+ \frac{k_6}{2}(1 + \beta)(u_{i+1,j+1} - u_{i,j}) \\ &+ \frac{k_8}{2}(1 + \beta)(u_{i-1,j+1} - u_{i,j}) \end{aligned}$$

$$\begin{aligned}
& +\frac{k_2}{2}(1-\beta)(w_{i-1,j-1}-w_{i,j}) \\
& -\frac{k_4}{2}(1-\beta)(w_{i+1,j-1}-w_{i,j}) \\
& +\frac{k_6}{2}(1-\beta)(w_{i+1,j+1}-w_{i,j}) \\
& -\frac{k_8}{2}(1-\beta)(w_{i-1,j+1}-w_{i,j})
\end{aligned} \tag{75}$$

The similar formulation is performed for finding an expression describing the motion in the z-direction.

$$m \frac{\partial^2 w}{\partial t^2} = F_z \tag{76}$$

$$\rho h^2 \frac{\partial^2 w}{\partial t^2} = F_z \tag{77}$$

Substituting eqn. (71) for F_z into eqn. (77) and combining like terms,

$$\begin{aligned}
\rho h^2 \frac{\partial^2 w}{\partial t^2} &= k_3(w_{i,j-1}-w_{i,j}) + k_7(w_{i,j+1}-w_{i,j}) \\
& +\frac{k_2}{2}(1+\beta)(w_{i-1,j-1}-w_{i,j}) \\
& +\frac{k_4}{2}(1+\beta)(w_{i+1,j-1}-w_{i,j}) \\
& +\frac{k_6}{2}(1+\beta)(w_{i+1,j+1}-w_{i,j}) \\
& +\frac{k_8}{2}(1+\beta)(w_{i-1,j+1}-w_{i,j})
\end{aligned}$$

$$\begin{aligned}
& +\frac{k_2}{2}(1-\beta)(u_{i-1,j-1}-u_{i,j}) \\
& -\frac{k_4}{2}(1-\beta)(u_{i+1,j-1}-u_{i,j}) \\
& +\frac{k_6}{2}(1-\beta)(u_{i+1,j+1}-u_{i,j}) \\
& -\frac{k_8}{2}(1-\beta)(u_{i-1,j+1}-u_{i,j})
\end{aligned} \tag{78}$$

Boundary Formulae for Mass-Spring Lattice Model

The boundary formulae are best expressed by investigating the treatment of masses in this model. Specifically, the distribution of the point masses to emulate the actual mass of the material is of concern. Fig. 7a diagrams one method of distributing the point masses of the model. The horizontal or vertical distance between adjacent point masses is h . It is assumed that the material is homogeneous and has density ρ . The dotted lines in the diagram signify the "field" of each point mass. This field is defined to be the area over which the point mass covers. In the mass distribution diagramed in Fig. 7a, the masses of the corner, edge, and inner point masses are different.

Let m_{corner} , m_{edge} and m_{inner} denote the masses on the corner, edge and inside, respectively. Because the material is homogenous with density ρ ,

$$m_{corner} = \rho \left(\frac{h}{2} \right)^2 = \rho \frac{h^2}{4} \quad (79)$$

$$m_{edge} = \rho(h) \left(\frac{h}{2} \right) = \rho \frac{h^2}{2} \quad (80)$$

$$m_{inner} = \rho(h)(h) = \rho h^2 \quad (81)$$

Thus, the mass of a point mass on the corner is one-quarter of that of a mass on the inside of the material, and the mass of an edge point mass is one-half of that of a mass on the inside. If the MSLM were to use this scheme in the distribution of masses, special treatment of the boundaries would be required, since the spring constants of the springs along the edge of boundary, as well as the masses of the point masses along the boundary, would be different from those of inner springs and masses.

In the MSLM, the point masses are distributed as shown in Fig. 7b. All the boundary masses in this model are located a distance of $h/2$ inside the boundary of the medium. This allows all the point masses to have equal mass, regardless of their positions.

$$m_{corner} = m_{edge} = m_{inner} \quad (82)$$

Accordingly, the springs connecting the point masses, which are located a distance $h/2$ from the boundary, need not be treated differently. The mass distribution scheme in Fig. 7b is also the way in which Harumi [1] developed his boundary formulae. This method obviates the need for special treatment of the boundary and reduces the complexity of the model. One advantage of this method, from a practical standpoint, is that the MSLM can now handle strip-like cracks without thickness. This feature makes the model useful and powerful in ultrasonic flaw detection.

Conclusion and Results

The mass-spring lattice model has been formulated and the resulting equations of motion have been introduced. In this section, the discussion of the MSLM is concluded by proving the ability of the MSLM to simulate wave propagation in both isotropic and transversely isotropic media.

In both cases, the following simplifications apply: 1) the spring constants k_1 and k_5 of the two horizontal springs are equal, 2) the spring constants k_3 and k_7 of the two vertical springs are equal and 3) the spring constants k_2 , k_4 , k_6 and k_8 of the four diagonal springs are equal.

In summary,

$$k_1 = k_5 \quad (83)$$

$$k_3 = k_7 \quad (84)$$

$$k_2 = k_4 = k_6 = k_8 \quad (85)$$

Therefore eqn. (75) is transformed into

$$\begin{aligned} \rho \frac{\partial^2 u}{\partial t^2} = & k_1 \left\{ \frac{(u_{i+1,j} + u_{i-1,j} - 2u_{i,j})}{h^2} \right\} \\ & + \frac{k_2}{2} \left\{ \frac{(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} - 4u_{i,j})}{h^2} \right\} \\ & + \beta \frac{k_2}{2} \left\{ \frac{(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} - 4u_{i,j})}{h^2} \right\} \\ & + \frac{k_2}{2} \left\{ \frac{(w_{i+1,j+1} - w_{i+1,j-1} - w_{i-1,j+1} + w_{i-1,j-1})}{h^2} \right\} \end{aligned}$$

$$+\beta \frac{k_2}{2} \left\{ \frac{(-w_{i+1,j+1} + w_{i+1,j-1} + w_{i-1,j+1} - w_{i-1,j-1})}{h^2} \right\} \quad (86)$$

Similarly, eqn. (78) is transformed into

$$\begin{aligned} \rho \frac{\partial^2 w}{\partial t^2} = & k_3 \left\{ \frac{(w_{i,j+1} + w_{i,j-1} - 2w_{i,j})}{h^2} \right\} \\ & + \frac{k_2}{2} \left\{ \frac{(w_{i+1,j+1} + w_{i+1,j-1} + w_{i-1,j+1} + w_{i-1,j-1} - 4w_{i,j})}{h^2} \right\} \\ & + \beta \frac{k_2}{2} \left\{ \frac{(w_{i+1,j+1} + w_{i+1,j-1} + w_{i-1,j+1} + w_{i-1,j-1} - 4w_{i,j})}{h^2} \right\} \\ & + \frac{k_2}{2} \left\{ \frac{(u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1})}{h^2} \right\} \\ & + \beta \frac{k_2}{2} \left\{ \frac{(-u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} - u_{i-1,j-1})}{h^2} \right\} \end{aligned} \quad (87)$$

Recall the finite difference formulae [5],

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i-1,j} + u_{i+1,j} - 2u_{i,j}}{h^2} \quad (88)$$

$$\frac{\partial^2 u}{\partial x \partial z} = \frac{u_{i+1,j+1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i-1,j-1}}{4h^2} \quad (89)$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} = \frac{(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} - 4u_{i,j})}{2h^2} \quad (90)$$

and,

$$\frac{\partial^2 w}{\partial z^2} = \frac{w_{i,j-1} + w_{i,j+1} - 2w_{i,j}}{h^2} \quad (91)$$

$$\frac{\partial^2 w}{\partial x \partial z} = \frac{w_{i+1,j+1} - w_{i-1,j+1} - w_{i+1,j-1} + w_{i-1,j-1}}{4h^2} \quad (92)$$

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial z^2} = \frac{(w_{i+1,j+1} + w_{i+1,j-1} + w_{i-1,j+1} + w_{i-1,j-1} - 4w_{i,j})}{2h^2} \quad (93)$$

Eqns. (86) and (87) can be written, using eqns. (88) through (93), as

$$\rho \frac{\partial^2 u}{\partial t^2} = k_1 \frac{\partial^2 u}{\partial x^2} + (1 + \beta)k_2 \left\{ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} \right\} + 2(1 - \beta)k_2 \frac{\partial^2 w}{\partial x \partial z} \quad (94)$$

$$\rho \frac{\partial^2 w}{\partial t^2} = k_3 \frac{\partial^2 w}{\partial z^2} + (1 + \beta)k_2 \left\{ \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial z^2} \right\} + 2(1 - \beta)k_2 \frac{\partial^2 u}{\partial x \partial z} \quad (95)$$

Isotropic Case

The two-dimensional version, on the x - z plane, of the equations of motion for isotropic media in terms of Lamé's constants [6] is

$$\rho \frac{\partial^2 u}{\partial t^2} = (\lambda + 2\mu) \frac{\partial^2 u}{\partial x^2} + (\lambda + \mu) \frac{\partial^2 w}{\partial x \partial z} + \mu \frac{\partial^2 u}{\partial z^2} \quad (96)$$

$$\rho \frac{\partial^2 w}{\partial t^2} = (\lambda + 2\mu) \frac{\partial^2 w}{\partial z^2} + (\lambda + \mu) \frac{\partial^2 u}{\partial x \partial z} + \mu \frac{\partial^2 w}{\partial x^2} \quad (97)$$

Comparing the coefficients of eqns. (94) and (95) with those of eqns. (96) and (97) reveals the following relations between the theoretical elasticity constants and the spring constants in the MSLM.

$$k_1 = k_3 = \lambda + \mu \quad (98)$$

$$k_2 = \frac{\lambda + 3\mu}{4} \quad (99)$$

$$\beta = \frac{\mu - \lambda}{\lambda + 3\mu} \quad (100)$$

The MSLM with the spring constants, which are obtained from eqns. (98) through (100), can simulate wave propagation in an isotropic elastic medium having Lamé's constants λ and μ . Note that if the rotational effect of the diagonal springs had not been developed, eqn. (100) shows that the condition $\lambda = \mu$, assumed in the model of Sato, would be satisfied.

Transversely Isotropic Case

The discussion so far rests on the assumption that the medium is isotropic; however, the MSLM can also be applied to anisotropic media. For transversely isotropic media, in particular, the model can be easily modified by varying the spring constants.

In general, the generalized Hooke's law [7] at a point in an elastic medium is expressed, in terms of the elastic constants c_{ij} , as

$$\sigma_{xx} = c_{11}u_x + c_{12}v_y + c_{13}w_z + c_{14}(v_z + w_y) + c_{15}(w_x + u_z) + c_{16}(u_y + v_x) \quad (101)$$

$$\sigma_{yy} = c_{12}u_x + c_{22}v_y + c_{23}w_z + c_{24}(v_z + w_y) + c_{25}(w_x + u_z) + c_{26}(u_y + v_x) \quad (102)$$

$$\sigma_{zz} = c_{13}u_x + c_{23}v_y + c_{33}w_z + c_{34}(v_z + w_y) + c_{35}(w_x + u_z) + c_{36}(u_y + v_x) \quad (103)$$

$$\tau_{yz} = c_{14}u_x + c_{24}v_y + c_{34}w_z + c_{44}(v_z + w_y) + c_{45}(w_x + u_z) + c_{46}(u_y + v_x) \quad (104)$$

$$\tau_{zx} = c_{15}u_x + c_{25}v_y + c_{35}w_z + c_{45}(v_z + w_y) + c_{55}(w_x + u_z) + c_{56}(u_y + v_x) \quad (105)$$

$$\tau_{xy} = c_{16}u_x + c_{26}v_y + c_{36}w_z + c_{46}(v_z + w_y) + c_{56}(w_x + u_z) + c_{66}(u_y + v_x) \quad (106)$$

where u , v and w denote the displacements at that point in x -, y - and z -directions, respectively, and σ and τ denote the normal and the shear stresses, respectively, and the first subscripts of σ and τ denote the direction normal to the face on which the stresses apply, and the second subscripts of σ and τ denote the direction in which the stresses apply, and c_{ij} are the elastic constants of the medium. Also, in eqns. (101) through (106), the subscripts of displacements stand for differentiation with respect to the variables.

The equations of motion for the general elastic medium is [8]

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} = \rho \ddot{u} \quad (107)$$

$$\frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} = \rho \ddot{v} \quad (108)$$

$$\frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} = \rho \ddot{w} \quad (109)$$

where ρ is the density of the medium.

A transversely isotropic medium, which has the x - y plane as its isotropic plane, is considered. Combining eqns. (101) through (109) with elastic constants for the transversely isotropic medium [9], the two-dimensional equations of motion on a transversely isotropic plane, for example, the x - z plane, are

$$\rho \frac{\partial^2 u}{\partial t^2} = c_{11} \frac{\partial^2 u}{\partial x^2} + c_{44} \frac{\partial^2 u}{\partial z^2} + (c_{44} + c_{13}) \frac{\partial^2 w}{\partial x \partial z} \quad (110)$$

$$\rho \frac{\partial^2 w}{\partial t^2} = c_{33} \frac{\partial^2 w}{\partial z^2} + c_{44} \frac{\partial^2 w}{\partial x^2} + (c_{44} + c_{13}) \frac{\partial^2 u}{\partial x \partial z} \quad (111)$$

Comparing eqns. (110) and (111) with eqns. (94) and (95),

$$k_1 = c_{11} - c_{44} \quad (112)$$

$$k_3 = c_{33} - c_{44} \quad (113)$$

$$k_2 = \frac{3c_{44} + c_{13}}{4} \quad (114)$$

$$\beta = \frac{c_{44} - c_{13}}{3c_{44} + c_{13}} \quad (115)$$

With these new spring constants, the model can be applied to the simulation of elastic wave propagation in a transversely isotropic medium. In particular, many fiber reinforced composite materials and large space structures may be modeled as transversely isotropic media.

The MSLM, in conclusion, can model wave propagation in both the isotropic and transversely isotropic media. Unlike the Harumi [1] model, the MSLM accounts for the torque on the hybrid diagonal springs by considering the displacements of the central mass in *both* x- and z-directions; the Harumi model, however, ignores the possible motions of the central mass in the z-direction when formulating the torque on its hybrid diagonal springs. The MSLM, therefore, is more complete and offers greater physical meaning in modeling wave propagation in isotropic and transversely isotropic media.

References

- [1] K. Harumi and T. Igarashi, "Computer Simulation of Elastic Waves by a New Mass-point System with Potentials", *Journal of Nondestructive Testing of Japan*, Vol. 12, 1978, pp. 807-816.
- [2] T. Matsuzawa, "Vibrations of a System of Particles", *Bull. Tokyo Dom. Sci. Inst.*, Vol. 12, 1972, pp. 66-69.
- [3] H. Takahashi, "Linear Distribution Constant Theory (IV)", *Iawami Basic Engineering Lectures*, Vol. 7, 1973, pp. 307-308.
- [4] Y. Sato, "Reflection and Diffraction at Cracks, Angles, Etc.", *Journal of Nondestructive Testing of Japan*, Vol. 27, 1978, pp. 180-181.
- [5] B. Carnahan, H.A. Luther and J.O. Wilkes, Applied Numerical Methods, John Wiley and Sons, Inc., 1969, pp. 430-431.
- [6] L.E. Malvern, Introduction to the Mechanics of a Continuous Medium, Prentice-Hall, Inc., 1969, pp. 499-500.
- [7] M.J.P. Musgrave, "On the Propagation of Elastic Waves in Aelotropic Media", *Proceedings of the Royal Society of London, Series A* Vol. 225, 1954, pp. 340-341.
- [8] L.E. Malvern, Introduction to the Mechanics of a Continuous Medium, Prentice-Hall, Inc., 1969, pp. 214-215.
- [9] S.G. Lekhnitskii, Theory of Elasticity of an Anisotropic Elastic Body, Holden-Day, Inc., 1963, pp. 23-24.

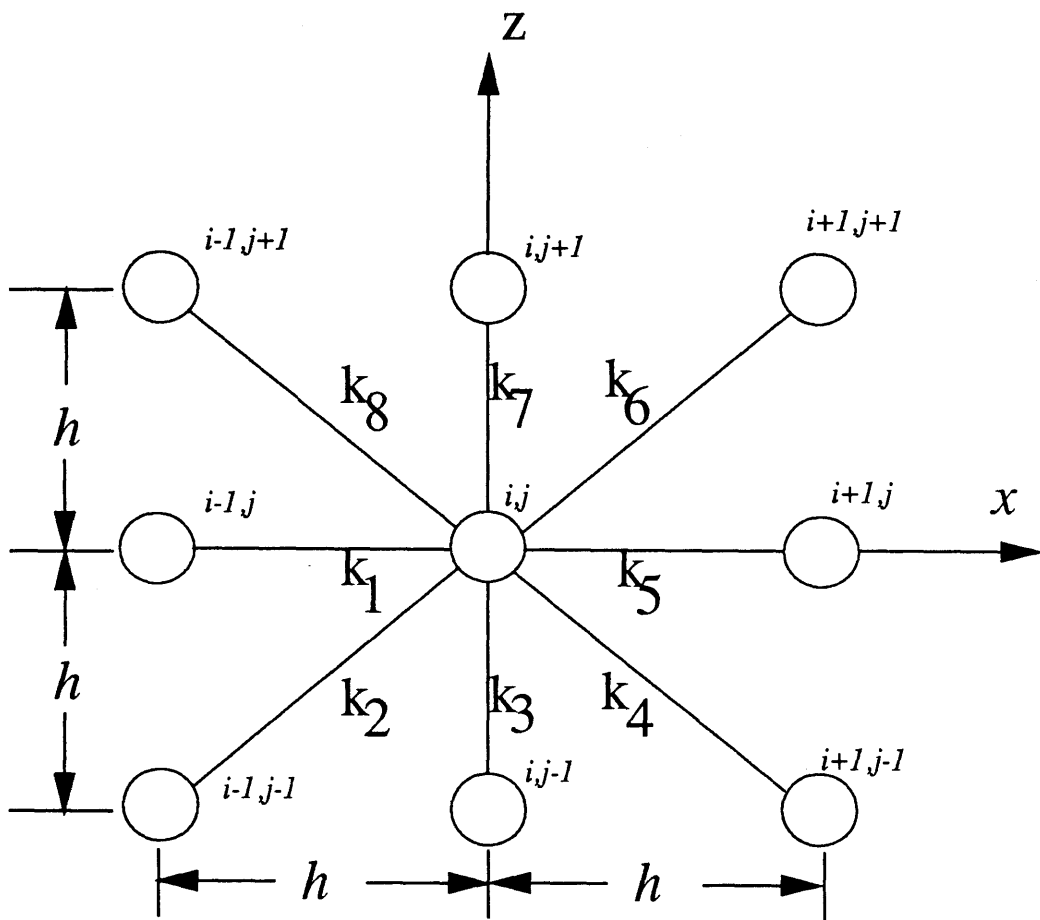


Fig. 1 Mass-spring lattice model.

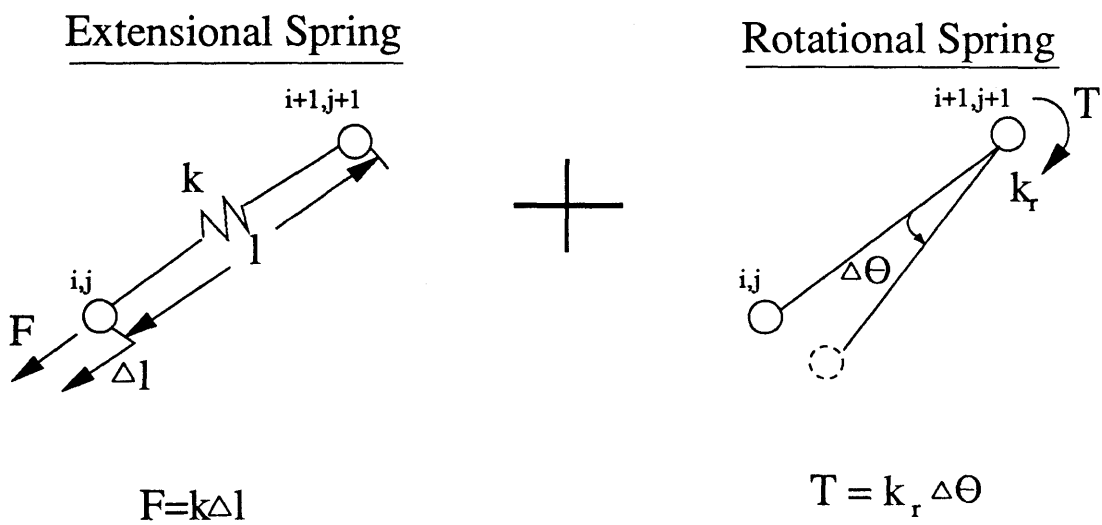


Fig. 2 Hybrid spring constitutive components.

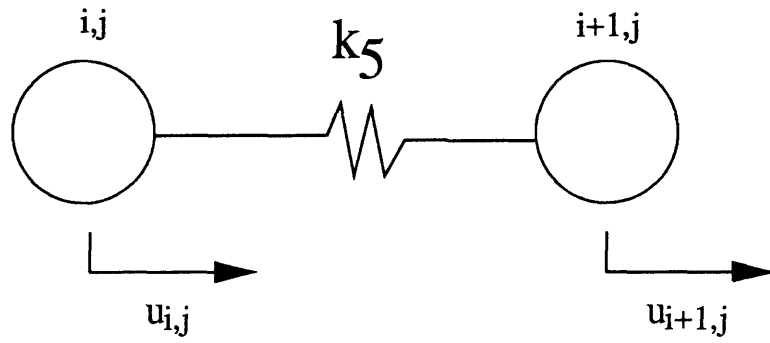


Fig. 3a Displacement of masses connected to k_s spring.

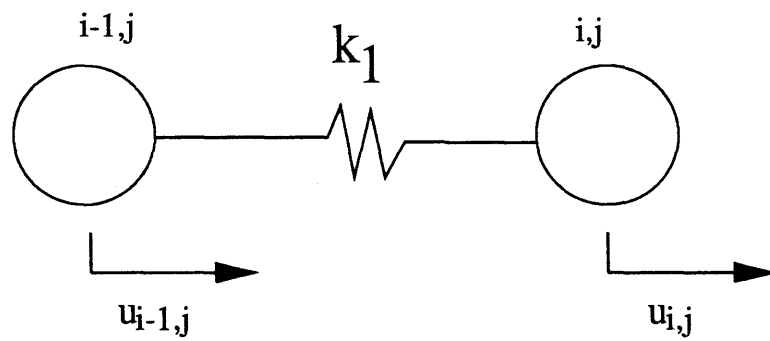


Fig. 3b Displacement of masses connected to k_1 spring.

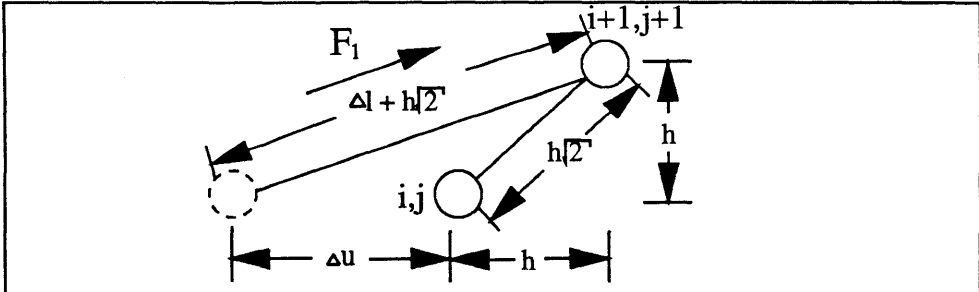


Fig. 4a Elongation of k_6 spring due to net horizontal displacement.

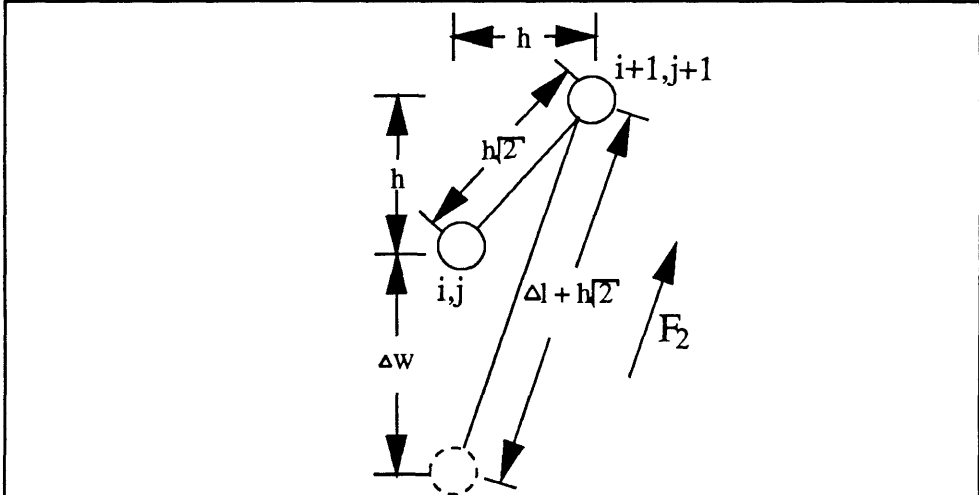


Fig. 4b Elongation of k_6 spring due to net vertical displacement.

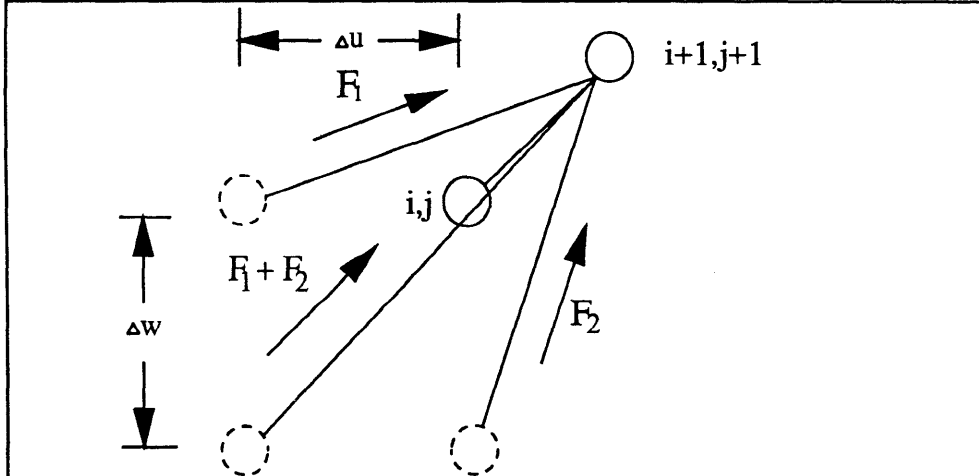


Fig. 4c Total elongation of k_6 spring.

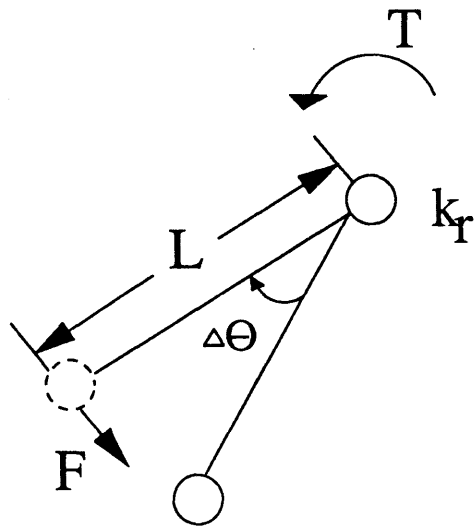
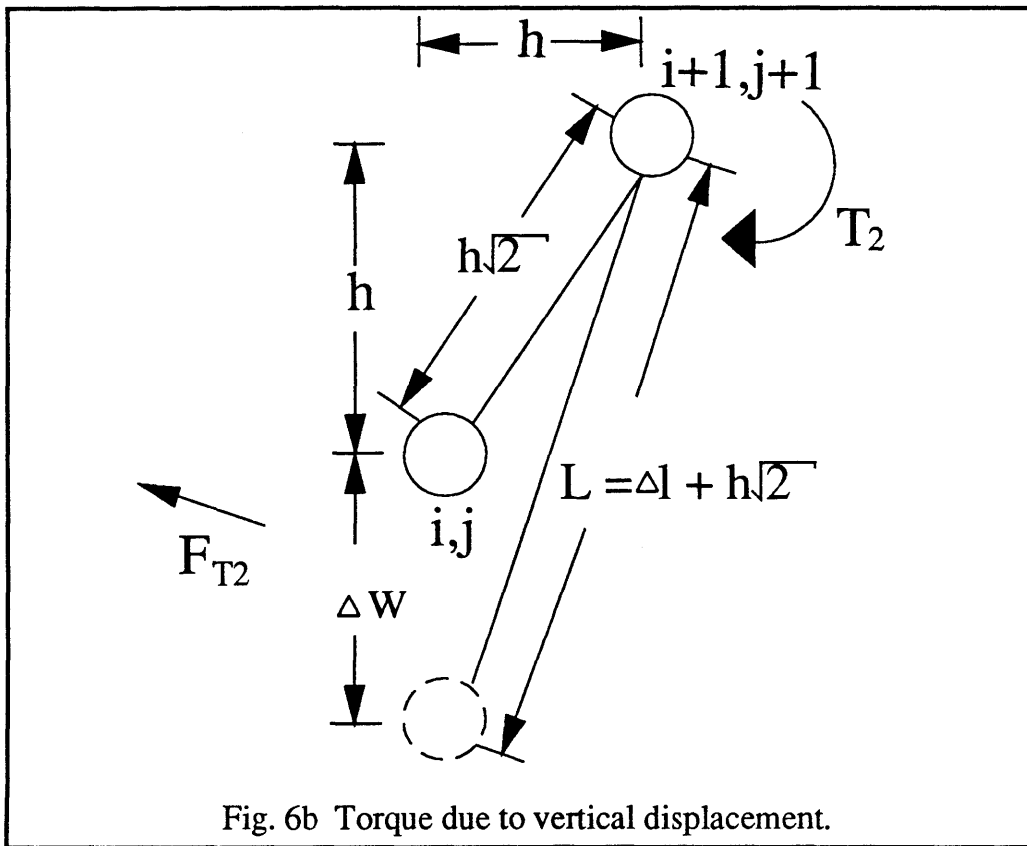
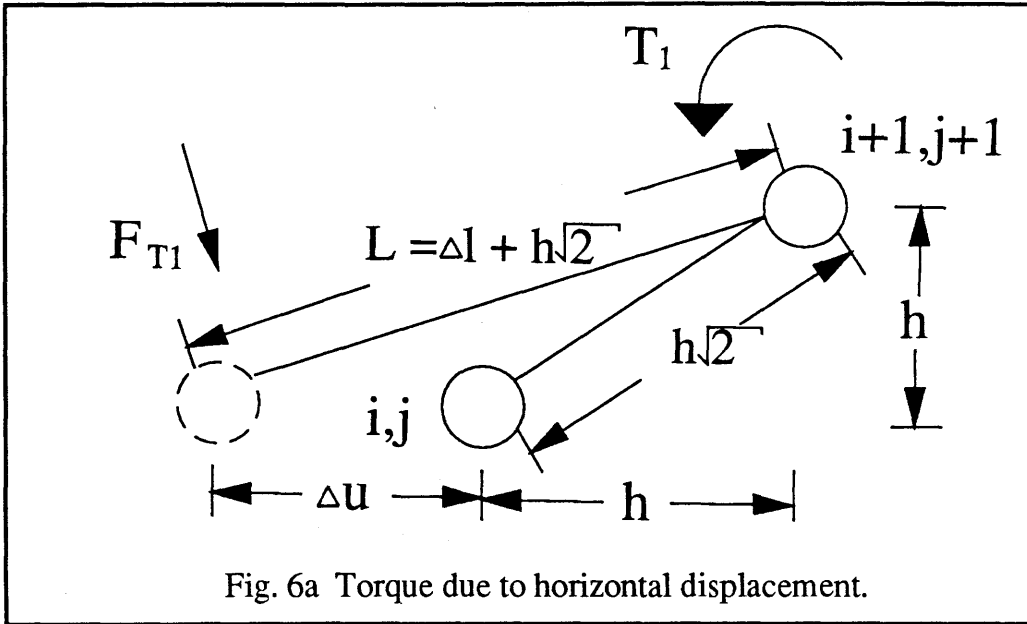


Fig. 5 Torque and corresponding force of rotational spring.



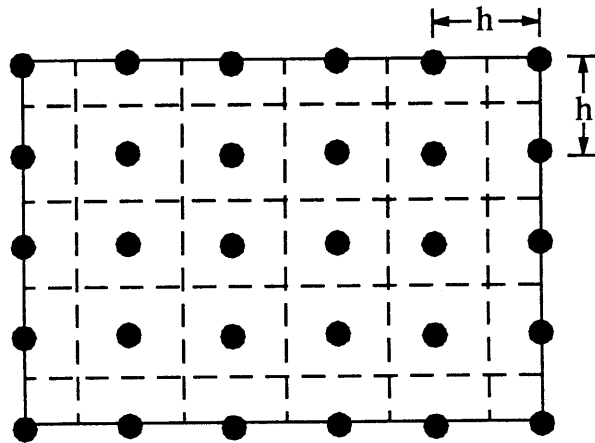


Fig. 7a Conventional mass distribution scheme.

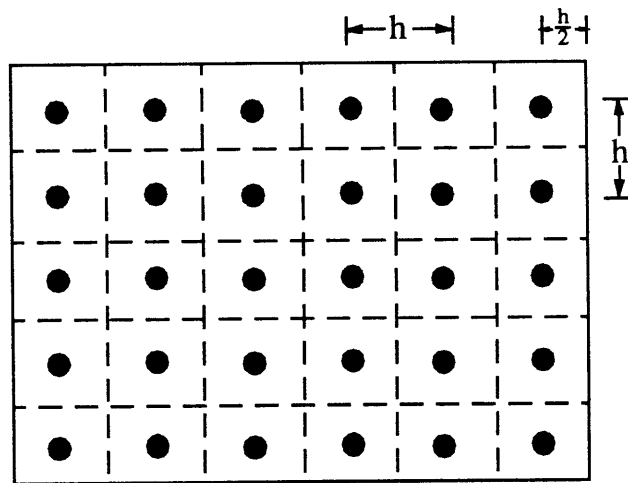


Fig. 7b MSLM mass distribution scheme.

Chapter 2:

*Computer Graphics Techniques for
Visualizing Wave Propagation*

Abstract

Four computer graphics techniques for visualizing two-dimensional wave propagation are presented. The techniques termed color contouring, vector fields, two-dimensional mesh deformation and three-dimensional surface deformation are developed for viewing data generated using a mass-spring lattice model (MSLM) which predicts wave propagation in both isotropic and anisotropic media. The techniques of two-dimensional mesh deformation and three-dimensional surface deformation are two *new* techniques for visualizing wave propagation. Specific software programs and procedures are devised for creating the appearance of wave propagation in two and three dimensions that is associated with these two techniques. The techniques of color contouring and vector fields are traditional visualization techniques, and new methods and procedures are developed for adapting these techniques for the data generated from the MSLM. The displacements of the MSLM point masses in a plane are used as the data for applying the various visualization techniques. All software is written in the C Programming Language to run on the Hewlett-Packard 9000/350 workstation. These visualization techniques also provide the flexibility of using any type of data as input including stress or strain as long as the input/output data protocol described is followed.

Introduction

The use of computer graphics in scientific computation is an emerging field. As a tool for applying computers to science, this emerging field, being termed Visualization in Scientific Computing (ViSC), offers a way to see the unseen [1]. Computer graphics techniques have been used to enhance the understanding of such phenomena as fluid flow, crack propagation and rigid body motion in the field of applied mechanics [2]. However, there is a dearth of information in applying modern computer graphics techniques to visualize wave propagation in elastic solids. This document will serve as one step towards addressing this shortcoming. Four techniques for visualizing two-dimensional wave propagation are devised and presented.

The techniques termed color contouring, vector fields, two-dimensional mesh deformation and three-dimensional surface deformation are developed to visualize wave propagation. Each of these techniques is discussed at length in the sections titled *Color Contouring*, *Vector Fields*, *Two-Dimensional Mesh Deformation* and *Three-Dimensional Surface Deformation*. These techniques are applied to time-dependent data generated from computer solutions created using a mass-spring lattice model (MSLM) [3-4]. The MSLM simulates the medium as a two-dimensional structure consisting of point masses and simple extensional and rotational springs. Since the x-y plane in this model is taken to be the plane of isotropy, the more interesting wave propagation phenomena occur in the x-z plane which is anisotropic. The MSLM, therefore, calculates the displacement in the x- and z-directions of each point mass for every time step. These displacement data are used as the input for the various visualization techniques.

Software for each of the techniques is written in the C Programming Language with the use of computer graphics functions from Hewlett-Packard's Starbase Graphics Library

[5]. The software assumes the medium to be a rectangular mesh with each node of the mesh representing a corresponding MSLM point mass. The features and details of this mesh representation of the medium are described in the next section. Each of the techniques uses the same mesh representation of the medium to generate the computer graphics.

The technique of color contouring assigns color values to each node of the mesh and then performs *smooth shading* on each mesh element to produce a color contoured two-dimensional rectangle. A mapping function calculates a color value for each node of the mesh based on the magnitude of the displacement of the corresponding MSLM point mass. The term *smooth shading* is commonly used in computer graphics to refer to a method of coloring an object. For example, an object such as a rectangle instead of being colored with a homogeneous color can be smooth shaded such that the color of the rectangle varies gradually from one region to another.

The use of color in viewing data has both advantages and disadvantages. Bertin [6] points out that color, in contrast to black-and-white, is richer in cerebral stimulation, and therefore captures and holds the attention of the viewer and assures better retention of the information. Some of the events occurring in wave propagation problems occur in tenths of microseconds. Visualization of such events in color enables the viewer to identify a particular wave interaction phenomenon more easily; whereas, the same phenomenon viewed in black-and-white may go unnoticed. One disadvantage of using color is that the viewer may have anomalies of chromatic perception (for example, color blindness). The more significant disadvantage as Tufte [7] mentions is that the mind does not give visual ordering to colors, except possibly for red to reflect higher levels than other colors. Without attempting to address the problem of chromatic perception, the problem of visual ordering can be solved by providing the viewer with a color legend as shown in Fig. 1 which enables

the viewer to know the corresponding mapping of colors to data ranges.

Vector fields are created using the same displacement data as described previously; however, the magnitude of the displacement and the direction of displacement are mapped to a vector's length and direction, respectively. The computer graphics software routines represent the vector as an arrow element. The arrow element is the most efficient and often the only technique for representing the complex movement of a point [8]. Arrow elements are drawn starting from each node of the mesh. The final visual effect is a region of arrow elements depicting the magnitude and the direction of motion of the MSLM point masses.

The two-dimensional mesh deformation technique depicts the medium as the mesh itself where each node of the mesh is the location of a MSLM point mass. The computer graphics software routines for this technique generate the mesh image by drawing each individual mesh element. The routines represent each mesh element as a polygon. As each point mass displaces in the x- and z-directions, the corresponding nodes of the mesh change their position, and polygon after polygon is drawn to render a new mesh configuration.

The three-dimensional surface deformation technique depicts the medium as a three-dimensional surface situated in x-y-z space. The computer graphics routines for this technique use the framework of the mesh to create a surface which deforms in three-dimensions. The mesh is situated in the x-z plane. The magnitude of the displacement of a point mass is represented as a displacement of a node of the mesh in the y-direction. The mesh in this technique, unlike the previous techniques, is a three-dimensional mesh with each node defined by a coordinate triplet of (x,y,z).

In the subsequent sections, each of the computer graphics techniques for visualizing wave propagation is discussed in detail. Each technique offers the visualization of the

magnitude and direction of wave propagation in uniquely different ways. In the section following immediately, the configuration of the mesh used by the software programs for generating the computer graphics is described.

Configuration of Mesh

Prior to discussing the details of each of the computer graphics techniques used to visualize wave propagation, it is necessary to understand the nature and concept of the mesh that is used as the framework for rendering the images for each of the techniques. The mesh used in this discussion is a discrete geometric representation of a two-dimensional structure situated in the x-z plane. A 5 by 5 mesh, for example, means a mesh consisting of 5 mesh elements in the x-direction and 5 mesh elements in the z-direction. Each node of the mesh is defined by a coordinate pair: (x,z). This 5 by 5 mesh has 25 elements and 36 nodes. In general, T_n , the total number of nodes in a system of quadrilateral mesh elements, can be defined as

$$T_n = (E_x + 1)(E_z + 1) \quad (1)$$

where E_x is the number of mesh elements in the x-direction or the number of columns in the mesh, and E_z is the number of mesh elements in the z-direction or the number of rows in the mesh. Since the mesh described here is a rectangular mesh, it is appropriate to use the terms rows and columns to reference elements within the mesh.

Node and Element Numbering Conventions of Mesh

In developing the computer graphics techniques for mapping data values to a particular node, there is a need to know what node numbers surround any particular mesh element. In Fig. 2, the node numbers as well as the mesh element numbers are shown for a 5 by 5 mesh. The mesh element numbers are in bold. When recalling nodes surrounding a particular

element, the ordering of nodes begins from the lower left node to the lower right node to the upper right node and finally to the upper left node. Mesh element number 13, for example, is surrounded by nodes 15, 16, 22 and 21. Node number 15 is termed the first node of the mesh element, node number 16 is termed the second node of the mesh element, node number 22 is termed the third node of the mesh element, and node number 21 is termed the fourth node of the mesh element.

To express this node numbering convention for a particular element, more generally, a convention is developed. In this convention, the term $N_{i,j}$ is used to denote the j 'th node of the i 'th mesh element. Thus, the nodes of element 13 from the above example can be expressed as follows

$$N_{13,1} = 15 \tag{2}$$

$$N_{13,2} = 16 \tag{3}$$

$$N_{13,3} = 22 \tag{4}$$

$$N_{13,4} = 21 \tag{5}$$

Before formulating a general expression for $N_{i,j}$ it is important to define the notation $\lfloor x \rfloor$, standardized by Knuth [9], to denote the greatest integer less than or equal to x . Using this notation, equations for finding the absolute node numbers of nodes 1, 2, 3 and 4 of a particular mesh element i are expressed as

$$N_{i,1} = i + \left\lfloor \frac{i-1}{E_x} \right\rfloor \quad (6)$$

$$N_{i,2} = i + \left\lfloor \frac{i-1}{E_x} \right\rfloor + 1 \quad (7)$$

$$N_{i,3} = i + \left\lfloor \frac{i-1}{E_x} \right\rfloor + 2 + E_x \quad (8)$$

$$N_{i,4} = i + \left\lfloor \frac{i-1}{E_x} \right\rfloor + 1 + E_x \quad (9)$$

From eqns. (6), (7), (8) and (9), a general expression for $N_{i,j}$ can be written as

$$N_{i,j} = i + \left\lfloor \frac{i-1}{E_x} \right\rfloor + 2 - |j-3| + \left\lfloor \frac{(j-1)}{2} \right\rfloor E_x \quad (10)$$

Therefore for a particular element i , the absolute node number of node j , which surrounds element i , can be found from eqn. (10).

Appendix A contains the software code for generating any rectangular mesh as well as the code for assigning the node numbers for the elements of the mesh by using eqn. (10).

Color Contouring

In this section, the technique of color contouring is presented. While color contouring is a traditional visualization technique for visualizing scalar data, new software and methods had to be developed for adapting and implementing this technique for the mesh configuration discussed in the previous section and for the displacement data generated from the MSLM. In order to visualize wave propagation using color contouring, specific procedures were developed for generating a color contour from the MSLM data. These procedures are the focus of this discussion. All software related to this technique was written in the C Programming Language to run on the Hewlett-Packard 9000/350 personal computer.

There are two steps involved in this technique: 1) mapping colors to the displacement values of each node in the mesh and 2) using the resulting values for color at each node to perform smooth shading on each mesh element to create a color contour of the entire mesh. The displacement values for each node are acquired from the displacement data generated from the MSLM. The form of the data is a pair of numbers which denote the displacements of the node in the x- and z-directions. For example, the mesh in Fig. 2 has data of the form shown in Fig. 3. The displacements of any node j in the x- and z-directions are denoted by u_j and w_j , respectively.

Mapping Colors to Nodal Displacement Data

The two steps for mapping colors to nodal displacement data involve the following: 1) determining the minimum and maximum displacement values among all the nodes for all time steps and 2) using a color mapping algorithm to assign a RGB (red, green, blue) value to each node. The term RGB is used in computer graphics to denote the red, green

and blue components which can be combined to create a spectrum of colors. A RGB value is a set of three numbers, each mapped to discrete values between zero and one, which determines the intensity of each component.

The MSLM model generates displacement data for a specified number of time steps. For each time step, therefore, there are data for the displacement of each node of the mesh. Fig. 4 offers one way of representing such data for n time steps. The first step in mapping colors involves finding the minimum and maximum magnitudes of displacement. The magnitude of displacement for a particular node j and at a particular time step t is denoted by $M_{j,t}$ and is expressed as

$$M_{j,t} = \sqrt{u_{j,t}^2 + w_{j,t}^2} \quad (11)$$

where $u_{j,t}$ and $w_{j,t}$ denote the displacement of node j at time step t in the x - and z -directions, respectively. To evaluate the minimum and maximum magnitudes of displacement, it is necessary to evaluate $M_{j,t}$ for each node j and at each time step t . In Appendix B, this calculation is performed and the minimum and maximum values are evaluated. The minimum and maximum values for the magnitude of displacement among all time steps and among all nodes are denoted by M_{\min} and M_{\max} .

Once M_{\min} and M_{\max} are calculated, the color mapping algorithm can be used to assign RGB values to each node of the mesh for a particular time step. Colors are mapped to each value of $M_{j,t}$ by assigning values to the red, green and blue components of color. If the convention (as indicated in Fig. 1) is to denote high values with shades of red, low values with shades of blue and intermediate values in the color ranges of orange, yellow and green, the RGB mapping described below can be used. In this convention, the red component for

a particular node and time step, denoted by $R_{j,t}$, varies with $M_{j,t}$ as shown by the line in Fig. 5. Mathematically, the relationship between $R_{j,t}$ and $M_{j,t}$ is formulated by considering the slope of the line.

$$\frac{R_{j,t}}{M_{j,t} - M_{\min}} = \frac{1}{M_{\max} - M_{\min}} \quad (12)$$

Therefore the red component is expressed as

$$R_{j,t} = \frac{M_{j,t} - M_{\min}}{M_{\max} - M_{\min}} \quad (13)$$

Similarly, the green component for a particular node and time step, denoted by $G_{j,t}$, varies with $M_{j,t}$ as shown by the two lines in Fig. 6. The green component is described by two expressions, one for the interval between M_{\min} and M_{mid} and the other for the interval between M_{mid} and M_{\max} , where M_{mid} is defined as

$$M_{\text{mid}} = \frac{M_{\max} + M_{\min}}{2} \quad (14)$$

For the interval between M_{\min} and M_{mid} , $G_{j,t}$ is

$$G_{j,t} = \frac{2(M_{j,t} - M_{\min})}{M_{\max} - M_{\min}}, \quad M_{\min} \leq M_{j,t} \leq M_{\text{mid}} \quad (15)$$

For the interval between M_{mid} and M_{\max} , $G_{j,t}$ is

$$G_{j,t} = \frac{2(M_{j,t} - M_{\max})}{M_{\min} - M_{\max}}, \quad M_{\text{mid}} < M_{j,t} \leq M_{\max} \quad (16)$$

Therefore, eqns. (15) and (16) describe the green component for variations in $M_{j,t}$.

The blue component for a particular node and time step, denoted by $B_{j,t}$, varies with $M_{j,t}$ as shown by the line in Fig. 7.

$$B_{j,t} = \frac{M_{j,t} - M_{\max}}{M_{\min} - M_{\max}} \quad (17)$$

Eqns. (13), (15), (16) and (17) describe the assignments of the RGB components for a given node at a specified time step. The range of values of the RGB components, moreover, are

$$0.0 \leq R_{i,t} \leq 1.0 \quad (18)$$

$$0.0 \leq G_{i,t} \leq 1.0 \quad (19)$$

$$0.0 \leq B_{i,t} \leq 1.0 \quad (20)$$

For every $M_{j,t}$, therefore, non-zero values of red, green and blue exist except at M_{\min} and M_{\max} . Appendix C contains the code to assign a RGB value to a value of $M_{j,t}$. Therefore, for a particular time step, each node of the mesh has a RGB value associated with it. In software, this association is performed through the use of arrays. In Fig. 8, a table is used to represent this association. In general, each node j has corresponding values of R_j , G_j and B_j for a particular time step.

It is important to point out that the reason that the color assignment scheme depicted in Figs. 5, 6 and 7 was chosen was in order to generate a smooth transition in color from blue to green to red. The issue of traversing RGB space smoothly across these color ranges is actually quite difficult. The scheme chosen here appeared to work efficiently, however, it is not perhaps the most optimal scheme.

Smooth Shading of Mesh Elements to Create Color Contour

Once RGB values are assigned to each node, the next step is to perform smooth shading for each mesh element to create the color contour. In order to perform smooth shading for a mesh element, it is necessary to know the RGB values of the four nodes of the element. Access to the RGB values of the four nodes of a mesh element can be acquired by knowing the node numbers of the element. The node numbers can be acquired through the use of eqn. (10). The node numbers, in turn, serve as indices for finding the RGB values associated with the node.

An example of locating the RGB values for a particular mesh element, in the 5 by 5 mesh described previously, will serve to illustrate this process. Given mesh element 18, for example, the nodes surrounding mesh element 18 are found using eqn. (10) to be

$$N_{18,1} = 21 \quad (21)$$

$$N_{18,2} = 22 \quad (22)$$

$$N_{18,3} = 28 \quad (23)$$

$$N_{18,4} = 27 \quad (24)$$

From Fig. 8, the RGB value associated with each node is found. In Fig. 9, this association of RGB values with the nodes of mesh element 18 is shown. Once the RGB values associated with the four corners are known, the smooth shading of the mesh element can be performed. Each mesh element is graphically treated as a polygon having five vertices with the first vertex and the fifth vertex being the same. The StarBase Graphics Library [10] contains the subroutines for coloring (or smooth shading) the polygon based on the RGB values assigned to the vertices of the polygon. In Appendix D, the code for performing the smooth shading for each mesh element is shown.

The smooth shading method used, however, does not eliminate Mach band effects.

This effect is one of exaggeration of intensity along a surface. The effect, from a visual standpoint, is caused by lateral inhibition of the receptors in the eye, whose response to light is influenced by adjacent receptors in inverse relation to the distance to the adjacent receptor [14].

As each mesh element is smooth shaded, the final result is an entire smooth shaded mesh. The color contouring technique described is applied to displacement data; however, the same technique can also be used for creating color contours for any other type of data as long as the mapping of colors-to-data protocol described above is followed.

Vector Fields

In this section, the technique of vector fields is presented. While vector fields are a traditional visualization technique for visualizing scalar and vector data, new software and methods had to be developed for adapting and implementing this technique for the mesh configuration discussed in the section titled *Configuration of Mesh* and for the displacement data generated from the MSLM. In order to visualize wave propagation using vector fields, specific procedures were developed for generating the vector fields from the MSLM data. These procedures are the focus of this discussion. All software related to this technique was written in the C Programming Language to run on the Hewlett-Packard 9000/350 personal computer.

The creation of vector fields, like the technique of color contouring, requires the use of the mesh described in the section titled *Configuration of Mesh*. Vector fields offer a way to visualize direction as well as the magnitude of wave propagation using the graphical arrow element. Typically, however, vector fields are more suited for visualizing direction of propagation.

Vector fields are created using the same displacement data described previously. Each node of the mesh serves as the starting point from which an arrow is drawn representing the displacement vector for a particular point mass. The magnitude of displacement and the direction of displacement are mapped into the length and direction of the arrow, respectively.

The first step in creating vector fields, like in color contouring, involves calculating the minimum and maximum displacement values for all time steps. The second step involves drawing arrow elements at each vertex of a mesh element for a particular time step. The final image is a field of arrow elements, which for each time step vary in lengths and pointing

directions.

Calculating the minimum and maximum displacement values for all time steps is done using the code in Appendix B. As in the color contouring technique, this code calculates M_{\min} and M_{\max} for all nodes and all time steps. Drawing arrow elements at each vertex of a mesh element is accomplished using the code in Appendix E. An arrow element has two coordinates as shown in Fig. 10a. The starting coordinate, or the tail of the arrow, is the location of a particular node denoted by the coordinate (x_j, z_j) . The ending coordinate, or the head of the arrow, is denoted by (x_e, z_e) and is calculated as

$$x_e = x_j + u_{j,t} \quad (25)$$

$$z_e = z_j + w_{j,t} \quad (26)$$

In Fig. 10b, an example of arrow elements drawn at each vertex of a particular mesh element is shown. The code in Appendix E draws arrow elements for each mesh element in the entire mesh, where the final result for the entire mesh at a given time may be similar to the vector field shown in Fig. 11.

Vector fields, unlike color contouring, serve to show direction as well as magnitude. Vector fields are particularly useful for visualizing changes in direction of wave propagation. Moreover, this technique also enables the visualization of surface wave propagation.

Two-Dimensional Mesh Deformation

In this section, a new technique for visualizing wave propagation termed the two-dimensional mesh deformation technique is described. This technique, like the previous techniques, requires the use of the mesh described in the section titled *Configuration of Mesh*. New methods and software were created to implement this technique. All software related to this technique was written in the C Programming Language to run on the Hewlett-Packard 9000/350 personal computer.

Two-dimensional mesh deformation is particularly useful for visualizing wave propagation not only within the bulk of the material but also at the edge of the material. In this technique, as each point mass displaces in the x- and z-directions, the corresponding nodes of the mesh change their positions and a new mesh configuration is rendered. In Figs. 12a and 12b, this change in mesh configuration between two arbitrarily chosen time steps is illustrated.

The creation of the two-dimensional mesh involves drawing each element of the mesh beginning at the bottom most row, starting with the left most element within that row. Each mesh element is created using a polygon element having five vertices where the first vertex and the fifth vertex are the same; therefore, a closed quadrilateral element is created. For a mesh element, the first vertex and fifth vertex are the bottom left node, the second vertex is the bottom right node, the third vertex is the top right node, the fourth vertex is the top left node. For example, in Fig. 9, the first vertex and fifth vertex would be the location of node 21, the second vertex would be the location of node 22, the third vertex would be the location of node 28, and the fourth vertex would be the location of node 27.

For each time step, new nodal positions are calculated based on the displacements of the point masses in the x- and z-directions and a new mesh configuration is created. In Appendix F, the code for rendering each new mesh configuration is shown.

Three-Dimensional Surface Deformation

In this section, a new technique for visualizing wave propagation termed the three-dimensional surface deformation technique is described. This technique, like the previous techniques, requires the use of the mesh described in the section titled *Configuration of Mesh*. The three-dimensional surface deformation technique is a new and unique method of graphically visualizing changes in the magnitude of the displacement vector during wave propagation. New methods and software were created to implement this technique. All software related to this technique was written in the C Programming Language to run on the Hewlett-Packard 9000/350 personal computer.

The mesh used in the previous techniques is now situated in x-y-z space as shown in Fig. 13, and each node of the mesh is defined by a coordinate triplet: (x,y,z). The x and z components, as in the previous techniques, are used to denote the position of the nodes in the x-z plane. In this technique, however, the additional y component is used to plot the changes in the magnitude of displacement of the corresponding node. Therefore, while the x and z components for a node remain fixed, the y component varies from time step to time step. Redrawing the mesh for each time step is a more complicated process than the two-dimensional techniques described in the previous sections since the nodes of the mesh no longer lie in the x-z plane. Therefore, curves are calculated along each set of nodes parallel to the x-axis and then along each set of nodes parallel to the z-axis to create a smooth three-dimensional wire mesh. Each mesh element of this three-dimensional wire mesh is then filled with a color to generate a three-dimensional surface which depicts overall changes in the magnitude of the displacement vector.

Prior to discussing each step involved in the three-dimensional surface deformation technique, the mathematics for calculating three-dimensional curves is described.

Calculating Three-Dimensional Curves

In this technique, it is necessary to calculate three-dimensional curves for each set of nodes parallel to the x- and z-axis. The calculation of a three-dimensional curve for a set of n nodes is equivalent to the general problem of finding a curve to fit a group of n points. Mathematically, this problem involves finding a polynomial of order (n-1) for a set of n points. Clearly, this problem becomes quite cumbersome and difficult to solve as the number of points, n, increases. Therefore, in the three-dimensional surface deformation technique, a practical approach is taken towards solving this problem by decomposing the problem into a simpler problem. More specifically, in this technique, piecewise polynomial curve segments, called *splines*, are calculated for each sub-group of four points, and these curve segments are connected together to produce one continuous three-dimensional curve. The number of splines, N_s , required to create one three-dimensional curve is given as

$$N_s = \frac{(n - 1)}{3} \quad (27)$$

where n is the total number of points through which the three-dimensional curve passes. Since N_s must be an integral value, n is selected to ensure an integral number of splines. The number of points n must be such that the quantity (n-1) is divisible by three. In Fig. 14, for example, the seven points shown in three-dimensional space require two splines to create a three-dimensional curve.

In this technique, polynomials of order three or *cubic* splines are selected as the class of piecewise polynomials for generating the three-dimensional curves. The main reason is that no lower-order representation of curve segments can provide continuity of position and slope at the point where curve segments meet and at the same time ensure that the ends of

the curve segments pass through specified points. Cubic splines, therefore, ensure that the curve segments join smoothly at the connection points of individual curve segments. Furthermore, while polynomials of higher-order can be used, these functions tend to have undesirable oscillations which require more complex methods for generating smooth curves [11].

A cubic spline is mathematically represented by three parametric equations as

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \quad (28)$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \quad (29)$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z \quad (30)$$

where a, b, c and d subscripted by x, y and z are coefficients for eqns. (28), (29) and (30), respectively and where t is the parametric variable with limits of

$$0 \leq t \leq 1 \quad (31)$$

Eqns. (28), (29) and (30) are rewritten in matrix form for greater clarity. In developing the mathematics for converting these equations to matrix form, eqn. (28) is taken as an example and the results are later applied to eqns. (29) and (30).

Eqn. (28) is rewritten as

$$x(t) = (t^3 \quad t^2 \quad t \quad 1) \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} \quad (32)$$

Taking the first derivative of eqn. (32) with respect to t gives

$$\frac{dx(t)}{dt} = (3t^2 \quad 2t \quad 1 \quad 0) \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} \quad (33)$$

Next eqns. (32) and (33) are evaluated at the endpoints of the cubic spline where $t=0$ and $t=1$. Eqn. (32) evaluated at $t=0$ gives

$$x(0) = (0 \quad 0 \quad 0 \quad 1) \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} \quad (34)$$

Evaluating eqn. (32) at $t=1$ gives

$$x(1) = (1 \quad 1 \quad 1 \quad 1) \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} \quad (35)$$

Evaluating eqn. (33) at $t=0$ gives

$$\dot{x}(0) = (0 \quad 0 \quad 1 \quad 0) \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} \quad (36)$$

Evaluating eqn. (33) at $t=1$ gives

$$\dot{x}(1) = (3 \quad 2 \quad 1 \quad 0) \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} \quad (37)$$

Eqns. (34), (35), (36) and (37) can be written in one equation as

$$\begin{pmatrix} x(0) \\ x(1) \\ \dot{x}(0) \\ \dot{x}(1) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} \quad (38)$$

Inverting eqn. (38) gives

$$\begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ \dot{x}(0) \\ \dot{x}(1) \end{pmatrix} \quad (39)$$

Substituting eqn. (39) into eqn. (32) gives

$$x(t) = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ \dot{x}(0) \\ \dot{x}(1) \end{pmatrix} \quad (40)$$

Using eqn. (40), $x(t)$ is evaluated for $t=0, 1/3, 2/3$ and 1 ; the result is written in matrix form as

$$\begin{pmatrix} x(0) \\ x(1/3) \\ x(2/3) \\ x(1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 20/27 & 7/27 & 4/27 & -2/27 \\ 7/27 & 20/27 & 2/27 & -4/27 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ \dot{x}(0) \\ \dot{x}(1) \end{pmatrix} \quad (41)$$

Inverting eqn. (41) gives

$$\begin{pmatrix} x(0) \\ x(1) \\ \dot{x}(0) \\ \dot{x}(1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -11/2 & 9 & -9/2 & 1 \\ -1 & 9/2 & -9 & 11/2 \end{pmatrix} \begin{pmatrix} x(0) \\ x(1/3) \\ x(2/3) \\ x(1) \end{pmatrix} \quad (42)$$

Substituting the elements $x(0)$, $x(1/3)$, $x(2/3)$ and $x(1)$ of the column matrix on the right hand side of eqn. (42) such that

$$x(0) = P_x^1 \quad (43)$$

$$x(1/3) = P_x^2 \quad (44)$$

$$x(2/3) = P_x^3 \quad (45)$$

$$x(1) = P_x^4 \quad (46)$$

where P_x^1, P_x^2, P_x^3 and P_x^4 are the x-coordinates of a set of four points that are used to calculate the spline.

Substituting eqns. (43), (44), (45) and (46) into eqn. (42) gives

$$\begin{pmatrix} x(0) \\ x(1) \\ \dot{x}(0) \\ \dot{x}(1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -11/2 & 9 & -9/2 & 1 \\ -1 & 9/2 & -9 & 11/2 \end{pmatrix} \begin{pmatrix} P_x^1 \\ P_x^2 \\ P_x^3 \\ P_x^4 \end{pmatrix} \quad (47)$$

Substituting eqn. (47) into eqn. (40) gives

$$x(t) = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -11/2 & 9 & -9/2 & 1 \\ -1 & 9/2 & -9 & 11/2 \end{pmatrix} \begin{pmatrix} P_x^1 \\ P_x^2 \\ P_x^3 \\ P_x^4 \end{pmatrix} \quad (48)$$

Eqn. (48) can be rewritten as

$$x(t) = \tilde{T} \tilde{M} \tilde{X} \quad (49)$$

where \bar{T} is

$$\bar{T} = (t^3 \quad t^2 \quad t \quad 1) \quad (50)$$

where \bar{M} is

$$\bar{M} = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -11/2 & 9 & -9/2 & 1 \\ -1 & 9/2 & -9 & 11/2 \end{pmatrix} \quad (51)$$

and is simplified to

$$\bar{M} = \frac{1}{2} \begin{pmatrix} -9 & 27 & -27 & 9 \\ 24 & -45 & 36 & -15 \\ -11 & 18 & -9 & 2 \\ 2 & 0 & 0 & 0 \end{pmatrix} \quad (52)$$

and where \bar{X} is

$$\bar{X} = \begin{pmatrix} P_x^1 \\ P_x^2 \\ P_x^3 \\ P_x^4 \end{pmatrix} \quad (53)$$

Eqn. (49) is the general expression for $x(t)$. Similar eqns. for $y(t)$ and $z(t)$ can be written as

$$y(t) = \bar{T}\bar{M}\bar{Y} \quad (54)$$

$$z(t) = \bar{T}\bar{M}\bar{Z} \quad (55)$$

where \bar{Y} is

$$\vec{Y} = \begin{pmatrix} P_y^1 \\ P_y^2 \\ P_y^3 \\ P_y^4 \end{pmatrix} \quad (56)$$

and where \vec{Z} is

$$\vec{Z} = \begin{pmatrix} P_z^1 \\ P_z^2 \\ P_z^3 \\ P_z^4 \end{pmatrix} \quad (57)$$

Eqns. (49), (54) and (55) together represent the equations of a cubic spline in three-dimensions. It is important to note that the spline passes through each of the four points. A three-dimensional curve created using the cubic spline, therefore, passes through all n points.

A three-dimensional curve, $P(t)$, is assembled using one or more cubic splines. Mathematically, $P(t)$ is expressed as

$$P(t) = \sum_{m=1}^{N_s} S^m(t) \quad (58)$$

where the superscript m denotes the number of the spline, and where $S(t)$ represents one cubic spline and is given as

$$S(t) = \{x(t), y(t), z(t)\} \quad (59)$$

Using eqn. (58), a three-dimensional curve can be calculated for a set of n points or n nodes. The spline used to generate the curve in eqn. (58) is commonly known as a Hermite curve [12].

Steps in Creating the Three-Dimensional Surface

The first step in creating the three-dimensional surface is to select a mesh of proper dimensions. The number of mesh elements E_x and E_z in the x- and z-directions, respectively, is expressed as

$$E_x = n_x - 1 \quad (60)$$

$$E_z = n_z - 1 \quad (61)$$

where n_x and n_z are the number of nodes in the x- and z-directions, respectively.

From the previous discussion on creating three-dimensional curves, eqn. (27) demands that one less than the number of points or nodes selected be divisible by three. In this case, it requires that $(n_x - 1)$ and $(n_z - 1)$ be divisible by three. This requirement along with eqns. (60) and (61) means that a mesh for this technique be selected such that the number of mesh elements along the x- and z-directions be divisible by three.

The second step involves calculating the y components for each node j of the mesh at time step t using the equation

$$Y_{j,t} = 1 - \frac{2(M_{\max} - M_{j,t})}{(M_{\max} - M_{\min})} \quad (62)$$

where $M_{j,t}$ is calculated using eqn. (11), and where $Y_{j,t}$ denotes the y component for node j at time step t . The y component, as evident from eqn. (62), is a normalized value which varies between -1 and 1. The software program shown in Appendix G performs the calculation of $Y_{j,t}$.

The third step involves calculating three-dimensional curves for each set of nodes parallel to the x-axis and for each set of nodes parallel to the z-axis to create a three-dimensional wire mesh. Using eqn. (58), this calculation is performed and the curves for each set of nodes is generated. In Appendix H, the calculation of the three-dimensional curves and the generation of the three-dimensional wire mesh are performed.

Once the three-dimensional wire-mesh is created, the final step in creating the three dimensional surface involves filling each mesh element with a color to create the appearance of a surface. Each mesh element, as in the two-dimensional mesh deformation technique, is represented as a polygon element. Appendix H also contains the code for performing this final step of filling in each polygon element with a color to render the three-dimensional surface.

In Figs. 15a, 15b, 15c and 15d, the overall process of creating a three-dimensional surface from a three-dimensional wire mesh is summarized. In Fig. 15a, a mesh of proper dimension is chosen such that the number of mesh elements along the x- and z-directions is divisible by three. Fig. 15b shows the effect on the mesh following the calculation of y components for each node at a particular time step. In Fig. 15c, curves are generated along each set of nodes parallel to the x- and z-axes. Finally, in Fig. 15d each mesh element is filled to created the appearance of a surface.

Conclusions

Four techniques for visualizing wave propagation have been discussed: color contouring, vector fields, two-dimensional mesh deformation and three-dimensional surface deformation. The techniques of two-dimensional mesh deformation and three-dimensional surface deformation are two new methods for visualizing wave propagation. Specific procedures and software were developed for implementing these techniques. The techniques of color contouring and vector fields, which are traditional visualization techniques, were adapted to handle the particular mesh configuration and data generated by the MSLM. While in the current presentation of the four techniques displacement data have been used as the input for visualization, there is no restriction that the data be only displacement data. The data may be any variable of interest, including velocities, accelerations, stresses or strains, as long as the data to mesh mapping protocol described above is followed.

Combining the features of some of the techniques may yield new techniques for visualizing wave propagation. For example, color contouring combined with vector fields may offer an even richer method for visualizing the direction and magnitude of propagation. In addition, visualizing various variables using different techniques, at the same time, may prove to be useful. For example, visualizing displacement with vector fields and stress using color contouring may prove to be effective. However, the key in such research is to enable the viewer to comprehend and to recognize patterns in complex visual information without overloading the viewer with too many visual cues. Such new techniques will be pursued in subsequent research.

References

- [1] B.H. McCormick, T.A. DeFanti and M.D. Brown, "Visualization in Scientific Computing", *Computer Graphics*, Vol. 21, No. 6, November 1987, pp. vii-viii.
- [2] B.H. McCormick, T.A. DeFanti and M.D. Brown, "Visualization in Scientific Computing", *Computer Graphics*, Vol. 21, No. 6, November 1987, pp. E1-E8.
- [3] S. Ayyadurai, "Formulation of Mass-Spring Lattice Model", Composite Materials and Nondestructive Evaluation Laboratory, MIT, 1990.
- [4] S. Ayyadurai, "Computational Wave Propagation in Isotropic and Anisotropic Media", Composite Materials and Nondestructive Evaluation Laboratory, MIT, 1990.
- [5] Starbase Reference Manual, Hewlett-Packard Corporation, 1980.
- [6] J. Bertin, Semiology of Graphics, The University of Wisconsin Press, 1983, pp. 85-97.
- [7] E.R. Tufte, The Visual Display of Quantitative Information, Graphics Press, 1983, pp. 153-175.
- [8] J. Bertin, Semiology of Graphics, The University of Wisconsin Press, 1983, pp. 346-347.
- [9] D.E. Knuth, The Art of Computer Programming, Vol. 1, Addison-Wesley Publishing Company, 1973, pp. 37-38.
- [10] R.H. Bartels, J.C. Beatty and B.A. Barsky, An Introduction to Splines for Use in Computer Graphics and Geometric Modeling, Morgan Kaufmann Publishers, Inc., 1987, pp. 321-369.
- [11] G.J. Peters, Interactive Computer Graphics Application of the Bi-Cubic Parametric Surface to Engineering Design Problems, AFIPS Press, 1974, pp. 491-511.
- [12] J.D. Foley and A. Van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley Publishing Company, 1982, pp. 505-572.
- [13] J.D. Foley and A. Van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley Publishing Company, 1982, pp. 521-523.
- [14] J.D. Foley and A. Van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley Publishing Company, 1982, pp. 581-582.

Value of Variable		Corresponding Color
60.0	-----	Red
50.0	-----	Purple
40.0	-----	Orange
30.0	-----	Yellow
20.0	-----	Green
10.0	-----	Blue

Fig. 1 Example of color legend.

31 21	32 22	33 23	34 24	35 25	36
25 16	26 17	27 18	28 19	29 20	30
19 11	20 12	21 13	22 14	23 15	24
13 6	14 7	15 8	16 9	17 10	18
7 1	8 2	9 3	10 4	11 5	12
1	2	3	4	5	6

Fig. 2 Element and node numbering convention.

Node j	x-displacement	z-displacement
1	u1	w1
2	u2	w2
3	u3	w3
4	u4	w4
5	u5	w5
6	u6	w6
7	u7	w7
8	u8	w8
9	u9	w9
10	u10	w10
11	u11	w11
12	u12	w12
13	u13	w13
14	u14	w14
15	u15	w15
16	u16	w16
17	u17	w17
18	u18	w18
19	u19	w19
20	u20	w20
21	u21	w21
22	u22	w22
23	u23	w23
24	u24	w24
25	u25	w25

Fig. 3 Form of data from MSLM for each node of mesh.

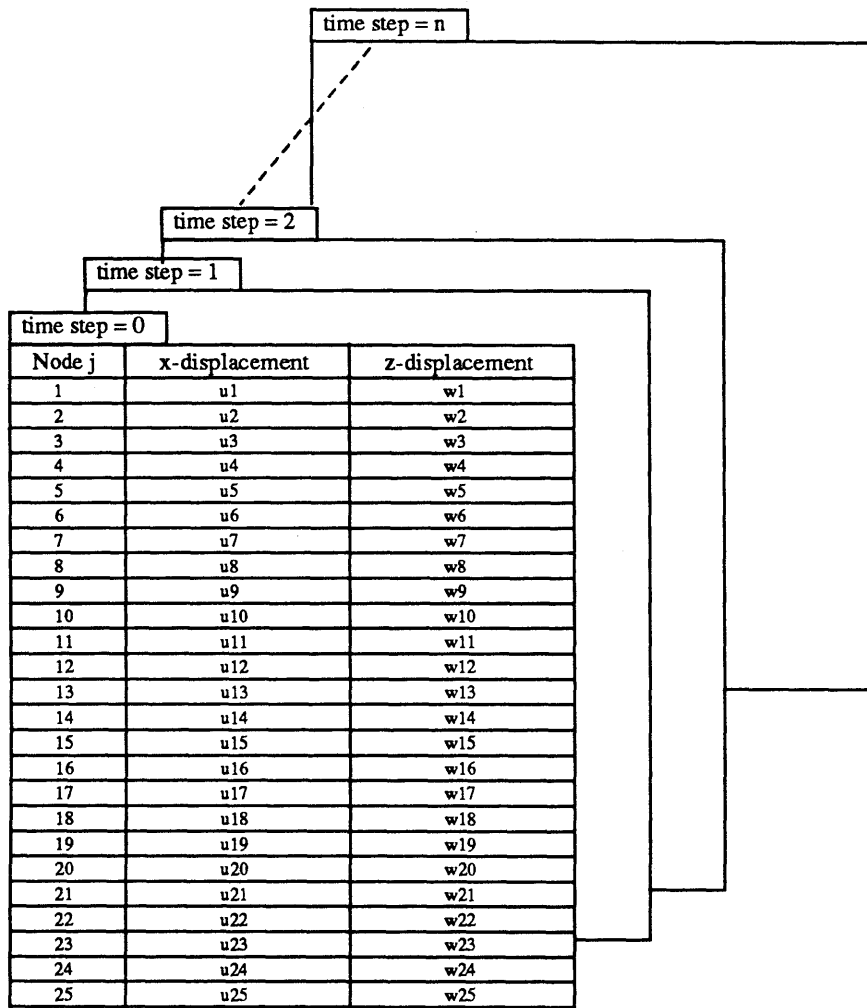


Fig. 4 Representation of displacement data for n time steps.

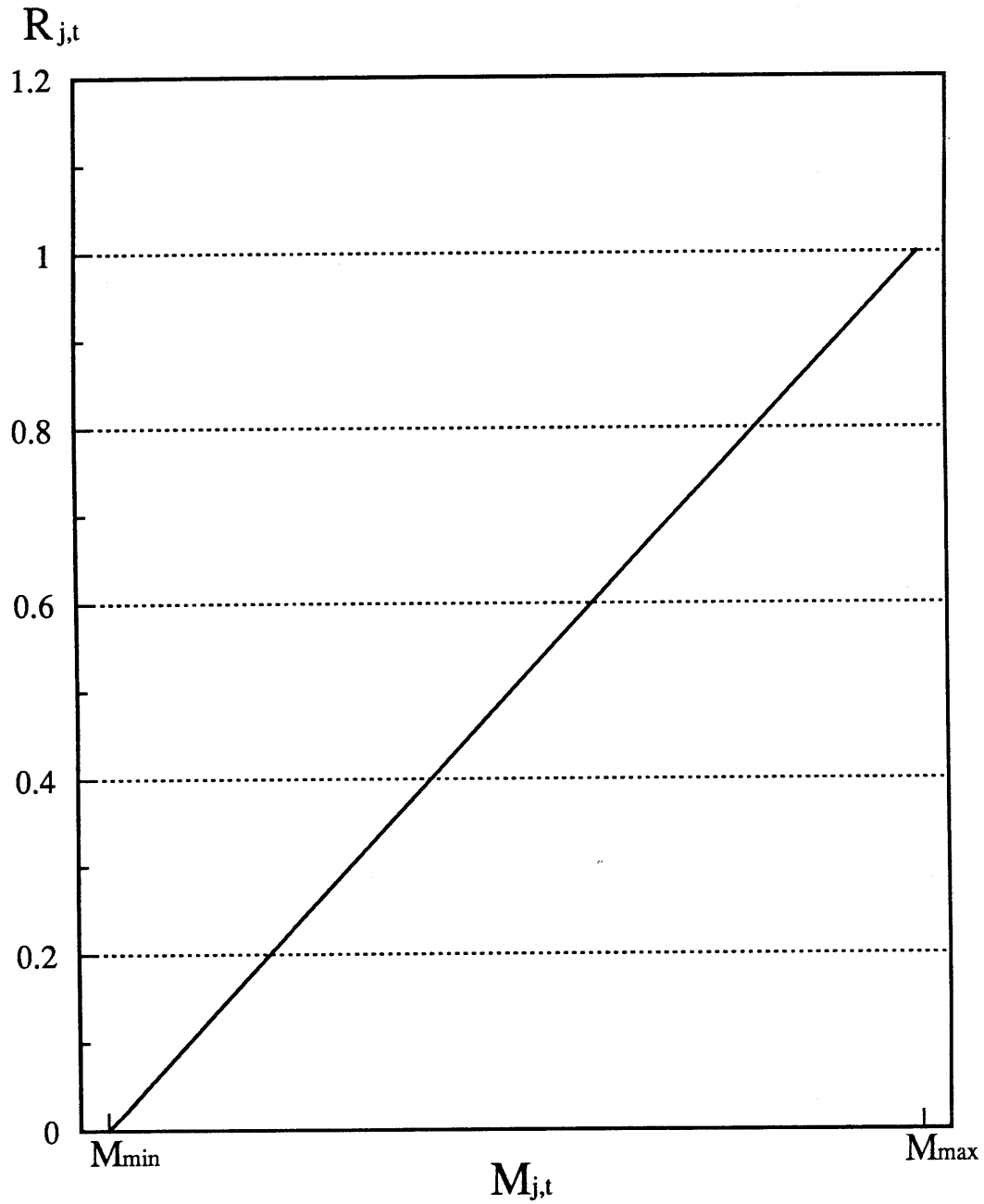


Fig. 5 Red component variation with magnitude of displacement.

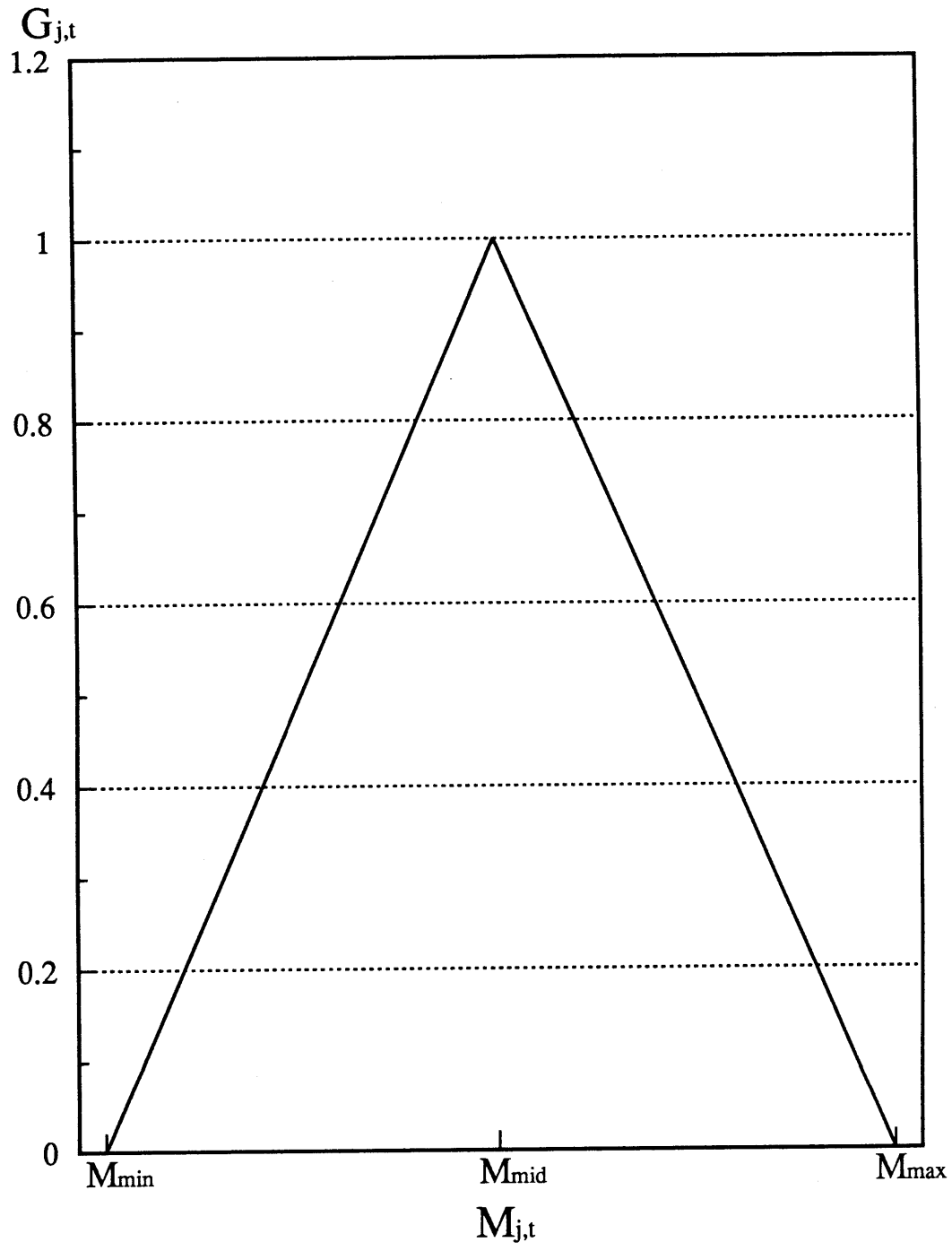


Fig. 6 Green component variation with magnitude of displacement.

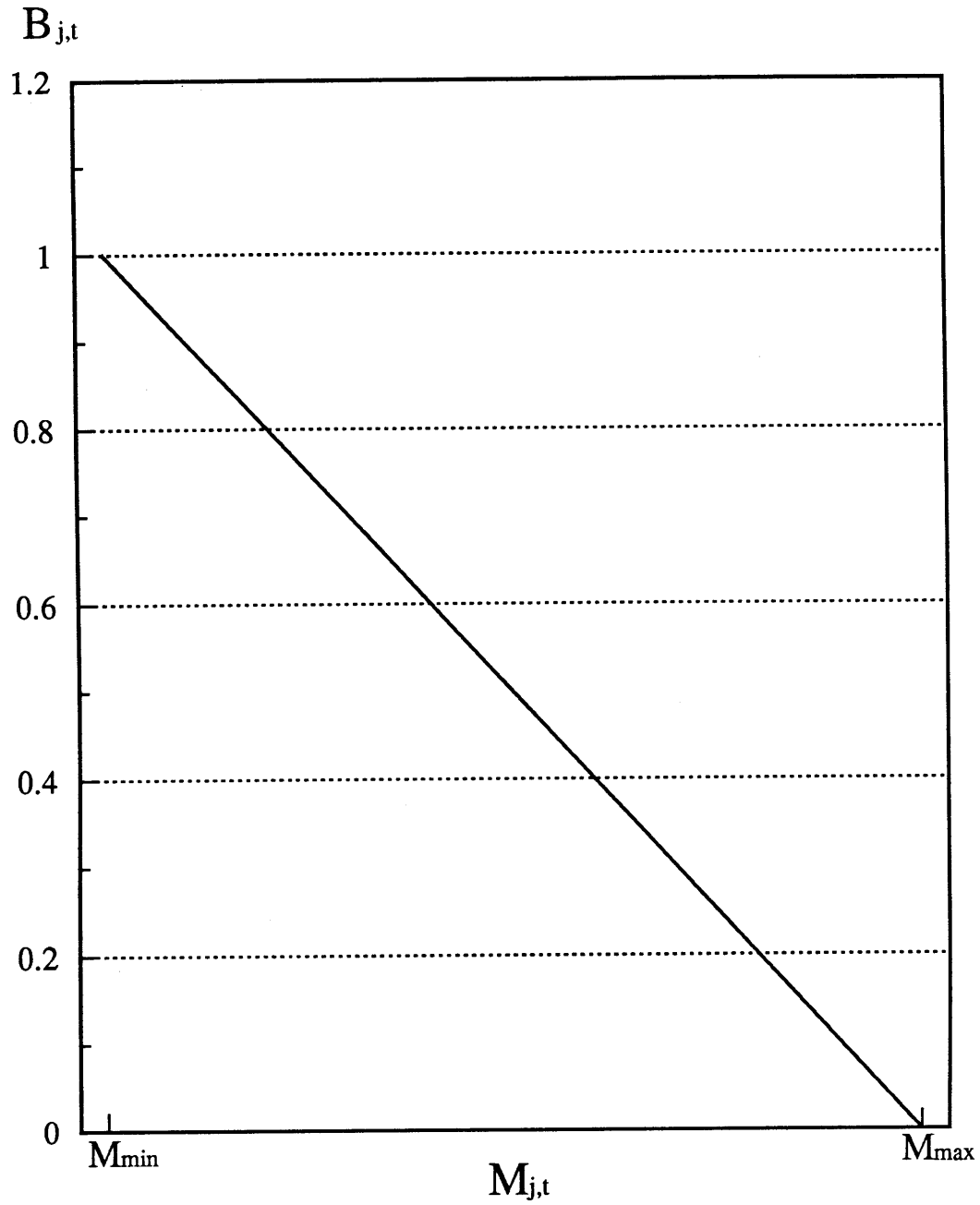


Fig. 7 Blue component variation with magnitude of displacement.

Node j	Rj	Gj	Bj
1	R1	G1	B1
2	R2	G2	B2
3	R3	G3	B3
4	R4	G4	B4
5	R5	G5	B5
6	R6	G6	B6
7	R7	G7	B7
8	R8	G8	B8
9	R9	G9	B9
10	R10	G10	B10
11	R11	G11	B11
12	R12	G12	B12
13	R13	G13	B13
14	R14	G14	B14
15	R15	G15	B15
16	R16	G16	B16
17	R17	G17	B17
18	R18	G18	B18
19	R19	G19	B19
20	R20	G20	B20
21	R21	G21	B21
22	R22	G22	B22
23	R23	G23	B23
24	R24	G24	B24
25	R25	G25	B25

Fig. 8 RGB assignment convention for each node at specified time.

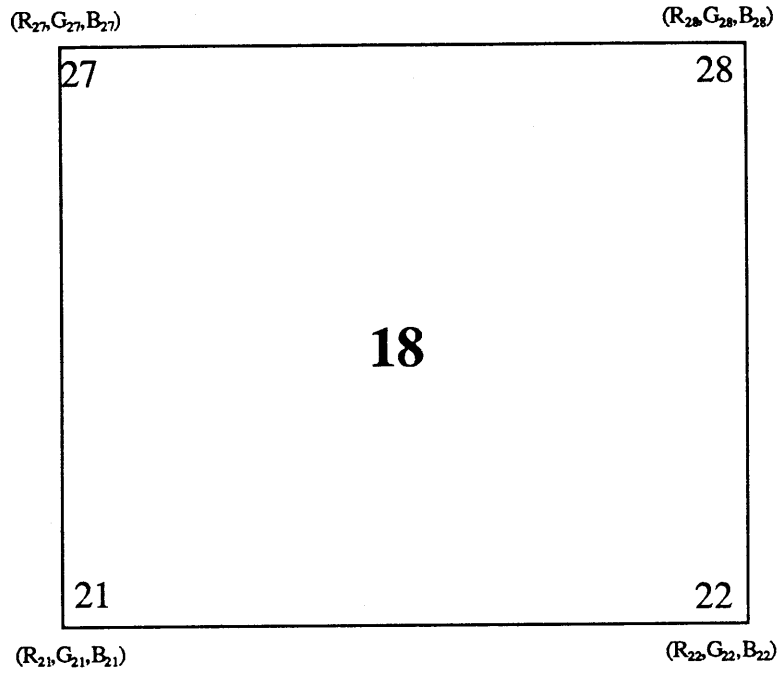


Fig. 9 RGB value corresponding to nodes of mesh element.

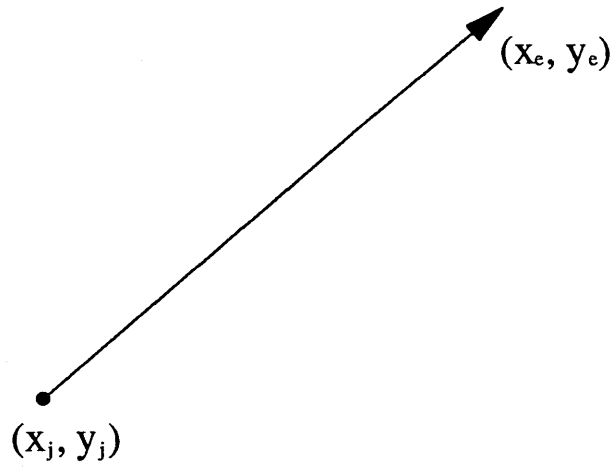


Fig. 10a Arrow element for creating vector fields.

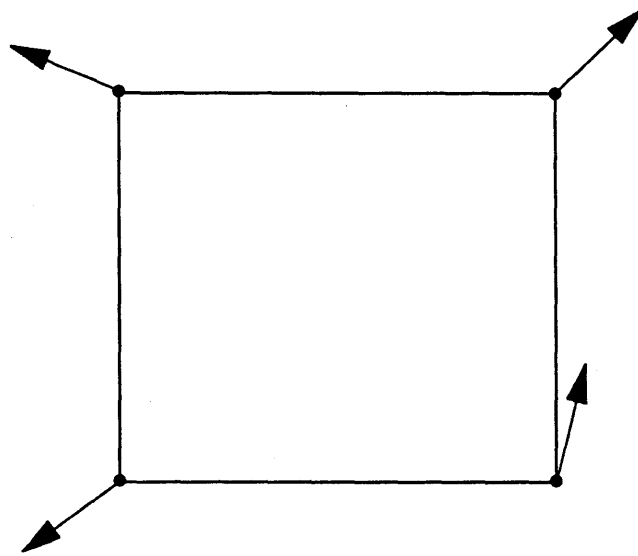


Fig. 10b Arrow elements drawn at each node of mesh element.

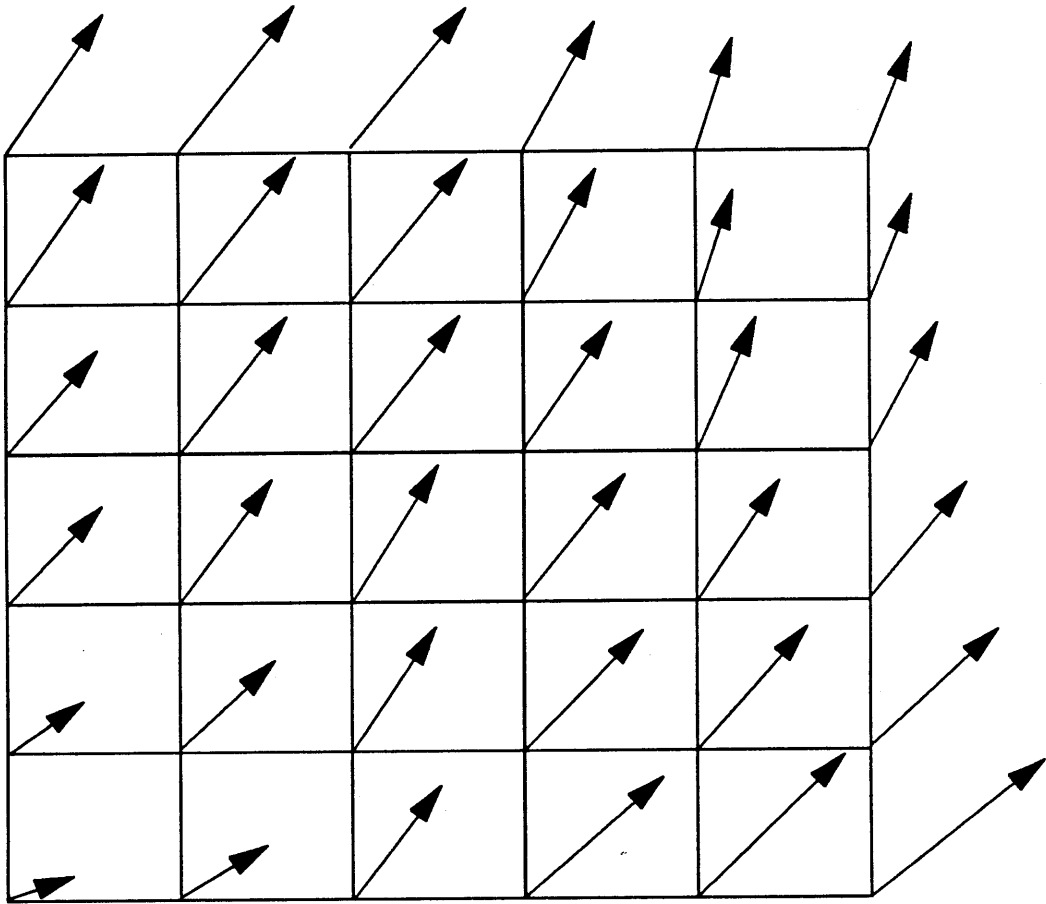


Fig. 11 Arrow elements drawn for mesh.

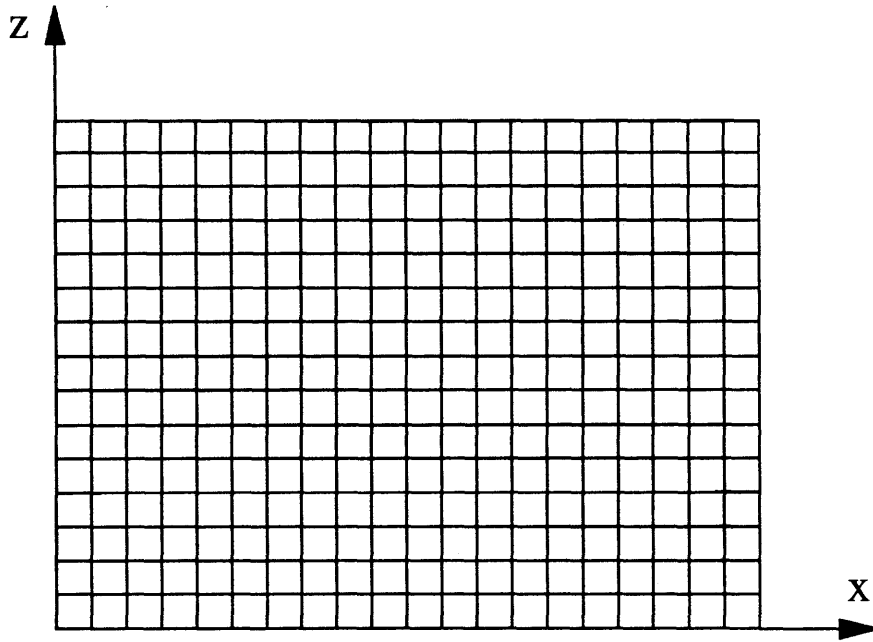


Fig. 12a Undeformed rectangular mesh.

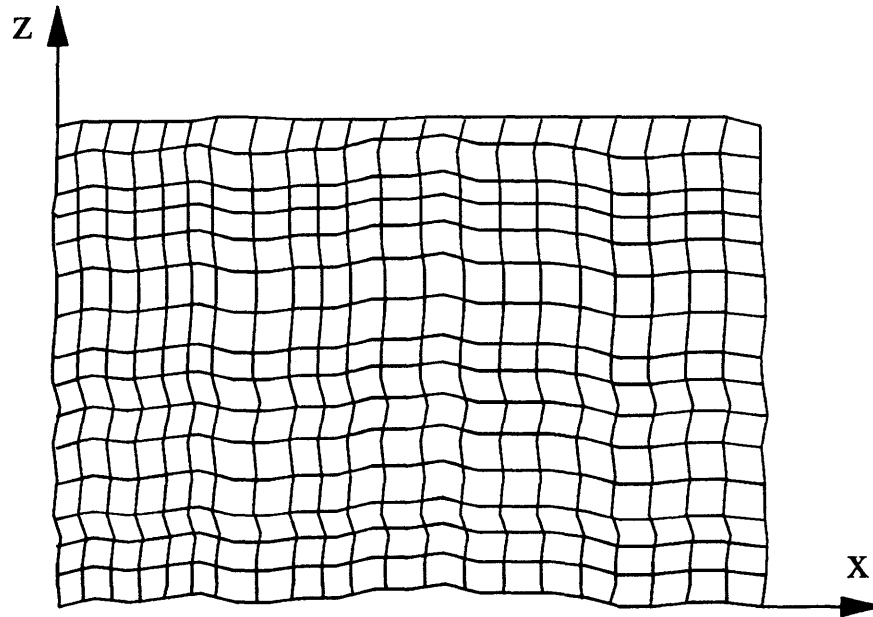


Fig. 12b Deformed rectangular mesh.

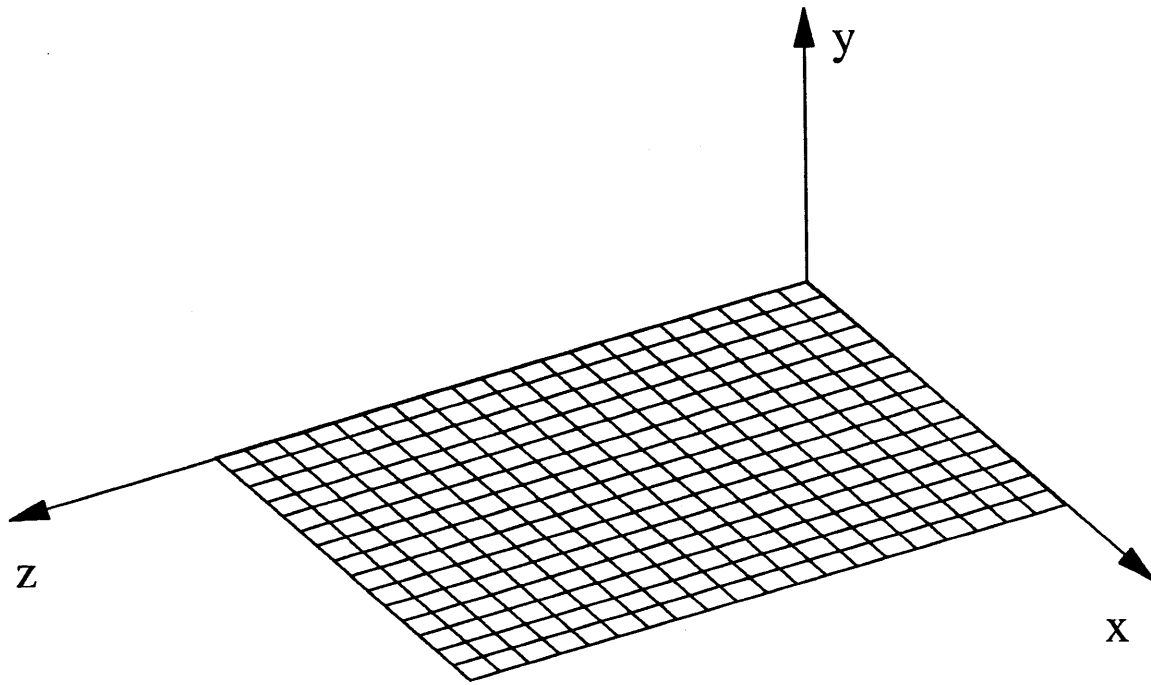


Fig. 13 Mesh situated in x-y-z space.

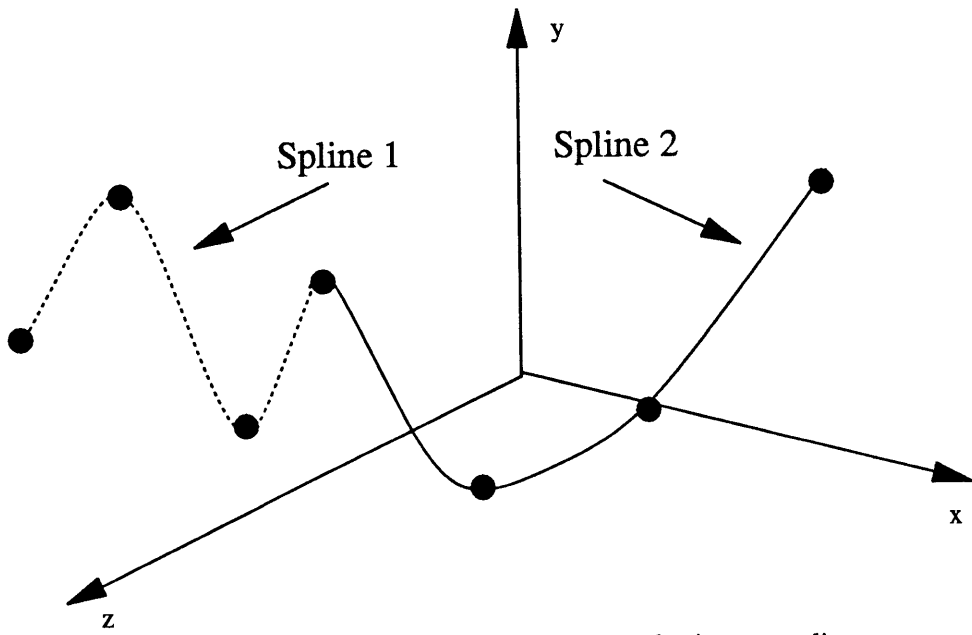


Fig. 14 Three-dimensional curve created using two splines.

Fig. 15a A 3 by 6 mesh.

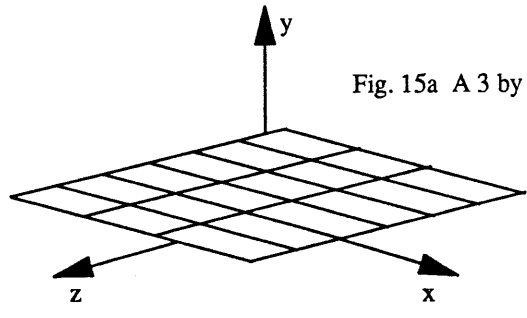


Fig. 15b Mesh with y component displaced.

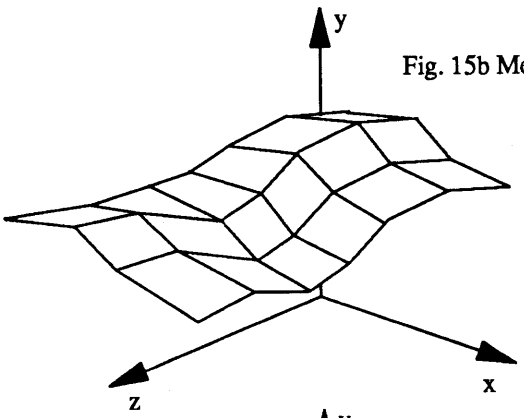


Fig. 15c Mesh created using three-dimensional curves.

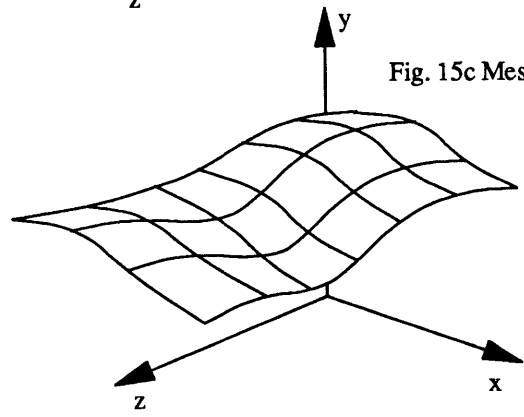
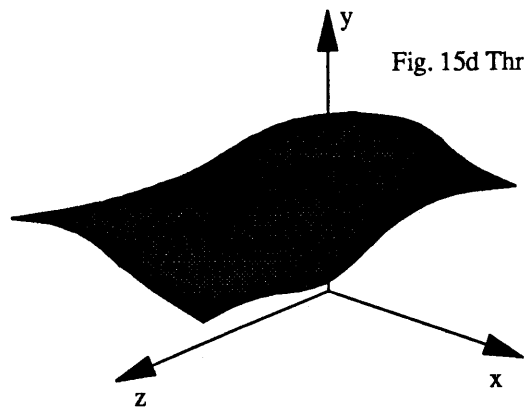


Fig. 15d Three-dimensional surface.



Appendix A: Program for Generating Mesh.

```

/* mesh.c */

/*****
*
*           MESH
*
* This program generates a mesh which is R_MNODES by
* R_NNODES. The mesh is defined by its connectivity. This
* connectivity data is stored in a file with extension MSH.
*
*****/

/* Include Files */
#include <math.h>
#include <stdio.h>
#include </lisp/users/vashiva/newwave.h>

main()
{

    int num_steps;           /* number of time steps */
    int step;                /* time step index */
    double elaps_tim;       /* amount of elapsed time */
    int i,j;                 /* loop indices */
    int n;                   /* type of boundary node */
    double delta_t;         /* time step */
    double period;          /* period of impulse */
    char mshnam[80];        /* mesh file name */
    FILE *fopen(), *fp;     /* file attributes */
    int num_nodes;          /* number nodes in mesh */
    int num_elems;          /* number elements mesh */
    int xnode_min, xnode_max; /* loop indices */
    int xelem_max, ynode_max; /* loop indices */
    int ylevel;             /* loop indices */
    double bottom;          /* loop indices */
    double xstep;           /* mesh spacing in x (m) */
    double ystep;           /* mesh spacing in y (m) */
    int melems;             /* elements in x */
    int nelems;             /* elements in y */
    double x[R_MNODES*R_NNODES]; /* location of nodes in x */
    double y[R_MNODES*R_NNODES]; /* location of nodes in y */
    int N[(R_MNODES-1)*(R_NNODES-
1)]; /* connectivity
    [NODES_ELEM]

```

```

/* Initialize the number of nodes and elements in the mesh */
num_nodes = R_MNODES*R_NNODES;
num_elems = melems*nelems;

```

```

/* Evaluate spacing of mesh in x and y direction */
xstep = delta_t/(sqrt(DENSITY*FAC*1.0E-10));
ystep = delta_t/(sqrt(DENSITY*FAC*1.0E-10));

```

```

/* calculate the x positions of each node */
xnode_min = 0;
xnode_max = melems;
for (i=0; i<num_nodes; i++){
  x[i] = xstep*((double)(i-xnode_min));
  if (i >= xnode_max){
    xnode_min = xnode_max + 1;
    xnode_max = xnode_min + melems;
  }
}

```

```

/* calculate the y positions of each node */
ynode_max = melems;
ylevel = 0;
bottom = 0.0;
for (i=0; i<num_nodes; i++){
  y[i] = bottom + (double)(ystep*ylevel);
  if (i>=ynode_max){
    ynode_max = i + melems + 1;
    ylevel = ylevel + 1;
  }
}

```

```

/* Evaluate the overall connectivity of the mesh */
ylevel = 0;
xelem_max = melems - 1;
for (i = 0; i<num_elems; i++){
  N[i][0] = i + ylevel + 1;
  N[i][1] = N[i][0] + 1;
  N[i][2] = N[i][1] + melems + 1;
  N[i][3] = N[i][2] - 1;

  if (i >= xelem_max){
    xelem_max = xelem_max + melems;
    ylevel = ylevel + 1;
  }
}

```

```

/* Open file for storing nodal x and y positions */
fp = fopen(mshnam, "w");

/* Write Header into MSH file */
fprintf (fp, "%d %d %d\n", num_steps, num_elems, num_nodes);

/* Write x and y nodal positions */
for (i=0; i<num_nodes; i++){
    fprintf (fp, "%1.10e %1.10e\n", x[i], y[i]);
}

/* Write connectivity information */
for (i=0; i<num_elems; i++){
    fprintf(fp, "%d %d %d %d\n", N[i][0], N[i][1], N[i][2], N[i][3]);
}

/* Close the mesh file */
fclose(fp);

}/* main */
/*=====*/

```

Appendix B: Program for Calculating Max. and Min. Displacements.

```
/* maxmin.c */

/*****
 *
 *           MAXMIN
 *
 *   This program evaluates the minimum and maximum values
 * across all time steps from the data stored in a file with
 * file extension ANM.
 *
 *****/

/* Include files */
#include <math.h>
#include <stdio.h>
#include <fcntl.h>
#include </lisp/users/vashiva/anim.h>
#include </lisp/users/vashiva/newwave.h>

/* Defines */
#define MAXFNAM 80           /* max. file name      */
#define BIG 1.79e+308      /* largest value      */
#define SMALL 1.79e-308   /* smallest value     */

MaxMin(m_min, m_max)
double *m_min, *m_max;
{
  char valnam[MAXFNAM];    /* ANM file name      */
  FILE *fopen(), *fp;     /* file pointers      */
  int i, t;               /* loop indices       */
  int num_steps;         /* number of steps    */
  int num_elems, num_nodes; /* elements and nodes */
  double dum1, dum2;     /* dummy variables    */

  /* Open the ANM File */
  sprintf (valnam, "%s.anm", ANIM_NAM);
  if ((fp = fopen(valnam, "r")) == NULL){
```

```
printf("Error: Cannot open file\n");
exit(-1);
}
fclose(fp);
```

```
/* Calculate the Minimum and maximum Values */
/* Under Consideration */
fp = fopen(valnam, "r");
*m_max = SMALL;
*m_min = BIG;
for (t=0; t<num_steps; t++){
  for (i=0; i<num_nodes; i++){
    fscanf(fp, "%lf %lf", &dum2, &dum1); /* dum2, critical */
    if (dum2 >= *m_max)
      *m_max = dum2;
    else if (dum2 <= *m_min)
      *m_min = dum2;
  }
}
```

```
/* Close the ANM File */
fclose (fp);
```

```
}/* MaxMin */
```

```
/*=====*/
=====*/
```

Appendix C: Program for Assigning RGB Value to a Node.

```
/* rgb.c */

/*****
*           RGB
*
*   This program assigns RGB values to a particular node
*   of the mesh.
*
*****/

#include <math.h>
#include <stdio.h>
#include <fcntl.h>
#include </lisp/users/vashiva/anim.h>
#include </lisp/users/vashiva/newwave.h>

#define MAXFNAM 80           /* max. file name      */
#define MAX_INDEX NODES_ELEM*5 /* array size of clist */
#define BIG 1.79e+308       /* largest value      */
#define SMALL 1.79e-308    /* smallest value     */

Rgb (m_node, r, g, b, F, f_min, f_max)
int m_node;
float *r, *g, *b;
double F[];
double f_min, f_max;
{
    double f_mid;

    /* Find cutoff value */
    f_mid = (f_max - f_min)/2.0 + f_min;

    /* Map Colors to Values */
    *r = (F[m_node] - f_min)/(f_max - f_min);
```

```
if (F[m_node] <= temp)
  *g = 2*(F[m_node] - f_min)/(f_max - f_min);
else
  *g = 2*(F[m_node] - f_max)/(f_min - f_max);

*b = (F[m_node] - f_max)/(f_min - f_max);
```

```
}/* Rgb */
```

```
/*=====
```

```
====*/
```

Appendix D: Program for Smooth Shading Mesh Element.

```
/* smooth.c */

/*****
*
*           SMOOTH
*
*   This program smooth shades a mesh element based on the
*   RGB values at the four vertices.
*
*****/

#include <math.h>
#include <stdio.h>
#include <fcntl.h>
#include </lisp/users/vashiva/anim.h>
#include </lisp/users/vashiva/newwave.h>

/*=====
***/
/* .doc smooth_shade
*/

smooth_shade(dev, elem, F, N, f_min, f_max, delet, x, y)
int dev;
int elem;
double F[];
int N[][NODES_ELEM];
double f_min;
double f_max;
int delet;
double x[], y[];

{
  int i;
  int e_node, m_node;
  float r, g, b;
  float clist[MAX_INDEX + 5]; /* contains 5 values per node: */
                             /* x,y,r,g,b */
  e_node = 0;

  for (i=0; i< MAX_INDEX; i = i + 5){
```



```

m_node = N[elem][e_node] - 1;

clist[0+i] = x[m_node];
clist[1+i] = y[m_node];

Rgb(m_node, &r, &g, &b, F, f_min, f_max);

clist[2+i] = r;
clist[3+i] = g;
clist[4+i] = b;

e_node = e_node + 1;

}/* for i */

/* close the polygon */
for (i=0; i<5; i++){
    clist[20+i] = clist[0+i];
}

/* draw the polygon */
polygon2d(dev, clist, 5, FALSE);
make_picture_current(dev);

}/* smooth_shade */
/*=====*/

```

Appendix E: Program for Rendering Vector at Each Node.

```
/* wavevect.c */

/*****
*
*           WAVEVECT
*
*
* This program generates vector fields using the data in
* in the MSH file and ANM file. A vector is denoted by a
* magnitude and a direction. Lines are drawn of a normal-
* length in a particular direction to emulate a vector.
* The vectors are drawn at each node of the mesh.
*
*****/

#include <math.h>
#include <stdio.h>
#include <fcntl.h>
#include </lisp/users/vashiva/anim.h>
#include </lisp/users/vashiva/newwave.h>

#define MAXFNAM 80 /* max. file name */
#define MAX_INDEX NODES_ELEM*5 /* array size of clist */
#define BIG 1.79e+308 /* largest value */
#define SMALL 1.79e-308 /* smallest value */
#define VECT_LEN 200.0 /* length of display vector */
#define VECT_R 0.0 /* red color of vector */
#define VECT_G 1.0 /* green color of vector */
#define VECT_B 0.0 /* blue color of vector */

main()
{
    int dev; /* output graphics device */
    char mshnam[MAXFNAM]; /* MSH file name */
    char valnam[MAXFNAM]; /* ANM file name */
    FILE *fopen(), *fp; /* file pointers */
    int i, k; /* loop indices */
    int num_steps; /* number of steps */
    int num_elems, num_nodes; /* elements and nodes */
    double x[R_MNODES*R_NNODES]; /* x position of node */
}
```

```

double y[R_MNODES*R_NNODES]; /* y position of node */
double xmin, xmax;          /* loop indices */
double ymin, ymax;          /* loop indices */
int elem;                    /* loop indices */
double f_min, f_max;        /* max/min of values */
double dum1, dum2;          /* dummy variables */
double vect_mag;            /* vector magnitude */
double u,w;                  /* displacement in x,y */
double factor;              /* length factor */
int Buffer = 0;              /* for animating */
int N[ (R_MNODES-1)*        /* mesh connectivity */
      (R_NNODES-1)]
      [NODES_ELEM];

```

```

/* Initialize Graphics Device */
dev = gopen(REN, OUTDEV, HP98721, INIT|THREE_D|MODEL_XFORM);
double_buffer(dev, TRUE, 12);
shade_mode (dev, CMAP_FULL|INIT, TRUE);
vertex_format (dev, 3, 3, TRUE, FALSE, COUNTER_CLOCKWISE);
vdc_extent (dev, 0.0, 1023.0, 0.0, 1279.0, 0.0, 1.0);
clip_indicator (dev, CLIP_TO_VDC);

```

```

/* Get file name of MSH and ANM files */
printf("Enter MSH file\n");
scanf ("%s", mshnam);
printf("Enter ANM file\n");
scanf ("%s", valnam);

```

```

/* Open MSH Data File */
if ((fp = fopen(mshnam, "r")) == NULL){
    printf("Error: Cannot open file\n");
    gclose(dev);
    exit(-1);
}

```

```

/* Read x, y Position Data. Calculate xmin, xmax, ymin, ymax */
xmin = ymin = BIG;
xmax = ymax = SMALL;
for (i=0; i<num_nodes; i++){
    fscanf (fp, "%lf %lf\n", &x[i], &y[i]);
}

```

```

if (x[i] > xmax)
    xmax = x[i];
if (x[i] < xmin)
    xmin = x[i];
if (y[i] > ymax)
    ymax = y[i];
if (y[i] < ymin)
    ymin = y[i];
}

/* Read Connectivity Data into Local Array, N */
for (i=0; i<num_elems; i++){
    fscanf(fp, "%d %d %d %d", &N[i][0],&N[i][1],&N[i][2],&N[i][3]);
}

/* Close the MSH File */
fclose(fp);

/* Check Degenerate Mesh */
if ((xmin == xmax) || (ymin == ymax)){
    printf ("Error: This is a one-dimensional mesh\n");
    gclose(dev);
    exit(-1);
}

/* Shape the x,y Values to Fit Screen */
shape (num_nodes, xmin, xmax, ymin, ymax, x, y);

/* Open the ANM File */
if ((fp = fopen(valnam, "r")) == NULL){
    printf("Error: Cannot open file\n");
    gclose(dev);
    exit(-1);
}
fclose(fp);

/* Calculate the Minimum and maximum Value of the Magnitude */
fp = fopen(valnam, "r");
f_max = SMALL;
f_min = BIG;
for (i=0; i<num_steps; i++){
    for (k=0; k<num_nodes; k++){

```

```

fscanf(fp, "%lf %lf", &dum1, &dum2);
vect_mag = (dum1*dum1) + (dum2*dum2); /* vector magnitude */
if (vect_mag >= f_max)
    f_max = vect_mag;
else if (vect_mag <= f_min)
    f_min = vect_mag;
}
}

```

```

/* Close the ANM File */
fclose (fp);

```

```

/* Set Max and Min of the Magnitude */
f_min = sqrt(f_min);
f_max = sqrt(f_max);

```

```

/* Report the Minimum and Maximum */
printf("fmin=%1.10e, fmax=%1.10e\n", f_min, f_max);

```

```

/* Initialize the factor */
factor = VECT_LEN/f_max;

```

```

/* Re-open the ANM file */
fp = fopen(valnam, "r");

```

```

/* Animate the Vector's */
for (i=0; i<num_steps; i++){
    dbuffer_switch(dev, Buffer = !Buffer);

```

```

for (k=0; k<num_nodes; k++){
    fscanf(fp, "%lf %lf", &dum1, &dum2);
    u = factor*dum1;
    w = factor*dum2;

```

```

/* Set the Vector's Line Color */
line_color(dev, 1.0, 1.0, 1.0);
move2d(dev, (float)x[k], (float)y[k]);
draw2d(dev, (float)(x[k]+u), (float)(y[k]+w));
line_color(dev, 1.0, 0.0, 0.0);
draw2d(dev, (float)(x[k]+u), (float)(y[k]+w));

```

```

    make_picture_current(dev);
  }
}

/* Close the ANM File */
fclose (fp);

}/* main */
/*=====
=*/

/*=====*
/
/* .doc shape static
re-shapes the x,y positions of the nodes for the current screen
*/
static shape(num_nodes, xmin, xmax, ymin, ymax, x, y)
int num_nodes;
double xmin, xmax, ymin, ymax;
double x[], y[];
{
    double rs;
    double xfact, yfact;
    int i;

    /* shape the x y location's with shape function */
    rs = (ymax - ymin)/(xmax - xmin);

    if (rs <= (YLEN/XLEN)){
        rs = rs*XLEN/YLEN;
        xfact = (((1.0 - XPCT)/2.0)*XPIX);
        yfact = (((YPCT/2.0) + 0.5)*YPIX) - ((1-rs)*YPCT*YPIX/2.0);

        for (i=0; i<num_nodes; i++){
            x[i] = (xfact + (XPCT*XPIX*((x[i] - xmin)/(xmax - xmin))));
            y[i] = (yfact - (rs*YPCT*YPIX*((y[i] - ymin)/(ymax - ymin))));
            if (x[i] < 0.0)
                x[i] = (int) (x[i] - 0.5);
            else
                x[i] = (int) (x[i] + 0.5);
        }
    }
}

```

```

    if (y[i] < 0.0)
        y[i] = (int) (y[i] - 0.5);
    else
        y[i] = (int) (y[i] + 0.5);
}
}/* if */
else {
    rs = (1/rs)*(YLEN/XLEN);
    xfact = (((1.0 - XPCT)/2.0)*XPIX) + ((XPCT*(1-rs)/2.0)*XPIX);
    yfact = (((YPCT/2.0) + 0.5)*YPIX);

    for (i=0; i<num_nodes; i++){
        x[i] = (xfact + (XPCT*rs*XPIX*((x[i] - xmin)/(xmax - xmin))));
        y[i] = (yfact - (YPCT*YPIX*(y[i]-ymin)/(ymax-ymin)));

        if (x[i] < 0.0)
            x[i] = (int) (x[i] - 0.5);
        else
            x[i] = (int) (x[i] + 0.5);

        if (y[i] < 0.0)
            y[i] = (int) (y[i] - 0.5);
        else
            y[i] = (int) (y[i] + 0.5);
    }
}/* else */

}/* end shape */
/*=====
*/

```

Appendix F: Program for Two-Dimensional Mesh Deformation.

```
/* wavemesh.c */

/*****
*
*           WAVEMESH
*
* This program deforms the mesh based on the x-y displace-
* ment.
*****/

/* INCLUDES */

#include <math.h>
#include <stdio.h>
#include "fcntl.h"
#include </lisp/users/vashiva/anim.h>
#include </lisp/users/vashiva/newwave.h>

/* DEFINES */
#define MAXFNAM 80
#define BIG 1.79e+308
#define SMALL 1.79e-308
#define BACK_R 0.4
#define BACK_G 0.4
#define BACK_B 0.4
#define MESH_SPAC 60.0

main()
{
    int dev; /* output graphics device */
    char mshnam[MAXFNAM]; /* MSH file name */
    char valnam[MAXFNAM]; /* ANM file name */
    FILE *fopen(), *fp; /* file pointers */
    int i,j, k; /* loop indices */
    int num_steps; /* number of steps */
    int num_elems, num_nodes; /* elements and nodes */
    double x[R_MNODES*R_NNODES]; /* x position of node */
    double y[R_MNODES*R_NNODES]; /* y position of node */
    double u[R_MNODES*R_NNODES]; /* displacement in x */
    double w[R_MNODES*R_NNODES]; /* displacement in y */
    double xmin, xmax; /* loop indices */
    double ymin, ymax; /* loop indices */
    double f_min, f_max; /* max/min of values */
}
```



```

double vect_mag;          /* vector magnitude      */
double factor;           /* length factor        */
double dum1, dum2;       /* dummy variables      */
int Buffer = 0;           /* for animating        */
int N[ (R_MNODES-1)*     /* mesh connectivity    */
      (R_NNODES-1)]
      [NODES_ELEM];
char titl1[80];
char titl2[80];
char cur_tim[80];

/* Initialize Output Device */
dev = gopen(REN, OUTDEV, HP98721, INIT|THREE_D|MODEL_XFORM);
double_buffer(dev, TRUE, 12);
vertex_format (dev, 0, 0, TRUE, FALSE, COUNTER_CLOCKWISE);
vdc_extent (dev, 0.0, 1023.0, 0.0, 1279.0, 0.0, 1.0);
clip_indicator (dev, CLIP_TO_VDC);

/* Set Text Characteristics */
printf("Enter Heading:\n");
scanf("%s", titl1);
sprintf(titl2, "Displacement - Mesh Deformation");
text_alignment (dev, TA_CENTER, TA_TOP, 0.0, 0.0);
text_font_index(dev, 6);
designate_character_set(dev, "usascii", 0);
character_height(dev, 40.0), character_width(dev, 30.0);
text_color (dev, 1.0, 0.0, 0.0);
text_orientation2d(dev,0.0,-1.0,1.0,0.0);

/* Get file name of MSH and ANM files */
printf("Enter MSH file\n");
scanf ("%s", mshnam);
printf("Enter ANM file\n");
scanf ("%s", valnam);

/* Open MSH File */
if ((fp = fopen(mshnam, "r")) == NULL){
  printf("Error: Cannot open file\n");
  exit(-1);
}

```

```

/* Report Number of Animation Steps */
fscanf (fp, "%d %d %d", &num_steps, &num_elems, &num_nodes);
printf("There will be %d animation steps, elems=%d, nodes=%d\n",
      num_steps, num_elems, num_nodes);

/* Read x, y position data. calculate xmin, xmax, ymin, ymax */
xmin = ymin = BIG;
xmax = ymax = SMALL;
for (i=0; i<num_nodes; i++){
  fscanf (fp, "%lf %lf", &x[i], &y[i]);
  if (x[i] > xmax)
    xmax = x[i];
  if (x[i] < xmin)
    xmin = x[i];
  if (y[i] > ymax)
    ymax = y[i];
  if (y[i] < ymin)
    ymin = y[i];
}

/* Read Connectivity Data */
for (i=0; i<num_elems; i++){
  fscanf(fp, "%d %d %d %d", &N[i][0],&N[i][1],&N[i][2],&N[i][3]);
}

/* Close MSH File */
fclose (fp);

/* Check Degenerate Mesh */
if ((xmin == xmax) || (ymin == ymax)){
  printf ("Error: This is a one-dimensional mesh\n");
  exit(-1);
}

/* Open ANM File */
if ((fp = fopen(valnam, "r")) == NULL){
  printf("Error: Cannot open file\n");
  gclose(dev);
  exit(-1);
}
fclose(fp);

```

```

/* Open ANM File */
fp = fopen(valnam, "r");

/* Calculate Minimum and maximum */
f_max = SMALL;
f_min = BIG;
for (i=0; i<num_steps; i++){
  for (k=0; k<num_nodes; k++){
    fscanf(fp, "%lf %lf", &dum1, &dum2);
    vect_mag = (dum1*dum1) + (dum2*dum2); /* vector magnitude */
    if (vect_mag >= f_max)
      f_max = vect_mag;
    else if (vect_mag <= f_min)
      f_min = vect_mag;
  }
}

/* Close ANM File */
fclose (fp);

/* Set Max and Min of the Magnitude */
f_min = sqrt(f_min);
f_max = sqrt(f_max);

/* Shape the Nodal Points to Screen Coordinates */
shape (num_nodes, xmin+f_min, xmax+f_max, ymin+f_min, ymax+f_max, x, y);

factor = MESH_SPAC/f_max;

/* Open ANM File */
fp = fopen(valnam, "r");

/* Animate Deformation of Mesh */
for (j=0; j<num_steps; j++){
  dbuffer_switch(dev, Buffer = !Buffer);

  /* Changing text */
  interior_style(dev, INT_SOLID, TRUE);
  fill_color (dev, 1.0, 0.0, 0.0);
  text2d(dev, 640.0, 75.0, titl1, VDC_TEXT, 0);
}

```

```

text2d(dev, 640.0, 150.0, titl2, VDC_TEXT, 0);
text2d(dev, 640.0, 830.0, "Time", VDC_TEXT,0);
sprintf(cur_tim, "%1.2e", j*DELTA_T);
line_color(dev, 1.0, 1.0, 1.0);
move2d(dev, 527.0, 875.0);
draw2d(dev, 750.0, 875.0);
draw2d(dev, 750.0, 930.0);
draw2d(dev, 527.0, 930.0);
draw2d(dev, 527.0, 875.0);
text2d(dev, 640.0, 880.0, cur_tim, VDC_TEXT, 0);
text2d(dev, 640.0, 930.0, "(secs)", VDC_TEXT,0);

for (k=0;k<num_nodes; k++) {
    fscanf(fp, "%lf %lf", &u[k], &w[k]);
    u[k] = u[k]*factor;
    w[k] = w[k]*factor;
}

fill_color (dev, 0.0, 0.4, 1.0);
for (i=0; i<num_elems; i++){
    disp_2d_elem(dev, i, N, x, y, u, w);
}
}

/* Close ANM File */
fclose (fp);

}/* main */
/*=====*/

/*=====*/
/* .doc disp_2d_elem static
*/
static disp_2d_elem (dev, elem, N, x, y, u, w)
int dev, elem, N[][NODES_ELEM];
double x[], y[];
double u[], w[];
{
    float clist[NODES_ELEM*2 + 2];
    int i;
    int num_vert;
    int e_node = 0;
    int m_node;

```

```

/* Deform the Mesh */
for (i=0; i<8; i = i + 2){
  m_node = N[elem][e_node] - 1; /* subtract 1 to bring to 0 base */
  clist[0+i] = (float) x[m_node] + u[m_node];
  clist[1+i] = (float) y[m_node] + w[m_node];
  e_node = e_node + 1;
}/* for i */

/* Close the Polygon */
clist[8] = clist[0];
clist[9] = clist[1];

polygon2d(dev, clist, 5, FALSE);
make_picture_current(dev);

}/* end draw_2d_elem */
/*=====*/

```

Appendix G: Program for Calculating Y Component for a Node.

```

/* zcoord.c */

/*****
/*
/*          YCOORD
/*   This program calculates the y component for the nodes
/* of a mesh based on the displacement value of the node.
*****/

/* Include Files */

#include <starbase.c.h>
#include <stdio.h>
#include <math.h>
#include </lisp/users/vashiva/newwave.h>

/*=====
/
YCoord(CtrlPts)
float CtrlPts[R_NNODES][R_MNODES][3]; /* Control Points */
{

/* Global Variables */
int u,v;                /* Knot Vectors */
double f_min, f_max;    /* min/max disp. values */
float x_step, y_step    /* virtual mesh spacing */
char valnam[MAXFNAM];  /* ANM file name */
int num_steps;         /* number of steps */
int num_elems, num_nodes; /* elements and nodes */
FILE *fopen(), *fp;    /* file pointers */
double dum1, dum2;     /* dummy variables */
int i, k;              /* loop indices */

/* Initialize the x, y coordinates of the mesh */
x_step = 1.0/((float) R_NNODES - 1.0);
z_step = 1.0/((float) R_MNODES - 1.0);
for (v=0; v<R_NNODES; v++){
  for (u=0; u<R_MNODES; u++){
    CtrlPts[v][u][0] = z_step*((float) u);
    CtrlPts[v][u][2] = x_step*((float) v);
  }
}
}

```

```

    CtrlPts[v][u][1] = 0.0;
  }
}

/* Calculate the minimum and maximum values of displacement */
MaxMin(&f_min, &f_max)

/* Open the file containing the displacement values */
fp = fopen(valnam, "r");

/* Calculate the z-coordinate for each node. */
for (v=0; v<R_NNODES; v++){
  for (u=0; u<R_MNODES; u++){
    fscanf(fp, "%lf %lf", &dum2, &dum1); /* dum2, critical */
    CtrlPts[v][u][1] = 1.0 - (float) (2.0*(f_max-dum2)/(f_max-f_min));
  }
}

/* Close the ANM File */
fclose (fp);

}/* end YCoord */

/*=====
*/

```

Appendix H: Program for Generating Cubic Spline Surface.

```

/* surface.c */

/*****
/*
/*          SURFACE
/* This program contains the graphics routines to render the
/* the three-dimensional surface.
/*
*****/

#include <starbase.c.h>
#include <stdio.h>
#include <math.h>
#include </lisp/users/vashiva/newwave.h>

#define AMP 0.5
#define deg      *M_PI/180
#define ZbufferOn TRUE
#define CullOn   TRUE
#define Gray(Lvl) Lvl, Lvl, Lvl
#define Red      1.0, 0.0, 0.0
#define Black    0.0, 0.0, 0.0
#define White    Gray(1.0)
#define On       TRUE
#define SpecularOn TRUE
#define BIG      1.79e+308 /* largest value */
#define SMALL    1.79e-308 /* smallest value */
#define MAXFNAM  80      /* max. file name */

/* Global Variables */
int fildes; /* Graphic Output Device */
camera_arg Camera; /* Camera */
int ans,splin_typ; /* Spline Type */
float CtrlPts[R_NNODES][R_MNO- /* Control Points */
DES][3]; /* Knot Vectors */
int u,v; /* Angle */
float Theta; /* Dbuffer Switch */
int Buffer= 0; /* MSH file name */
char mshnam[MAXFNAM]; /* ANM file name */
char valnam[MAXFNAM]; /* number of steps */

```



```

int num_steps;                /* elements and nodes */
int num_elems, num_nodes;    /* min/max disp. values */
double f_min, f_max;         /* file pointers */
FILE *fopen(), *fp;         /* dummy variables */
double dum1, dum2;          /* loop indices */
int i, k;
float refy;

```

```

/*=====
*/

```

```

main()
{

```

```

float x_step;
float z_step;

```

```

printf ("Display mode? 1 = Wire Mesh, 2 = Surface\n");
scanf ("%d", &ans);
if (ans == 2){
    printf("What spline type?\n");
    printf("1 = BI-LINEAR\n");
    printf("2 = BI-QUADRATIC\n");
    printf("3 = BI-CUBIC\n");
    scanf ("%d", &splin_typ);
}

```

```

printf("\33h\33J"), fflush(stdout);
fildes = gopen("/dev/crtren", OUTDEV, "hp98721",
    INIT | THREE_D | MODEL_XFORM);

```

```

/* Get file name of MSH and ANM files */
printf("Enter MSH file\n");
scanf ("%s", mshnam);
printf("Enter ANM file\n");
scanf ("%s", valnam);

```

```

/* Open MSH Data File */
if ((fp = fopen(mshnam, "r")) == NULL){
    printf("Error: Cannot open file\n");
    gclose(fildes);
    exit(-1);
}

```

```

/* Report Number of Animation Steps */
fscanf(fp, "%d %d %d", &num_steps, &num_elems, &num_nodes);
printf("There will be %d animation steps, elems=%d, nodes=%d\n",
      num_steps, num_elems, num_nodes);

fclose(fp);

/* Open the ANM File */
if ((fp = fopen(valnam, "r")) == NULL){
    printf("Error: Cannot open file\n");
    fclose(fildes);
    exit(-1);
}
fclose(fp);

/* Set the Camera */
Camera.refx = 0.0;

refy = Camera.refy = 1.0 - (float) (2.0*f_max/(f_max-f_min));
Camera.refz = 0.0;
Camera.upx = 0.0, Camera.upy = 1.0, Camera.upz = 0.0;
Camera.front = -5.0, Camera.back = 5.0;
Camera.projection = CAM_PERSPECTIVE;
Camera.field_of_view = 25;

/* set resolution */
curve_resolution(fildes, STEP_SIZE, 0.1, 0.1, 0.1, 0.1);

/* set for animating */
double_buffer(fildes, TRUE, 12);

/* shade and hidden on */
shade_mode(fildes, CMAP_FULL|INIT, On);
hidden_surface(fildes, ZbufferOn, CullOn);

/* set fill color */
fill_color(fildes, 0.7, 0.3, 1.0);

/* set lighting */
light_ambient(fildes, 1.0, 1.0, 0.0);

```

```

light_source(fildes, 1, POSITIONAL, Gray(0.7), 10.0, 10.0, -10.0);
light_source(fildes, 2, POSITIONAL, Gray(0.4), 0.0, 5.0, -3.0);
light_switch(fildes, 7);

/* create splines */
surface_model(fildes, SpecularOn, 50, 1.0, 0.7, 0.7);
/* DrawSurface() */
vertex_format(fildes, 3, 3, FALSE, FALSE, COUNTER_CLOCKWISE);

DrawSurface();
gclose(fildes);

}/* end main */
/*=====
*/

/*=====
*/
/* .doc DrawSurface static
*/
static DrawSurface()
{

float x_step;
float z_step;

Theta = 0.0;
/* Re-read the data and Normalize all the data between -1 and 1 */
fp = fopen(valnam, "r");
for (i=0; i<num_steps; i++){

Camera.camx = 4.0;
Camera.camy = 2.0;
Camera.camz = 5.0;

view_camera(fildes, &Camera);
dbuffer_switch(fildes, Buffer = !Buffer);
zbuffer_switch(fildes, 1);

line_color(fildes, 0.0, 0.3, 1.0);
move3d(fildes, 0.0, -1.0, 0.0);/* y axis */
draw3d(fildes, 0.0, 1.0, 0.0);
move3d(fildes, 0.0, refy, -1.0);/* z-axis */

```

```

draw3d(fildes, 0.0, refy, 1.0);
move3d(fildes, -1.0, refy, 0.0);/* x-axis */
draw3d(fildes, 1.0, refy, 0.0);

/* Calculate the Z-Coordinate for each control point
ZCoord(CtrlPts);

/* animate */
if (ans == 1){
  for (v= 0; v< R_NNODES; v++){
    move3d(fildes, CtrlPts[v][0][0], CtrlPts[v][0][1], CtrlPts[v][0][2]);
    for (u=0; u< R_MNODES; u++){
      draw3d(fildes, CtrlPts[v][u][0], CtrlPts[v][u][1], CtrlPts[v][u][2]);
    }
  }

  for (u= 0; u < R_MNODES; u++){
    move3d(fildes, CtrlPts[0][u][0], CtrlPts[0][u][1], CtrlPts[0][u][2]);
    for (v=1; v<R_NNODES; v++)
      draw3d(fildes, CtrlPts[v][u][0], CtrlPts[v][u][1], CtrlPts[v][u][2]);
  }
}
else {
  line_color_index(fildes, 0.0, 0.3, 1.0);
  if (splin_typ == 1){
    spline_surface(fildes, CtrlPts, R_MNODES, R_NNODES, LINEAR, LINEAR,
NONRATIONAL);
  }
  else if (splin_typ == 2){
    spline_surface(fildes, CtrlPts, R_MNODES, R_NNODES, QUADRATIC, QUA-
DRATIC, NONRATIONAL);
  }
  else if (splin_typ == 3){
    spline_surface(fildes, CtrlPts, R_MNODES, R_NNODES, CUBIC, CUBIC, NON-
RATIONAL);
  }
}

}/* for num_steps */

/* Close the ANM File */
fclose (fp);

}/* end DrawSurface */

/*=====*/

```



Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
Ph: 617.253.2800
Email: docs@mit.edu
<http://libraries.mit.edu/docs>

DISCLAIMER

MISSING PAGE(S)

123

Abstract

Video animations are produced to visualize wave propagation in zinc, uni-directional reinforced graphite epoxy and beryl. The production of these animations required assembling special computer and video hardware and writing new computer software for automating and directing the script. The animations serve to exemplify the power of using the techniques of computer graphics and animation with methods from computational mechanics. The video animations are produced using computer graphics techniques developed for visualizing wave propagation along with results generated from a computational wave propagation model termed the mass-spring lattice model (MSLM).

Introduction

Large scale computer simulations combined with visualization techniques are giving researchers a third means of investigating scientific phenomena. Traditionally, theoretical and experimental methods were used. However, today computer simulations are making it possible to investigate natural events that could not be evaluated by physical experimentation. The video animations that are presented in this discussion are one example of the power of using mathematical modeling techniques with computer graphics and animation techniques for visualizing wave propagation in elastic solids. The video animations were created from computer solutions generated by the mass-spring lattice model (MSLM) [1]. This document consists of this introductory section and an attached video tape which contains computer graphics animations of wave propagation in three materials using four different computer graphics techniques.

The twelve video animations on the attached video tape can be divided into three sets. The first set of four animations depicts wave propagation in the transversely isotropic medium zinc; the second set of four animations depicts wave propagation in uni-directional reinforced graphite epoxy which is also transversely isotropic; and, the third set of animations depicts wave propagation in the isotropic medium beryl. Each set of animations demonstrates the use of four different computer graphics techniques which were especially developed for visualizing wave propagation [2].

The animations were created using data output by the MSLM. The MSLM models the media as a two-dimensional rectangular framework of point masses and springs. The rectangular framework is situated in the x-z plane since the plane of transverse isotropy for zinc and graphite is taken to be the x-z plane as illustrated in Fig. 1a. For beryl, the x-z

plane is also used as the region of visualization. Therefore, the plane of visualization for all three materials is the x-z plane as shown in Fig. 1b.

The input to the MSLM is a sinusoidal pulse of frequency ν defined as

$$\nu = 5 \times 10^6 \text{ Hz} \quad (1)$$

The 5 MHz sinusoidal pulse is injected at a point along one edge of the medium as illustrated in Figs. 2a and 2b. The pulse duration or period is denoted by τ which is defined as

$$\tau = \frac{1}{\nu} = 2 \times 10^{-6} \quad (2)$$

For each animation sequence, the MSLM models the event occurring over a time duration of three pulse periods or 2×10^{-6} seconds. Since each animation sequence lasts for 15 seconds, the actual event has been slowed down by a factor of 2.5×10^6 .

For each material four different techniques are used to visualize the data generated by the MSLM. First color contouring is used, then vector fields, then two-dimensional mesh deformation and finally three-dimensional surface deformation. Each of these techniques offers a different and unique approach to visualizing wave propagation in zinc, graphite and beryl.

Video

Refer to attached video tape.

References

- [1] S. Ayyadurai, "Formulation of Mass-Spring Lattice Model", Composite Materials and Nondestructive Evaluation Laboratory, MIT, 1990.
- [2] S. Ayyadurai, "Computer Graphics Techniques for Visualizing Wave Propagation", Composite Materials and Nondestructive Evaluation Laboratory, MIT, 1990.

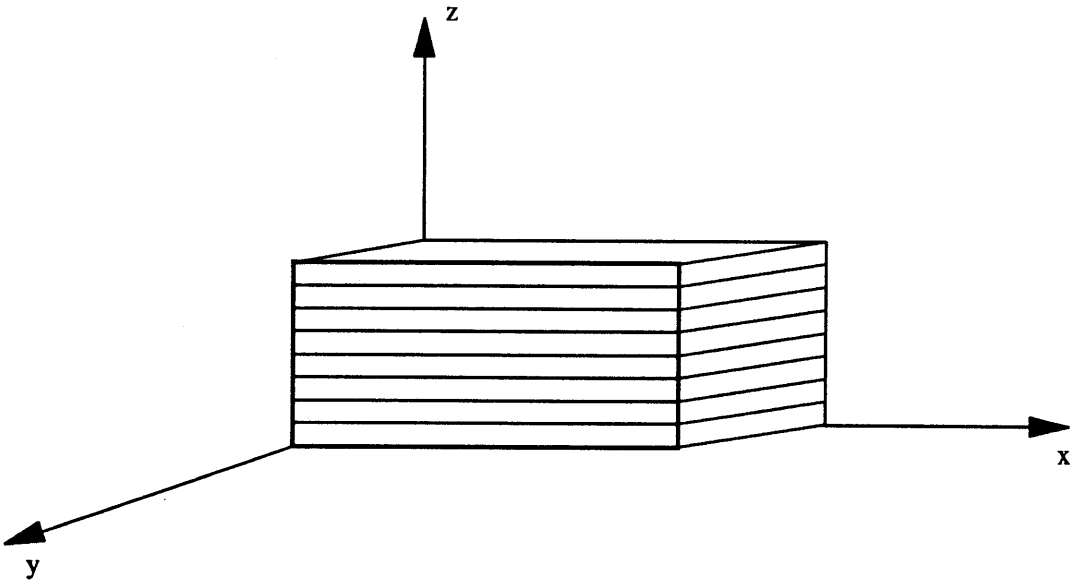


Fig. 1a Reference frame for transversely isotropic media.

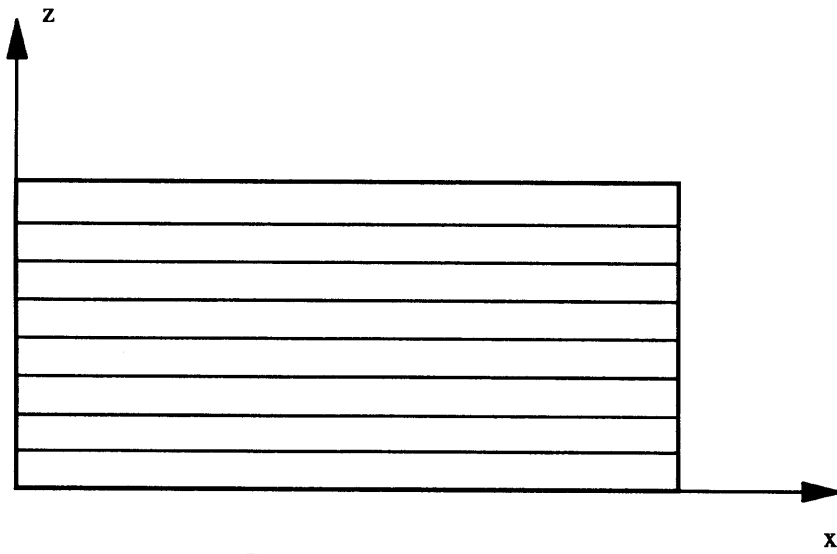


Fig. 1b Region of visualization.

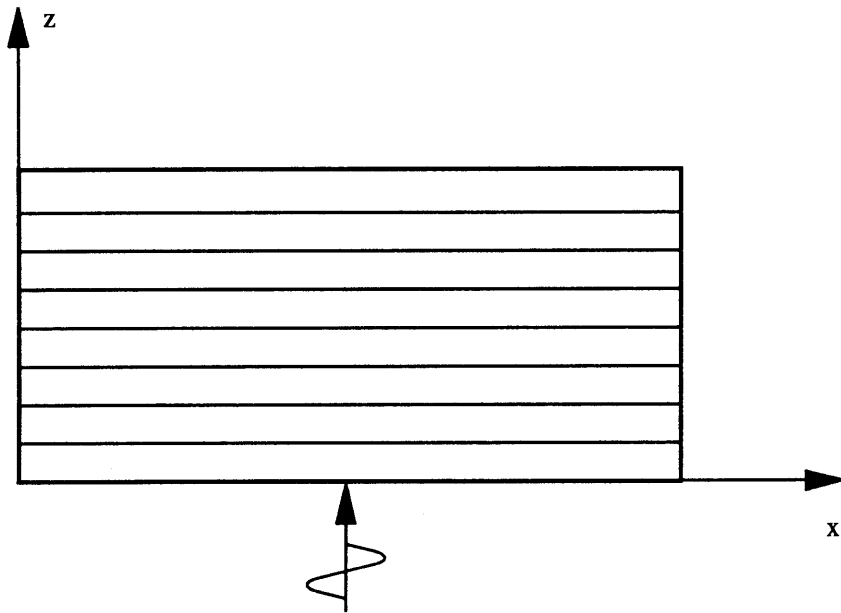


Fig. 2a Sinusoidal pulse injected at center of edge of material.

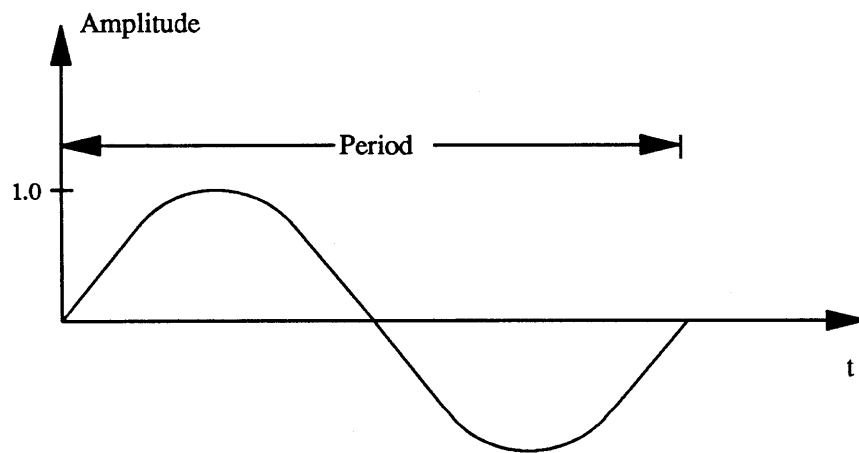


Fig. 2b Sinusoidal pulse.