AD-767 010

3-D SEISMIC CODE FOR ILLIAC-IV

Gerald A. Frazier, et al

Systems, Science and Software

Prepared for:

Defense Nuclear Agency
Advanced Research Projects Agency

9 February 1973

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

AD 767010

DNA 3071Z
FEBRUARY 1973
SSS-R-73-1506

SYSTEMS, SCIENCE AND SOFTWARE

P.O. BOX 1620, LA JOLLA, CALIFORNIA 92037. TELEPHONE (714) 453-0060

3-D SEISMIC CODE FOR ILLIAC IV

INTERIM REPORT

G. A. Frazier
J. H. Alexander
C. M. Petersen

Prepared for

Advanced Research Projects Agency
Arlington, Virginia 22209

Supervised by

Director, Defense Nuclear Agency
Washington, D. C. 20305

SYSTEMS, SCIENCE AND SOFTWARE
La Jolla, California 92037

Contract No. DNA 001-72-C-0154

DDC
RECEIVED
OCT 1 1973
B

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

Approved for Public Release; Distribution Unlimited

118

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Systems, Science and Software
P. O. Box 1620, La Jolla, California 92037

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

3-D SEISMIC CODE FOR ILLIAC IV

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Interim Report

5. AUTHOR(S) *(First name, middle initial, last name)*

Gerald A. Frazier, James H. Alexander, Christian M. Petersen

6. REPORT DATE

February 9, 1973

7a. TOTAL NO. OF PAGES

113

7b. NO. OF REFS

11

8a. CONTRACT OR GRANT NO.

DNA 001-72-C-0154

b. PROJECT NO. ARPA: Z

c. Task and Subtask: M102

d. Work Unit: 01

9a. ORIGINATOR'S REPORT NUMBER(S)

SSS-R-73-1506

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

DNA 3071Z

10. DISTRIBUTION STATEMENT

Approved for public release: distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Advanced Research Projects Agency
and Defense Nuclear Agency

13. ABSTRACT

Finite element (FE) and finite difference (FD) spatial differencing schemes for linearly elastic materials are compared on a common basis. Conditions are established under which the two methods are identical. Features commonly associated with FD methods are combined with a FE spatial treatment to obtain an explicit time stepping algorithm suitable for processing 3-D linearly elastic wave calculations on the ILLIAC IV system. Based on projected timing estimates, the ILLIAC code should be capable of processing wave calculations for a 10^4-node grid at the rate of 0.4 sec per time step. The algorithm has provisions for irregular 3-D geometries, artificial damping for suppressing spurious numerical oscillations, and a nonreflecting boundary condition.

Test calculations are presented to compare FE and FD codes at S^3. Comparisons are made for two problems: A spherically symmetric explosion with an exponentially decaying step pressure and Lamb's problem – an impulse loading applied to the free surface of a homogeneous half space. Plots are presented to show the 2-D displacement, velocity, and kinetic energy fields at various stages of the FE calculations for Lamb's problem. As predicted from theoretical comparisons, no significant advantages in either scheme became apparent for the problems tested.

Test calculations are presented for exploring certain features of the 3-D time stepping algorithm both on S^3's Univac 1108 and on the ILLIAC simulator at UCSD. The algorithm has proven to be very fast on the Univac, time stepping at the rate of 0.5 sec per time step for a particular 404-node 3-D grid. Sample 3-D problems are presented to demonstrate the use of artificial damping, to illustrate the effectiveness of the nonreflecting boundary condition, and to test the ability of the numerical scheme for propagating a step velocity (particle velocity) through 160 elements without excessive spreading at the wave front.

What appears to be a highly efficient algorithm has been developed for multiplying a very large sparse matrix times a core-contained column vector on the ILLIAC IV. The algorithm is independent of the location of elements in the sparse matrix. However, a tedious process is required to arrange the nonzero elements of the matrix on the disk storage. The algorithm is particularly effective for repetitive multiplies involving the same sparse matrix and, therefore, should prove valuable for iterative and time stepping calculations.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
ILLIAC IV						
Seismic						
Numerical Wave Propagation						
3-D Finite Element						
3-D Finite Difference						
Artificial Damping						
Nonreflecting Boundaries						
Lamb's Problem						
3-D Grid Generation						
Sparse Matrix Algebra						

DNA 3071Z
FEBRUARY 1973
SSS-R-73-1506

SYSTEMS, SCIENCE AND SOFTWARE

P.O. BOX 1620, LA JOLLA, CALIFORNIA 92037, TELEPHONE (714) 453-0060

3-D SEISMIC CODE FOR ILLIAC IV

INTERIM REPORT

This work was sponsored by the Defense Nuclear Agency
under ARPA Subtask ZM102-01

G. A. Frazier
J. H. Alexander
C. M. Petersen

Prepared for

Advanced Research Projects Agency
Arlington, Virginia 22209

Supervised by

Director, Defense Nuclear Agency
Washington, D. C. 20305

DDC
RECEIVED
OCT 1 1973

Contract No. DNA 001-72-C-0154
Program Code Number 2F10, ARPA Order Number 2102

Approved for Public Release; Distribution Unlimited

ABSTRACT

Finite element (FE) and finite difference (FD) spatial differencing schemes for linearly elastic materials are compared on a common basis. Conditions are established under which the two methods are identical. Features commonly associated with FD methods are combined with a FE spatial treatment to obtain an explicit time stepping algorithm suitable for processing 3-D linearly elastic wave calculations on the ILLIAC IV system. Based on projected timing estimates, the ILLIAC code should be capable of processing wave calculations for a 10^4-node grid at the rate of 0.4 sec per time step. The algorithm has provisions for irregular 3-D geometries, artificial damping for suppressing spurious numerical oscillations, and a nonreflecting boundary condition.

Test calculations are presented to compare FE and FD codes at S^3. Comparisons are made for two problems: A spherically symmetric explosion with an exponentially decaying step pressure and Lamb's problem – an impulse loading applied to the free surface of a homogeneous half space. Plots are presented to show the 2-D displacement, velocity, and kinetic energy fields at various stages of the FE calculations for Lamb's problem. As predicted from theoretical comparisons, no significant advantages in either scheme became apparent for the problems tested.

Test calculations are presented for exploring certain features of the 3-D time stepping algorithm both on S^3's Univac 1108 and on the ILLIAC simulator at UCSD. The algorithm has proven to be very fast on the Univac, time stepping at the rate of 0.5 sec per time step for a particular 404-node 3-D grid. Sample 3-D problems are presented to demonstrate the use of artificial damping, to illustrate the effectiveness of the nonreflecting boundary condition, and to test the

ability of the numerical scheme for propagating a step velocity (particle velocity) through 160 elements without excessive spreading at the wave front.

What appears to be a highly efficient algorithm has been developed for multiplying a very large sparse matrix times a core-contained column vector on the ILLIAC IV. The algorithm is independent of the location of elements in the sparse matrix. However, a tedious process is required to arrange the nonzero elements of the matrix on the disk storage. The algorithm is particularly effective for repetitive multiplies involving the same sparse matrix and, therefore, should prove valuable for iterative and time stepping calculations.

FOREWORD AND ACKNOWLEDGEMENTS

This formal technical report entitled "3-D Seismic Code for ILLIAC IV," is submitted by Systems, Science and Software (S^3) to the Advanced Research Projects Agency (ARPA) and to the Defense Nuclear Agency (DNA). The report presents the first phase of a continuing effort to develop a versatile numerical scheme for computing 3-D elastic waves on the ILLIAC IV computing system. This work was accomplished under Contract No. DNA-001-72-C-0154 which was funded by ARPA and monitored by DNA. Colonel David C. Russell has been the ARPA program manager and Major F. J. Leech has been the DNA Project Scientist.

Dr. Gerald Frazier has been the S^3 Project Manager. The technical results presented in this report represent the work of a number of S^3 staff members in addition to the authors, notably: J. T. Cherry, K. G. Hamilton, and E. J. Halda for their finite difference calculations, W. Z. Savage for his assistance with the finite element calculations, and J. M. Bareno for his contributions in developing GLYPNIR and ASK coding. Also, valuable technical assistance has been obtained from Donna Firth of the IAC and Terry Layman of the University of Illinois (presently at UCSD).

TABLE OF CONTENTS

		Page
I.	INTRODUCTION	1
II.	NUMERICAL THEORY	3
	2.1 FINITE ELEMENT AND FINITE DIFFERENCE METHODS	3
	2.2 FE AND FD DIFFERENCE COEFFICIENTS	6
	2.3 DESIRABLE FEATURES FOR 3-D WAVE CALCULATIONS	18
	2.4 DISCRETE EQUATIONS, TIME STEPPING, AND ARTIFICIAL DAMPING	20
	2.5 NON-REFLECTING BOUNDARIES	25
III.	ELASTIC WAVE CALCULATIONS USING NON-PARALLEL PROCESSING	29
	3.1 SPHERICALLY SYMMETRIC ELASTIC EXPLOSION CALCULATIONS	29
	3.2 LAMB'S PROBLEM	32
	3.3 3-D SWIS CALCULATIONS	41
IV.	ILLIAC CODE FOR 3-D ELASTIC WAVES	49
	4.1 NUMERICAL PROBLEM DEFINITION AND 3-D GRID GENERATOR	49
	4.2 GENERATION AND STORAGE OF THE DIFFERENCE EQUATIONS	50
	4.3 SORTING THE A MATRIX	53
	4.3.1 The Sort Problem	53
	4.3.2 Brief Description of the Procedures	54
	4.3.3 A Lower Bound on the Time Required	55
	4.3.4 Phase A – Segmenting the File Into Bins	56

Preceding page blank

TABLE OF CONTENTS (Cont'd)

		Page
	4.3.5 Phase B – The Internal Sort	59
	4.3.6 Phase C – Reassembling the Pages	61
4.4	TIME STEPPING	62
	4.4.1 Algorithm	62
	4.4.2 Ideal Timing	68
4.5	COMPUTER RESULTS	70
V.	TEST CALCULATIONS USING 3-D ILLIAC CODE	73
5.1	PROBLEM SET UP AND RESULTS	73
5.2	TIMING RESULTS	80
VI.	SUMMARY AND CONCLUSIONS	83
VII.	REFERENCES	87
APPENDIX A:	FINITE ELEMENT FORMULATION	89
A.1	VIRTUAL WORK	89
A.2	FINITE ELEMENT SPATIAL REDUCTION	89
A.3	ISOPARAMETRIC ELEMENTS	94
APPENDIX B:	ARTIFICIAL DAMPING	101
B.1	CAUSES OF NUMERICAL OSCILLATIONS	101
B.2	MATHEMATICAL ANALYSIS OF ARTIFICIAL DAMPING	103
B.3	ESTIMATING AN OPTIMUM DAMPING COEFFICIENT	107

I. INTRODUCTION

During the past decade, the time period when high speed digital computers became readily available in the U. S., we have seen remarkable progress in our ability to analyze stress waves in solids. Our analysis techniques are no longer restricted to linear processes and geometrically simple shapes. With few exceptions, if stresses are induced into an elastic medium that can be approximated using one- or two-dimensional geometry, then state of the art computer codes can be applied to numerically simulate the process.

If, on the other hand, either the geometry or the source of excitation cannot be suitably approximated using only two spatial dimensions, then we find severe limitations in our ability to predict the resulting motions using existing computer codes. Conventional third-generation computers are heavily burdened by the massive calculations involved in simulating 3-D wave propagation.

The advent of the ILLIAC IV computing system will most certainly prove to be a valuable asset for processing the 3-D calculations in a parallel mode. It is our opinion, based on six months of designing and programming parallel algorithms for processing wave calculations, that the ILLIAC concept is well suited for the task. We anticipate processing 3-D wave computations on the ILLIAC IV about 40 times faster than on conventional computers, about 10 times faster than on the CDC 7600.

We expect to achieve further savings over existing 3-D codes as the result of the algorithm that is being employed in the ILLIAC code. The state-of-the-art in performing 3-D wave calculations is not sufficiently well developed to state how much savings is to be realized from our algorithm. It

is instructive, however, to note that the number of multiply and add operations per node that are required to complete one time step (250) is competitive with 2-D computer schemes. (We have recently formulated a new scheme for performing 3-D wave calculations in materials with arbitrarily nonlinear constitutive properties that requires only about 2000 multiply and add operations per element per time step).

Based on timing estimates for the ILLIAC IV, we have gauged the execution speed for performing 3-D linear stress wave propagation at 40 μsec per node per time step or 0.4 sec per time step for a 10^4-node arbitrarily skewed grid. There are a number of pressing problems in earthquake and explosion seismology that require such a 3-D capability. We note the following:

1. Earthquake ground motions generated by the spontaneous rupture of subsurface materials along a plane of weakness in a prestressed region in the earth's crust.

2. Explosion-generated stress waves in the elastic regime surrounding the inelastic zone of the explosion; also the anamolous seismic signal that results from the dynamic relaxation that takes place as the explosion-induced fracture zone is created in a prestressed geologic formation.

3. Ground motion and the influence of the subsurface geologic configuration on the amplitude, frequency, and duration of the surface motions.

4. The interaction between propagating seismic waves and underground structures.

The applicability of the 3-D elastic code will, of course, extend far beyond the seismic field.

II. NUMERICAL THEORY

2.1 FINITE ELEMENT AND FINITE DIFFERENCE METHODS

Both finite element (FE) and finite difference (FD) techniques have been widely used for calculating stress waves in solids. Considerable experience has been accumulated in the use of FD techniques for performing step-by-step calculations of propagating high intensity stress waves through geologic continua; while the major application of the FE techniques has been for performing wave calculations in geologic continua and civil structures where wave lengths on the order of the structural system are of interest.

The diversity in the application of these techniques probably comes much more from their historical development and application than from inherent advantages or restrictions in the two techniques. For example, we note that a reference to a dynamic FE computer code often carries with it the following numerical implications:

1. Element configurations that combine structural elements (beams, shells, etc.) with continuum elements (tetrahedra, hexahedra, etc.) into a single problem representation. No restrictions are placed on the location of node points.

2. Substitution of the constitutive laws into the equation of motion prior to discretization. The stiffness matrix of the FE displacement method combines three operations into one: spatial differencing of nodal displacements to get strain, constitutive laws to relate stress to strain, and spatial differencing of the stress field to obtain the body forces and

inertial forces. Stresses and strains are computed separately from the "main-stream" calculations for the purpose of computer output and, in non-linear calculations, for the purpose of modifying the elastic moduli.

3. Massive storage requirements due to retaining all of the influence coefficients between adjacent node points (the stiffness matrix).

4. Non-explicit time stepping using either modal superposition or implicit time stepping.

These connotations, which apply to most but not all FE computer codes, come in addition to our understanding of what is meant by the FE method: "A numerical technique for approximating continua by a discrete system composed of elements. The behavior of the continua within each element is characterized by interpolation functions; the amplitude of field variables at isolated node points serve as the participation coefficients for the element interpolation functions. Energy principles are used to determine the particular combination of interpolation functions that best satisfies the governing conservation equations. This procedure results in many algebraic equations involving nodal values of the field variables." The scheme for processing these algebraic equations should not be confused with the scheme used for generating the equations.

FD techniques for computing propagating stress waves also carry numerical implications that are independent of the basic FD method, which we will describe as: " A numerical technique in which a continuum is characterized at isolated node points, i.e., the values of field variables are prescribed only at isolated node points. A derivative of a field variable at a node point is approximated by some

predetermined combination of the values of the field variable at neighboring node points. Using this approach, the governing differential equations are approximated point by point throughout the continuum to yield a set of algebraic equations involving nodal values of the field variables."

We note some chracteristics shared by many of the large FD computer codes:

1. Zoning by planes that extend completely through the continuum so that each zone appears in the shape of a skewed rectangle (bricks in 3-D) with limited facility for accomodating structural appendages.
2. Explicit time stepping.
3. Minimal storage requirements due to the repetitious calculation of the influence coefficients between adjacent node points at each time step.
4. Separation of the calculations into stages so that constitutive laws are applied at an intermediate stage in each time step loop. Developing stresses from strains explicitly in each time step loop is most convenient for accomodating nonlinear material response in the numerical calculations.

We note that both the FE and the FD method characterize the governing equations for continua by a set of algebraic equations involving nodal values of the field variables. The FE method accomplishes this discretization in a particular way that involves interpolation functions and energy principles. When enumerated in this manner, we are led to conclude that the FE method is in fact a scheme for generating difference equations and therefore should be categorized as a type of FD method. Even so, there are some features of the FE

method that do not generally appear in conventional FD procedures:

1. Because the FE method is based on energy principles, the results of a consistent FE calculation provide bounds on certain quantities, e.g., strain energy and resonant frequencies.

2. Because the FE method is developed element-by-element, no special considerations are needed to model sharp spatial discontinuities in continuum properties; also no special treatment is needed to apply tractions to element surfaces, either on internal or boundary elements.

3. Because the discrete difference equations generated using the FE method relate nodal forces and nodal displacements, the influence coefficients between neighboring node points can be viewed physically as spring constants. This physical insight into the numerical scheme allows considerable flexibility in joining structural appendages into a numerical treatment.

2.2 FE and FD DIFFERENCE COEFFICIENTS

In order to emphasize the basic similarities between the FE and FD methods, we will present the difference coefficients that correspond to the two most widely used FE and FD methods. In so doing we will be providing a common basis from which to compare the two techniques. Indeed, we find that the least elegant FE analysis and the least elegant FD analysis are, in fact, identical insofar as the spatial discretization of linear isotropic continua using a uniform rectalinear grid. The other FE and FD techniques that are investigated in this section differ somewhat, but, when a

gross approximation is used in the integration scheme for evaluating the element stiffness matrix in the FE method, these two techniques also result in identical spatial differencing techniques.

Adopting subscript notation, we substitute the most general linear constitutive equation for an inhomogeneous anisotropic medium

$$\sigma_{ij} = E_{ijkl}\,\varepsilon_{kl}$$

$$= E_{ijkl}\,\frac{\partial u_k}{\partial x_l}$$

into the equations of motion

$$\frac{\partial \sigma_{ij}}{\partial x_j} + f_i = \rho \ddot{u}_i$$

to obtain the equation of motion expressed in terms of displacement

$$\frac{\partial}{\partial x_j}\left(E_{ijkl}\,\frac{\partial u_k}{\partial x_l}\right) + f_i = \rho \ddot{u}_i \tag{2.1}$$

where

x_i is the Cartesian coordinate (i=1,2,3)

u_i is the particle displacement

ε_{ij} is the strain

σ_{ij} is the stress

f_i is the body force

ρ is the mass density of the material

E_{ijkl} expresses the various elastic moduli. For isotropic materials we have

$$E_{ijkl} = \mu\,\delta_{ik}\delta_{jl} + \mu\,\delta_{il}\delta_{jk} + \lambda\delta_{ij}\delta_{kl}$$

Both FE and FD methods result in discrete representations of the spatial derivatives appearing in Eq. (2.1) of the form

$$\frac{\partial^2 u_k}{\partial x_i \partial x_j} \cong \frac{1}{\Delta x_i \Delta x_j} \sum_{\alpha=-1}^{+1} \sum_{\beta=-1}^{+1} \sum_{\gamma=-1}^{+1} w_{\alpha\beta\gamma}\, u_{k(\alpha\beta\gamma)} \tag{2.2}$$

for a rectilinear 3-D grid having a uniform grid spacing Δx_1, Δx_2, Δx_3, illustrated in Fig. 2.1. The various numerical schemes are characterized by the values of the 27 difference coefficients $w_{\alpha\beta\gamma}$, $\alpha = -1,0,+1$; $\beta = -1,0,+1$; $\gamma = -1,0,+1$.

The difference coefficients $w_{\alpha\beta\gamma}$ for the most elementary FD scheme are presented first. The coefficients for the particular mixed partial derivative $\frac{\partial^2}{\partial x_1 \partial x_2}$, which are typical of those for other mixed partial derivatives, are given by

$$w_{+1,+1,0} = w_{-1,-1,0} = +\tfrac{1}{4}$$

$$w_{+1,-1,0} = w_{-1,+1,0} = -\tfrac{1}{4} \tag{2.3}$$

with the remaining 23 coefficients equal to zero. The most elementary FD scheme for approximating $\frac{\partial^2}{\partial x_i^2}$ uses the difference coefficients

$$w_{\pm 1,0,0} = 1$$

$$w_{0,0,0} = -2 \tag{2.4}$$

and the remaining 24 coefficients are zero. This difference scheme for mixed partial and straight partial derivatives is

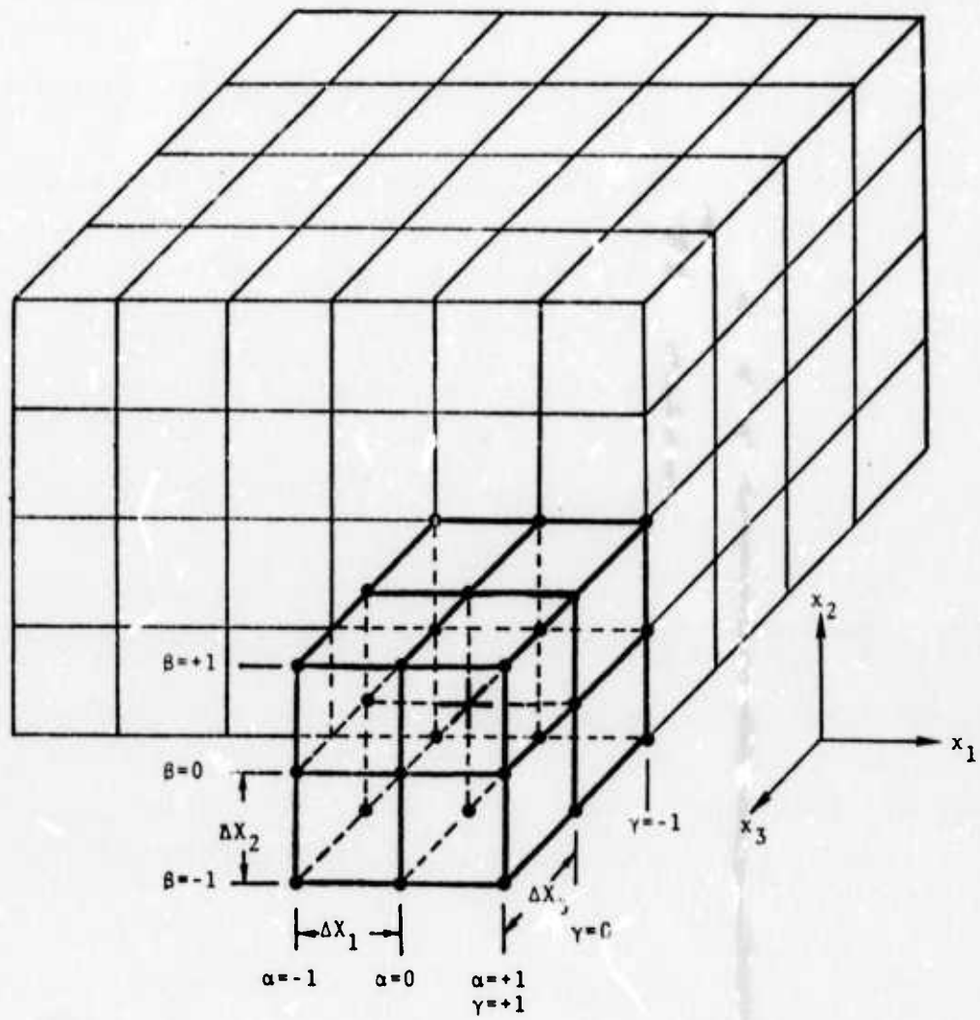


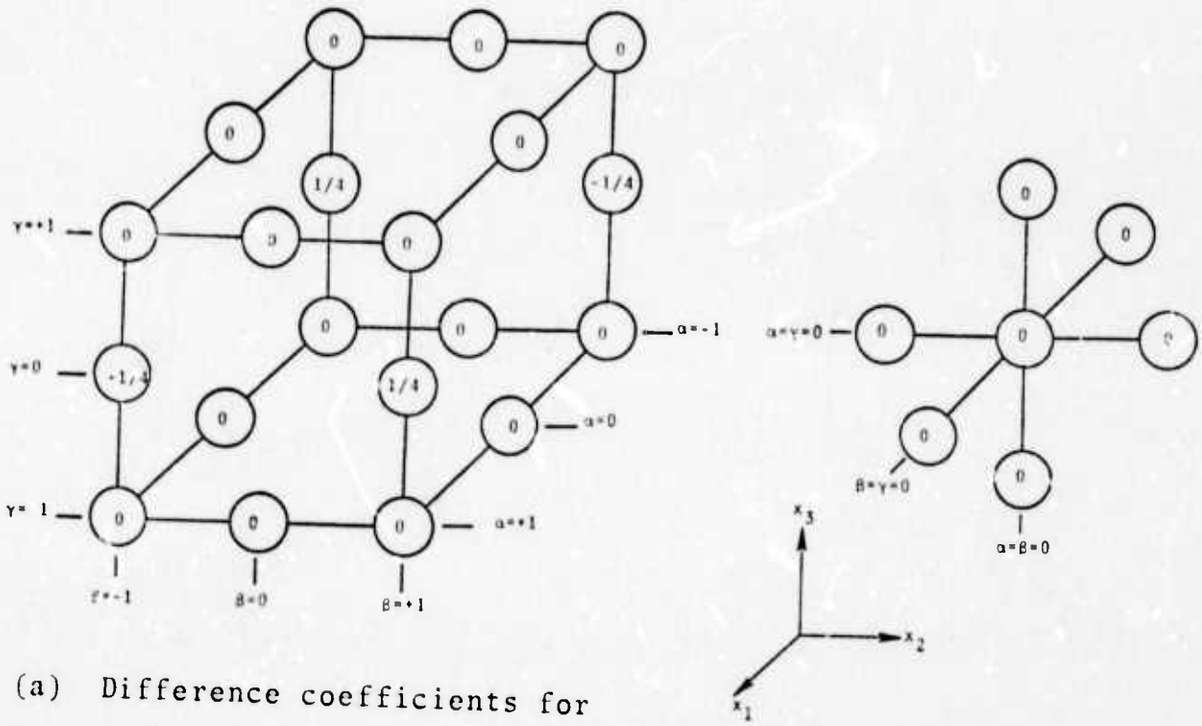
Fig. 2.1--Uniform rectalinear 3-D grid illustrating the concept of neighbor nodes.

presented graphically in Fig. 2.2.

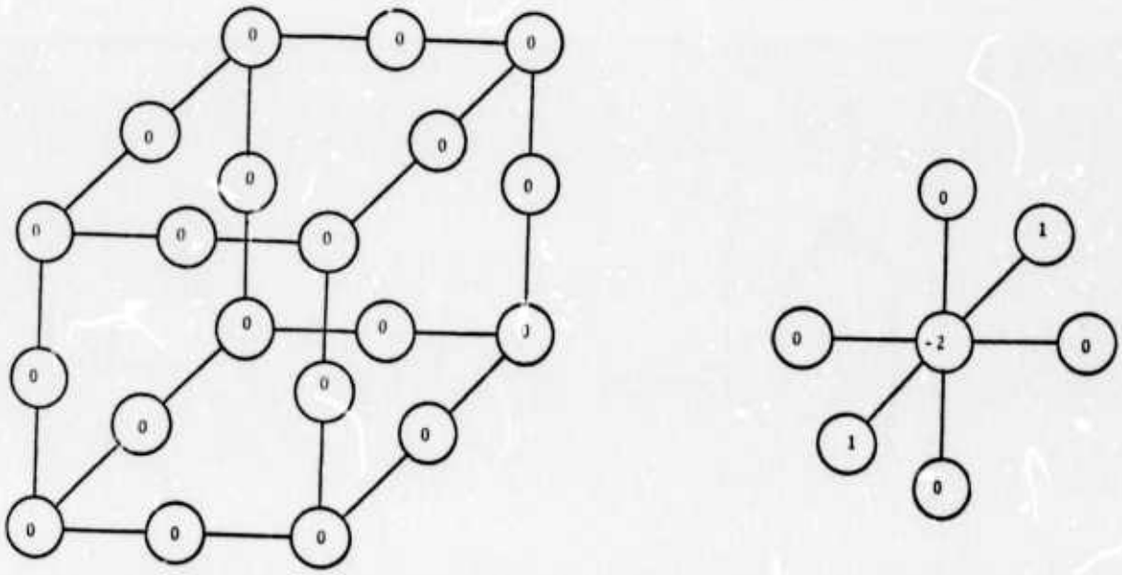
In the most elementary FE scheme, a brick element is formed from two complimentary sets of five tetrahedral elements, illustrated in Fig. 2.3. The corresponding difference coefficients using this FE configuration are found to be identical with those presented above and in Fig. 2.2 Hence, we are led to conclude that for a uniform rectilinear grid the most elementary FE and FD schemes will yield identical results if both schemes use the same time stepping procedure. This conclusion assumes the use of a simple lumped mass approximation in the FE scheme and comparable treatment of boundary conditions and forcing terms in both schemes.

We now examine a FD scheme, which is generally considered to be well suited for performing nonlinear calculations, based on its wide use in 2-D shock codes. It is developed in two stages: first the stresses (involving first partial derivatives of the displacement field) are computed at the centroid of the zones, then these stresses are differenced to give stress gradients (the second partial derivatives of displacement appearing in Eq. (2.1)) at the node points. We will refer to this FD procedure as the cell-centered-stress differencing method. The difference coefficients for the mixed partial derivative $\frac{\partial^2}{\partial x_1 \partial x_2}$ using this scheme become

$$
\begin{aligned}
w_{+1,+1,\pm 1} &= w_{-1,-1,\pm 1} = +1/16 \\
w_{+1,-1,\pm 1} &= w_{-1,+1,\pm 1} = -1/16 \\
w_{+1,+1,0} &= w_{-1,-1,0} = +1/8 \\
w_{+1,-1,0} &= w_{-1,+1,0} = -1/8
\end{aligned}
\tag{2.5}
$$



(a) Difference coefficients for $\frac{\partial^2}{\partial x_1 \partial x_2}$



(b) Difference coefficients for $\frac{\partial^2}{\partial x_1^2}$

Fig. 2.2--Difference coefficients for a 3-D rectalinear grid using the most elementary FD and FE (trahedrons) method.

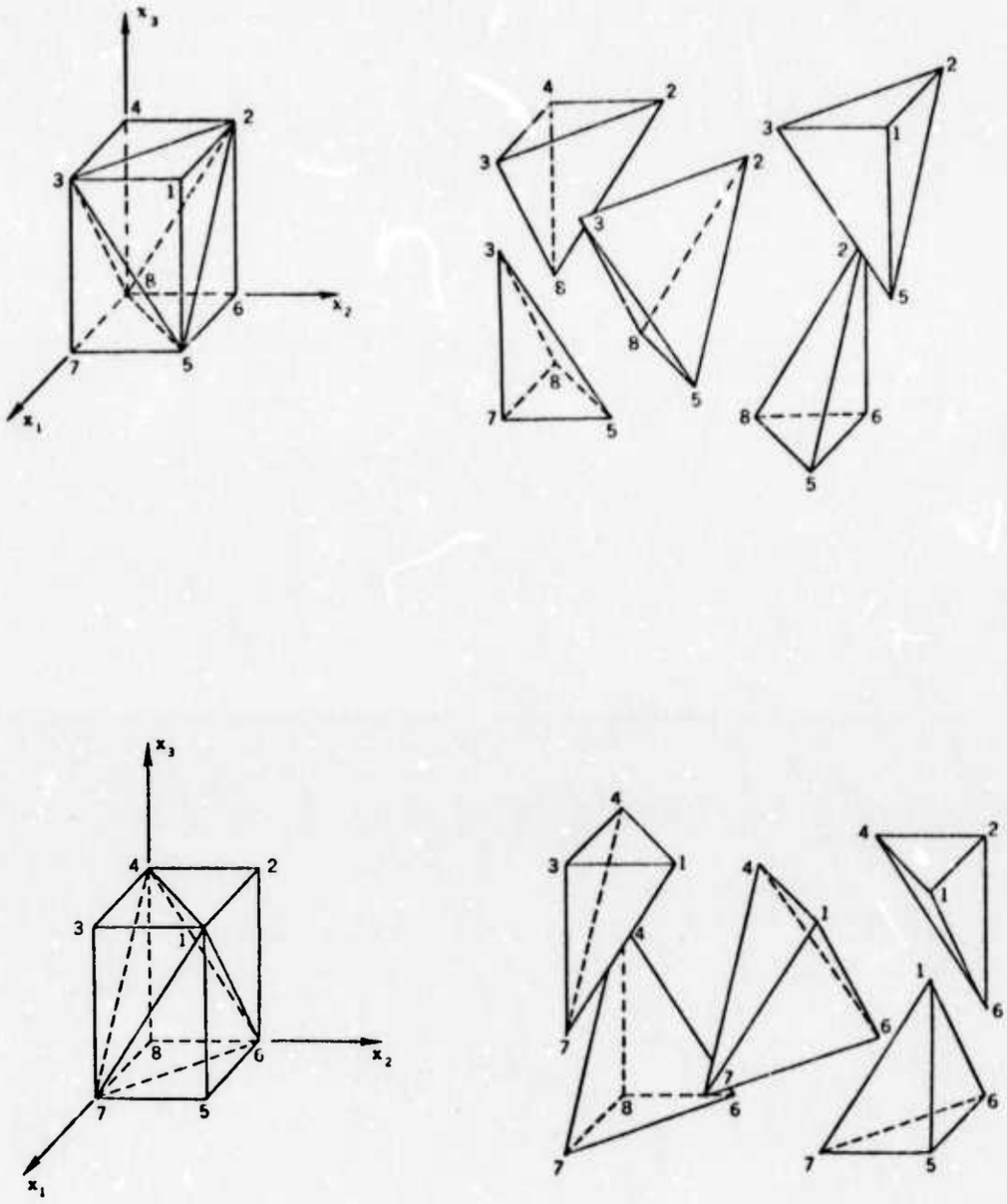


Fig. 2.3--Two complementary configurations for forming a brick from five tetrahedral elements.

with the remaining 15 influence coefficients equal to zero. The difference coefficients for $\frac{\partial^2}{\partial x_1^2}$ are given by

$$w_{\pm 1,\pm 1,\pm 1} = +1/16$$

$$w_{\pm 1,\pm 1,0} = w_{\pm 1,0,\pm 1} = +1/8$$

$$w_{0,\pm 1,\pm 1} = -1/8$$

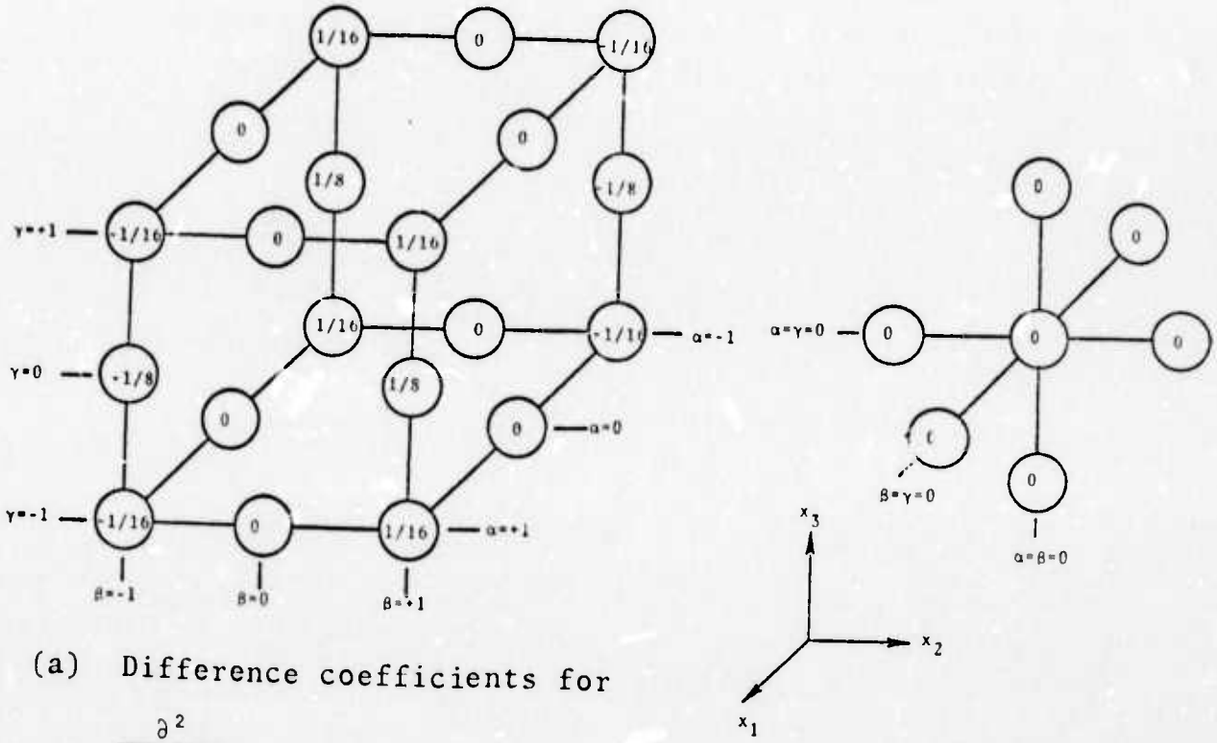
$$w_{\pm 1,0,0} = +1/4$$

$$w_{0,0,\pm 1} = w_{0,\pm 1,0} = -1/4$$

$$w_{0,0,0} = -1/2 \tag{2.6}$$

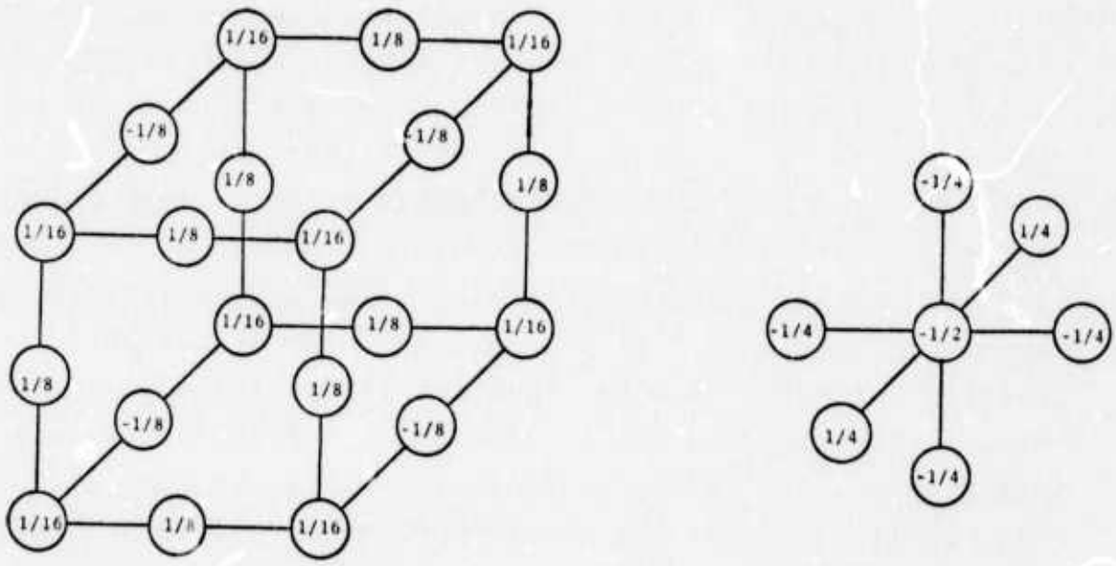
The coefficients for the cell-centered-stress differencing scheme are presented graphically in Fig. 2.4.

The cell-centered-stress differencing method has one undesirable feature that does not appear in any of the other methods presented in this report. The matrix that comprises the complete set of difference coefficients for a solid with traction-free boundary conditions has eighteen zero eigenvalues, twelve more than the six zero eigenvalues that correspond to the six rigid body displacements (three translation and three rotation). This means that a numerical grid can undergo deformations other than the six rigid body deformations without generating stresses. In particular, we see that all of the second order partial derivatives of Eq. (2.1) are identically zero for the twelve special displacement configurations



(a) Difference coefficients for

$$\frac{\partial^2}{\partial x_1 \partial x_2}$$



(b) Difference coefficients for $\frac{\partial^2}{\partial x_1^2}$

Fig. 2.4--Difference coefficients using the cell-center-stress FD method.

$$u_k(\alpha\beta\gamma) = (-1)^{\alpha+\beta}$$

$$u_k(\alpha\beta\gamma) = (-1)^{\alpha+\gamma}$$

$$u_k(\alpha\beta\gamma) = (-1)^{\beta+\gamma}$$

$$u_k(\alpha\beta\gamma) = (-1)^{\alpha+\beta+\gamma}$$

with $k = 1, 2, 3$ when we use the cell-centered-stress differencing scheme. This feature of the cell-centered-stress method is sometimes referred to as "hour glass instability".

The FE scheme used in the 3-D test calculations in Sections III and V of this report employs a brick element which permits linear variations in the displacement field along the twelve edges of the element, quadratic variations along skewed lines on the six faces of the element, and cubic variations along skewed lines passing through the element. For the case of a rectilinear brick geometry, the displacement field is approximated by

$$u_i(\underset{\sim}{x}) \cong \sum_{p,q,r=0}^{1} C_{i(pqr)} x_1^p x_2^q x_3^r$$

$$= \frac{1}{\Delta x_1 \Delta x_2 \Delta x_3} \sum_{\alpha,\beta,\gamma=0}^{1} (-1)^{\alpha+\beta+\gamma+1} \left| x_1 + x_{1(\alpha)} - 2x_{1(\frac{1}{2})} \right| \cdot$$

$$\left| x_2 + x_{2(\beta)} - 2x_{2(\frac{1}{2})} \right| \cdot \left| x_3 + x_{3(\gamma)} - 2x_{3(\frac{1}{2})} \right| u_{i(\alpha\beta\gamma)} \qquad (2.7)$$

where $x_{1(\alpha)}$, $x_{2(\beta)}$, $x_{3(\gamma)}$ denotes the Cartesian coordinates of the α, β, γ planes of Fig. 2.1, respectively, and $x_i(\frac{1}{2})$ denotes the element centroid. The theory behind the development of this trilinear (displacement varies linearly in x_1, x_2, and x_3) brick element is presented in Appendix A. The difference coefficients for the mixed partial derivative $\frac{\partial^2}{\partial x_1 \partial x_2}$

that result from the trilinear brick element are given by

$$w_{+1,+1,\pm 1} = w_{-1,-1,\pm 1} = +1/24$$

$$w_{+1,-1,\pm 1} = w_{-1,+1,\pm 1} = -1/24$$

$$w_{+1,+1,0} = w_{-1,-1,0} = +1/6$$

$$w_{+1,-1,0} = w_{-1,+1,0} = -1/6 \tag{2.8}$$

with the remaining 15 coefficients equal to zero. The trilinear brick element coefficients for $\frac{\partial^2}{\partial x_1^2}$ are

$$w_{\pm 1,\pm 1,\pm 1} = +1/36$$

$$w_{\pm 1,\pm 1,0} = w_{\pm 1,0,\pm 1} = +1/9$$

$$w_{0,\pm 1,\pm 1} = -1/18$$

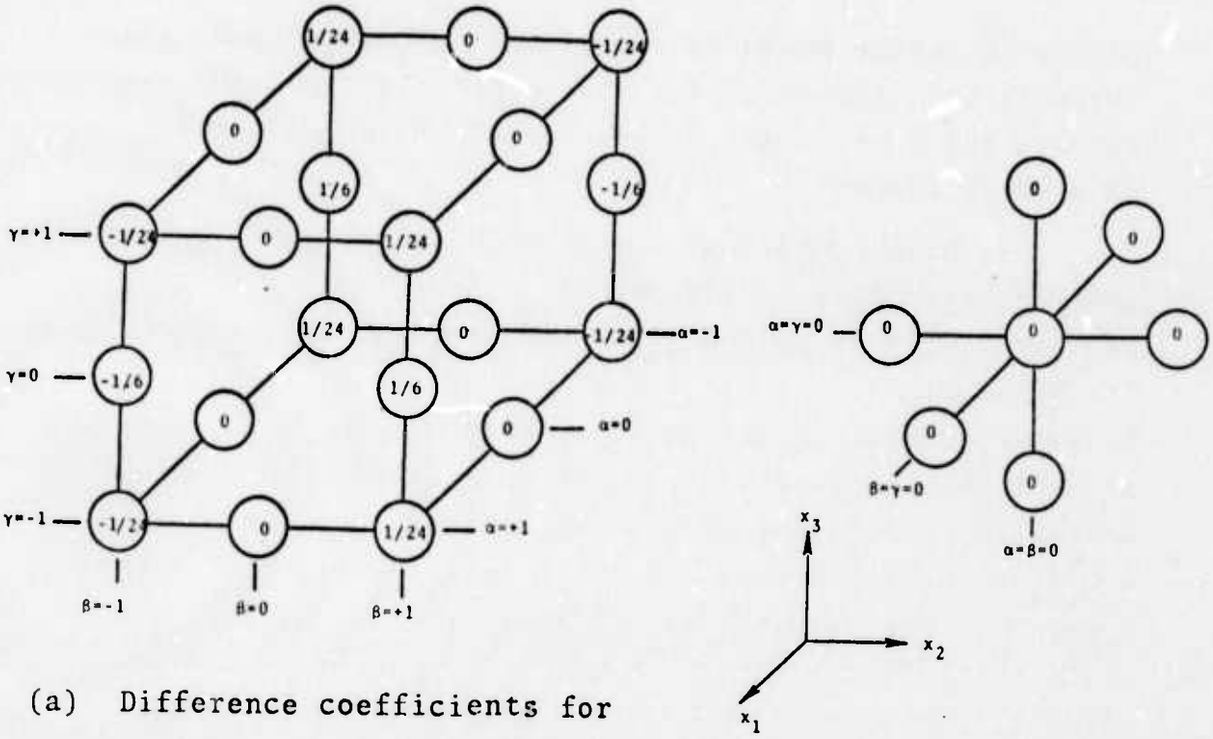
$$w_{\pm 1,0,0} = +4/9$$

$$w_{0,0,\pm 1} = w_{0,\pm 1,0} = -2/9$$

$$w_{0,0,0} = -8/9 \tag{2.9}$$

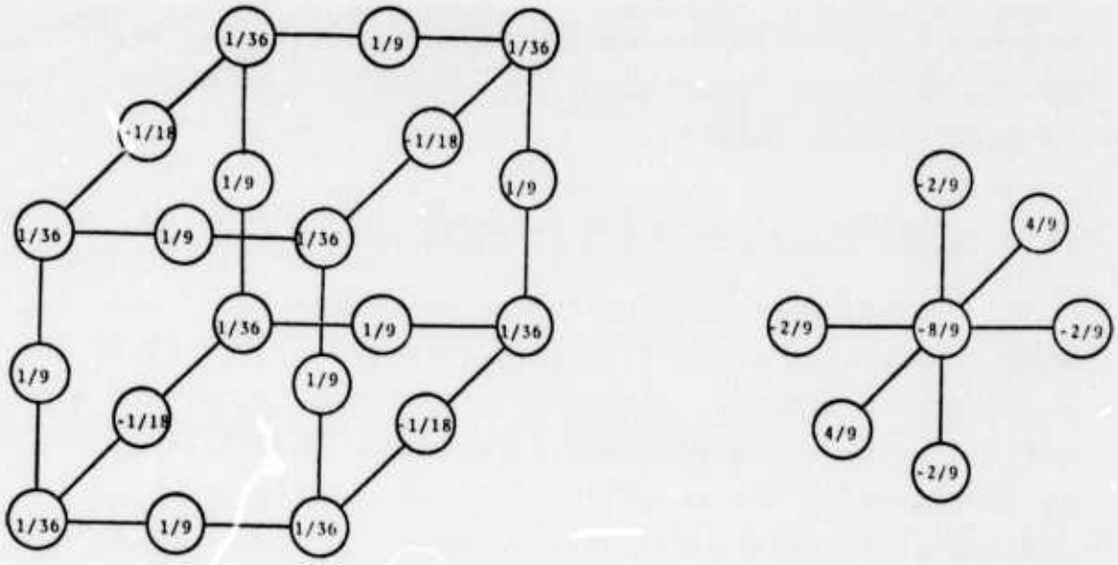
These coefficients are presented graphically in Fig. 2.5.

We note from Figs. 2.4 and 2.5 that the cell-centered-stress FD method does not correspond to the trilinear brick FE idealization. The trilinear FE scheme would be obtained, however, from a FD method in which the stresses are calculated slightly closer to the node at which the equation of motion is being applied, namely at the 1/3 points of the cells.



(a) Difference coefficients for

$$\frac{\partial^2}{\partial x_1 \partial x_2}$$



(b) Difference coefficients for $\frac{\partial^2}{\partial x_1^2}$

Fig. 2.5--Difference coefficients using trilinear brick elements in the FE method.

Such a FD scheme would be rid of the so-called "hour glass instability"; however, it would be excessively cumbersome to compute and store eight stress tensors at the eight 1/3 points of a brick element.

We have found that the 3-D FE brick reduces to the cell-centered-stress differencing scheme if strain energy density in the FE method is assumed to be constant throughout the brick element. This is equivalent to using a 1-point Gaussian quadrature integration procedure for evaluating the element stiffness matrix. Nonlinear displacement modes that appear in the element displacement field (x_1x_2, x_1x_3, x_2x_3, $x_1x_2x_3$ of Eq. (2.7)) result in no strain energy using the 1-point integration scheme. That is to say, grid displacements that give rise exclusively to nonlinear interelement element displacement modes (hour glass grid deformations) result in no restoring forces (stress) in the continuum and therefore go unchecked in the numerical calculations. It is customary in FE codes to employ 2- or 3-point integration schemes for each dimension in the strain energy integral. Consequently the "hour glass instability" does arise in the FE differencing scheme.

2.3 DESIRABLE FEATURES FOR 3-D WAVE CALCULATIONS

In order to develop an optimal scheme for computing elastic waves in solids, we have selected from both FE and FD computing schemes. The following features, listed approximately in the order of their importance, have been selected to characterize the computing technique for processing 3-D elastic waves on the Illiac IV.

1. Explicit Time Stepping — A small time step on the order of the Courant value ($\Delta t_c = (\Delta x/V_p)$ = grid dimension/P-wave velocity) is needed to achieve satisfactory accuracy in the propagation of sharp

wave fronts (wave length equal to about 10 grid dimensions. This is true even for implicit schemes, which may be stable using much greater time steps. Numerical experimentation at S^3 indicates that a three-point ($t-\Delta t$, t, $t+\Delta t$) explicit scheme using a time step of $0.5\ \Delta t_c$ can match the accuracy of the best three-point implicit scheme that uses twice this time step; yet the implicit scheme requires many times more operations per time step. From Table I we see that a direct implicit scheme, which operates using the two-pass back substitution procedure (most commonly used procedure in implicit FE codes) at each time step, requires nearly thirty times more calculations per time step than an explicit scheme for a 20 by 20 by 20 3-D grid. Furthermore, the explicit time stepping scheme allows much greater flexibility for modifying constitutive laws as the calculation proceeds.

2. Flexible Grid Configurations — In the zoning of 3-D solids, care must be exercised to achieve something close to an optimal grid configuration. Rectilinear grids, for example, would probably be inappropriate for modeling a buried explosion in 3-D geometry. The node numbering scheme of FE, in which the nodes are numbered consecutively from one through the total number of nodes in the grid, accomodates arbitrary grid configurations and therefore has been adopted. Also, the FE state of the art for developing difference equations for complex grid configurations is more advanced than that for conventional FD methods. Curved tetrahedron and hexahedron elements are systematically developed in the FE method using

isoparametric element techniques, developed in Appendix A.

3. Quasi-linear Constitutive Laws — In the interest of developing a fast computing scheme for treating very large grids ($\sim 10^5$ nodes), the initial code configuration assumes no change in the elastic constants of the material as the time stepping proceeds. However, a restricted class of nonlinear behavior can be accommodated in the present version of the code. Less restrictive nonlinear constitutive laws can be accommodated at some later date. In the present code configuration, mass storage is used effectively to avoid the repetitive calculation of influence coefficients at each time step (see Table I).

2.4 DISCRETE EQUATIONS, TIME STEPPING, AND ARTIFICIAL DAMPING

Wave propagation through a spatially discrete linear system can be described using matrix notation by the equation

$$[M]\{\ddot{U}_i(t)\} + [C_{ij}]\{\dot{U}_j(t)\} + [K_{ij}]\{U_j(t)\} = \{F_i(t)\} \qquad (2.10)$$

where $\{U_i(t)\}$ is a column listing of the $i^{\underline{th}}$ component of displacement of the node points throughout the system, $\{F_i(t)\}$ is a column listing of the $i^{\underline{th}}$ component of nodal forcing terms, $[M]$ is a diagonal listing of the mass lumped at the various node points, $[K_{ij}]$ is a sparsely populated banded matrix of interaction terms between neighboring node points, and $[C_{ij}]$ is matrix that is introduced for the special purpose of damping certain features of the numerical solution. The matrix equation above results from spatially differencing the wave equation for an elastic continuum, Eq. (2.1).

TABLE I

3-D ELASTIC STRESS WAVE PROPAGATION

	Step-by-Step Integration of the Equations of Motion			
	Explicit		Implicit	
	Difference Eqs. Calc. Initially and Stored	Difference Eqs. Calc. at Each Time Step	Difference Eqs. Calc. Initially and Stored	Difference Eqs. Calc. at Each Time Step
Number of Multiply and Add Operations per Time Step	250 N*	4000 N	Direct** $(40+18\ N^{2/3})N$ Iterative*** 250 NI	Direct Unfeasible Iterative 4000 NI****
Storage Requirement	251 N	39 N	Direct $(15+2N^{2/3})N$ Iterative 251 N	Direct $(45+2N^{2/3})N$ Iterative 39 N
Outstanding Advantages	1. Very fast 2. Minimal routing between PE's	1. Easily extended to nonlinear behavior 2. Minimal storage requirements	1. Can use large time steps; static solutions can be computed 2. Somewhat better accuracy than explicit schemes	
S' Numerical Wave Propagation Computer Codes		1-D Finite Diff. SKIPPER RIP		
	2-D Finite Element in preparation	2-D Finite Diff. CRAM HELP	2-D Finite Element DYNA	
	3-D Finite Element SW1S		3-D Finite Element SAP3	

* N is the number of nodes in the 3-D grid.

** The matrix of influence coefficients is triangularized initially with back substitution at each time step.

*** An iterative scheme is used to evaluate the advanced displacements.

**** I is the number of iterations.

The "stiffness" matrix $[K_{ij}]$ contains the difference coefficients discussed in Section 2.2. The notation is typical of that used in the finite element literature, Frazier (1972). Equation (2.10) is developed in Appendix A.

The influence that certain special types of damping matrices $[C_{ij}]$ have on the resulting calculations can be determined for linear systems by transforming Eq. (2.10) into modal coordinates, Frazier (1969), also see Appendix B. In so doing, the equations become decoupled (independent from one another) for particular forms of the damping matrix. For these special cases the influence of the damping on the transient behavior of the discrete system can be predicted. The result is that

$$[C_{ij}] = \beta[K_{ij}] \tag{2.11}$$

causes the eigenvectors to damp as the square of their natural frequencies, i.e., ω^2 damping; whereas

$$[C_{ij}] = \alpha\ \delta_{ij}[M]$$

cause uniform damping of all of the eigenvectors, i.e., frequency independent damping, as developed in Appendix B. The β-type damping satisfies the condition of no damping for zero strain rate in an element. The α-type damping has little if any utility for wave propagation since this form damps all frequencies, even rigid body displacements with no strain.

The β-type damping of Eq. (2.11) is substituted into Eq. (2.10) and a more compact notation is adopted using underscores for the directional component subscripts i and j to obtain

$$[M]\{\ddot{\underline{U}}_t\} + \beta[\underline{\underline{K}}]\{\dot{\underline{U}}_t\} + [\underline{\underline{K}}]\{\underline{U}_t\} = \{\underline{F}_t\} \tag{2.12}$$

for the collection of equations of motion at time t. We use a central difference approximation in time to obtain

$$\frac{1}{\Delta t^2}[M]\{\underline{U}_{t+\Delta t} - 2\underline{U}_t + \underline{U}_{t-\Delta t}\} + \frac{\beta}{2\Delta t}[\underline{\underline{K}}]\{\underline{U}_{t+\Delta t} - \underline{U}_{t-\Delta t}\}$$

$$+ [\underline{\underline{K}}]\{\underline{U}_t\} = \{\underline{F}_t\} \tag{2.13}$$

where

$$\Delta t \leq \Delta t_c = \frac{\Delta x}{V_p} \tag{2.14}$$

is the time step, bounded by the Courant time Δt_c, which is equal to the time it takes for a P-wave to cross one grid dimension, Δx.

We solve for the nodal displacements at the advanced time step from Eq. (2.14) to obtain our algorithm for stepping the numerical calculations through time

$$\{\underline{U}_{t+\Delta t}\} = [\underline{\underline{1}}+\underline{\underline{D}}]^{-1}\left(\{\underline{P}_t\} - [\underline{\underline{1}}-\underline{\underline{D}}]\{\underline{U}_{t-\Delta t}\} + [\underline{\underline{2}}-\Delta t^2 M^{-1}\underline{\underline{K}}]\{\underline{U}_t\}\right) \tag{2.15}$$

where

$$[\underline{\underline{D}}] = \frac{\Delta t}{2}\beta[M]^{-1}[\underline{\underline{K}}]$$

and

$$\{\underline{P}_t\} = \Delta t^2[M]^{-1}\{\underline{F}_t\}$$

We note that, while the inversion of the diagonal matrix $[M]$ is trivial, the inversion of the nondiagonal matrix $[\underline{\underline{1}}+\underline{\underline{D}}]$ is very tedious. However, for the case when $\beta = 0$ (no damping), this term reduces to an identity matrix. This leads us to believe that perhaps an approximate inverse can be obtained by perturbation techniques, since the influence of the damping terms should only be to perturb the undamped solution.

Following this reasoning we approximate the inverse by

$$[\underline{\underline{1}}+\underline{\underline{D}}]^{-1} = [\underline{\underline{1}}-\underline{\underline{D}}+\underline{\underline{D}}^2 \ldots]$$

$$\cong [\underline{\underline{1}}-\underline{\underline{D}}] \tag{2.16}$$

The geometric expansion converges to the desired inverse provided the highest eigenvalue of $[\underline{\underline{D}}]$, λ_{max}, is less than unity, i.e.,

$$1 \geq \lambda_{max} = \frac{\Delta t}{2} \beta \omega_{max}^2$$

$$\cong \frac{\Delta t}{2} \beta \left[\frac{4(\lambda+2\mu)}{\rho \Delta x^2} \right]$$

$$= 2 \frac{\Delta t}{\Delta x^2} \beta V_P^2$$

therefore

$$\beta \leq \beta_{max} = \frac{\Delta x^2}{2 \Delta t V_P^2} \tag{2.17}$$

where ω_{max} is the highest natural frequency of the discrete system, λ and μ are elastic material constants, ρ is the mass density, and $V_P = \sqrt{\frac{\lambda+2\mu}{\rho}}$ is the P-wave velocity.

A technique for predicting an optimum damping coefficient has been developed in Appendix B. Here we find that optimal values of β for removing high frequency oscillations from the numerical calculations of a sharp wave front lie within the convergence criteria of Eq. (2.17). This is true provided the oscillations can be tolerated for about ten time steps following their initiation by rapid changes in the loading function or the constitutive laws.

Using the approximate inverse expressed by Eq. (2.16), we obtain our time stepping algorithm, arranged so as to minimize the number of multiplication and add operations per time step,

$$\{\underline{U}_{t+\Delta t}\} = \{\underline{P}_t\} - \{\underline{U}_{t-\Delta t}\} + 2\{\underline{U}_t\}$$

$$- \Delta t^2 [M]^{-1} [\underline{\underline{K}}] \left\{ \frac{\beta}{2\Delta t} \underline{P}_t - \frac{\beta}{\Delta t} \underline{U}_{t-\Delta t} + \left(\frac{\beta}{\Delta t} + 1\right) \underline{U}_t \right\} \qquad (2.18)$$

in which we have dropped terms involving $[\underline{\underline{K}}][\underline{\underline{K}}]$ for the purpose of computing economy. The errors introduced by dropping these terms are of the same order as those for approximating the inverse in Eq. (2.16); such terms do not exist with $\beta = 0$.

The suitability of the time stepping algorithm above is demonstrated in Section 3.3 by test calculations involving sharp wave fronts propagating through a chain of 3-D elements.

2.5 NON-REFLECTING BOUNDARIES

Numerical treatments of elastic waves in the earth are frequently plagued by waves that reflect from grid boundaries. Often, the arrival of these reflected waves is sufficiently delayed by simply extending the grided region so as to not interfere with calculations in the source region. This procedure can be very costly in terms of computer effort, perhaps excessively so for many 3-D calculations.

A more appropriate solution for avoiding these reflected waves would be to develop boundary conditions that do not reflect incident waves. Many researchers have addressed this problem with limited success. The degree of success of the various schemes depends on either the frequency content of the incident wave or its angle of incidence with the boundary.

We have developed and adopted an extremely simple boundary condition that reflects less than 4 percent of the normal incident wave energy. On a practical basis, the elimination of normal incident waves is of greatest importance since the waves that hit the boundaries at an angle do not reflect directly back at their source. We have not yet tested the scheme for non-normal incident waves to determine to what extent these waves will be eliminated from the grided region.

The reflected waves from normal incident plane waves at a free surface have the same displacement character as the incident waves; they are simply propagating in the opposite direction. The reflected waves from normal incident plane waves at a rigid boundary surface also have the same character as the incident waves but with the opposite sign. Therefore a numerical solution that is taken as the average of the rigid boundary and the free boundary calculations should possess no reflected waves. Denoting the free boundary solution as $\left\{\underline{U}^{(2)}_{t+\Delta t}\right\}$ and the rigid boundary solution as $\left\{\underline{U}^{(1)}_{t+\Delta t}\right\}$, then the non-reflecting solution is given by

$$\left\{\underline{U}^{(3)}_{t+\Delta t}\right\} = 1/2\left\{\underline{U}^{(1)}_{t+\Delta t}\right\} + 1/2\left\{\underline{U}^{(2)}_{t+\Delta t}\right\}. \tag{2.19}$$

One way to apply this scheme for eliminating boundary reflections would be to carry out two complete solutions, $\{U^{(1)}\}$ and $\{U^{(2)}\}$. Then from these two solutions, the solution with boundary reflections removed could be computed from Eq. (2.19). Such a technique would be excessively tedious. A preferable technique for removing boundary reflections is to apply Eq. (2.19) over the duration of a single time step taking into account the appropriate initial conditions for the time step, $\{U_t\}$ and $\{U_{t-\Delta t}\}$. This is the technique that is developed below.

First we will explain how $\left\{\underline{U}^{(2)}_{t+\Delta t}\right\}$ is obtained. Since the time stepping algorithm given by Eq. (2.18) applies to free boundaries with $\{\underline{P}_t\}$ zero on the boundaries, the free surface solution computed over a single time step comes directly from the time stepping algorithm, i.e., $\left\{\underline{U}^{(2)}_{t+\Delta t}\right\} = \left\{\underline{U}_{t+\Delta t}\right\}$ for all nodes.

The rigid boundary solution $\left\{\underline{U}^{(1)}_{t+\Delta t}\right\}$ differs from $\{U_{t+\Delta t}\}$ as taken directly from Eq. (2.18) only at points on the non-reflecting boundary. For node points along the non-reflecting boundary, the rigid boundary nodal displacements are given by

$$\left\{\underline{U}^{(1)}_{t+\Delta t}\right\} = \left\{\left(1 + \frac{\beta}{\Delta t}\right)\underline{U}_t - \frac{\beta}{\Delta t}\,\underline{U}_{t-\Delta t}\right\} .$$

This is the term in Eq. (2.18) that allows boundary displacements to influence internal points over the duration of a single time step. We note that, with $\beta = 0$, this is equivalent to setting the particle velocity at $t + \frac{1}{2}\Delta t$ equal to zero for those nodes on a rigid boundary.

Consequently, the advanced displacement, taken as the average of the rigid and free advanced displacements, becomes

$$\left\{\underline{U}^{(3)}_{t+\Delta t}\right\} = \left\{\underline{U}_{t+\Delta t}\right\}$$

for nodes not along the non-reflecting boundary and

$$\left\{\underline{U}^{(3)}_{t+\Delta t}\right\} = 1/2\left\{\underline{U}_{t+\Delta t}\right\} + 1/2\left\{\left(1 + \frac{\beta}{\Delta t}\right)U_t - \frac{\beta}{\Delta t}\,U_{t-\Delta t}\right\} \qquad (2.20)$$

for those nodes along the non-reflecting boundary. The advance displacements $\left\{\underline{U}^{(3)}_{t+\Delta t}\right\}$ are then used in Eq. (2.18) at the next time step.

Test calculations using this technique for eliminating boundary reflections from normal incident waves are presented in Section 3.3.

III. ELASTIC WAVE CALCULATIONS USING NON-PARALLEL PROCESSING

3.1 SPHERICALLY SYMMETRIC ELASTIC EXPLOSION CALCULATIONS

As the first problem for performing comparative calculations using conventional FE and FD techniques, we have computed the elastic compression waves that radiate from a spherical cavity undergoing an exponentially decaying pressure step. The exact solution, Blake (1952), contains a discontinuity in the stresses and the particle velocity at the wave front and therefore serves as a severe test of the numerical techniques for simulating body wave propagation.

The parameters used in the calculation are as follows:

r_{cavity} = 10 m

Δr = zone size = 0.1 m

V_P = P-wave velocity = 5 km/sec

V_S = S-wave velocity = 2.5 km/sec

ν = Poisson's ratio = 1/3

ρ = mass density = 2.0 g/cc

$f_{(t)}$ = cavity load (kbar) = $e^{-t}\ 10^{-3}$

Computed particle velocities at 2.06 msec following the application of the pressure loading are presented in Fig. 3.1.

The FD calculations have been reported on previously, Cherry, et al. (1972). They were performed using the 1-D SKIPPER code for spherically symmetric waves with an explicit time stepping scheme.

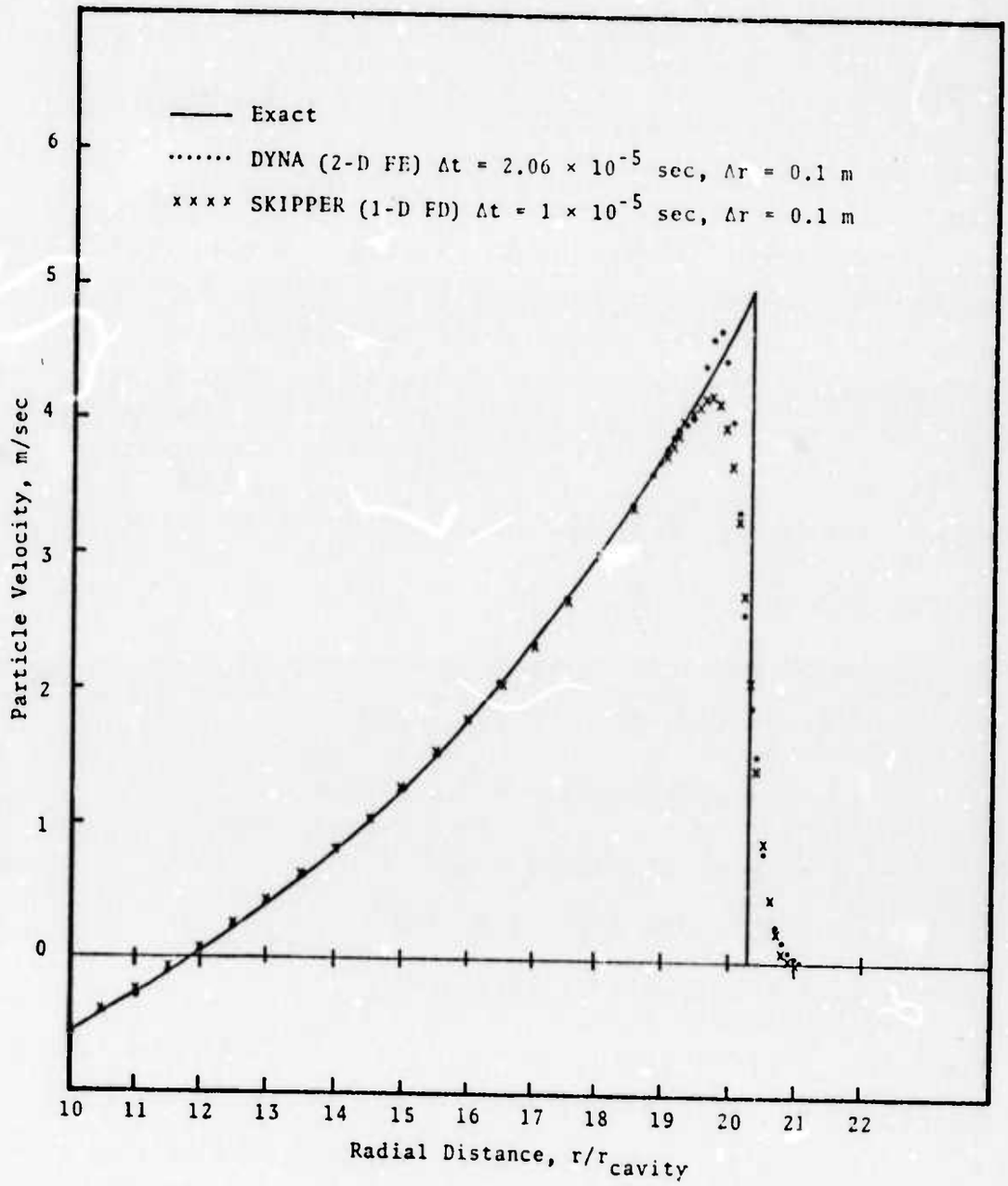


Fig. 3.1--FE and FD computed particle velocity at 0.00206 sec following the application of an exponentially decaying pressure step to a spherical cavity.

The FE calculations were performed for an axisymmetric wedge of elements using the 2-D DYNA code. The DYNA code, which uses an implicit time stepping scheme, was originally developed by Wilson (1969). In this code, Wilson's original time stepping scheme has been replaced by an extended form of Newmark's β method, Newmark (1959), see also Goodreau (1970).

Based on the FE and FD calculations for the pressure-loaded cavity, we have made the following observations:

1. Both techniques result in numerical oscillations (similar to the Gibb's phenomena associated with a truncated Fourier series) following the discontinuous wave front when no artificial damping is used in the calculations.

2. A reduction in Δt does not significantly improve either numerical calculation unless it is accompanied by a reduction in the zone size. We note that the implicit FE calculations use a time step equal to the Courant time step ($\Delta t_c = \Delta r/V_p$), which is twice as large as the time step used in the explicit FD calculations.

3. The maximum particle velocity from the FE calculation is somewhat higher than the FD peak; however, this is probably mainly the result of slightly less artificial damping in the FE calculations.

4. The displacements that are computed by the two codes follow the exact solution equally well throughout the calculation with a maximum of 3 percent deviation from the exact displacement at the cavity wall.

5. Neither solution deteriorates significantly at times much greater than that shown in Fig. 3.1.

The spherically symmetric elastic explosion calculations were also performed using a two-dimension axisymmetric grid over one quadrant in order to test for deviations from spherical symmetry. The FD calculations were performed using the 2-D Lagrangian CRAM code. Five decimal places of spherical symmetry were achieved in the FD calculations. The FE DYNA calculations displayed about 2 percent variation from true spherical symmetry as the result of approximations in terms containing 1/r, a consequence of axisymmetric geometry. This result has no direct implication on 3-D calculations, since terms of this type do not appear.

3.2 LAMB'S PROBLEM

Certainly one of the most challenging problems for testing the limitations of a numerical procedure for computing elastic waves is Lamb's Problem, Lamb (1904), Ewing, et al., (1957): An impulse loading applied at a point on the free surface of a half-space. The exact solution involves sharply peaked wave configurations. A significant portion of the seismic energy is channeled along the free surface in the form of a Rayleigh wave.

The solution to a delta function loading in time and space applied at the surface of the half-space serves as the Green's function for obtaining solutions to problems with nonsingular source conditions. Viecelli (1971) has used this approach for obtaining smoothed wave forms for comparison with numerical results obtained using the 2-D FD TENSOR code. He treated the free surface response to a uniform strip loading (of width 2a = 8 cm = 8 grid dimensions) applied normal to the free surface with the time history

$$p(t) = \frac{L}{2a\pi\tau_0[1+(t/\tau_0)^2]} \tag{3.1}$$

for $-\infty < t < +\infty$, see Fig. 3.2. The sharpest loading history that Viecelli treated was for $\tau_0 = 5$ μsec ($\Delta t \simeq 1$ μsec) with a total impulse $L = 1.97 \times 10^4$ dyne-sec per cm normal to the section of plane strain. Using 25,000 zones in the 2-D plane-strain calculation, Viecelli was able to reproduce the Rayleigh wave displacements at the free surface quite well.

Halda and Cherry (1972) employed S^3's 2-D FD CRAM code for computing the Rayleigh wave described in Viecelli's report. The following parameters were used in the calculation:

ρ = mass density = 2.77 g/cm^3

V_P = P-wave velocity = 5.55×10^5 cm/sec

V_S = S-wave velocity = 3.145×10^5 cm/sec

V_R = Rayleigh wave velocity = 2.898×10^5 cm/sec

Δt = time step = 1 μsec

$\Delta x = \Delta y$ = zone size = 1 cm

$2a$ = width of strip load = 8 cm

$$p(t) = \text{surface load for } t \geq -20 \ \mu\text{sec} = \frac{1.7\times10^8}{1+(t/5)^2} \ \text{dyn/cm}^2$$

$$L = \text{total impulse} = 2a \int_{-20}^{+\infty} p(t)dt = 1.97 \times 10^4 \ \text{dyn-sec/cm}$$

Results from the CRAM calculation are compared with Viecelli's analytic solution along the free surface at t = 80 μsec in Fig. 3.3. The agreement between the numerical and the analytical displacement of the free surface is comparable to that achieved by Viecelli using the TENSOR code. Incidently, both the CRAM and the TENSOR code use the cell centered stress differencing scheme presented in Section 2.2.

It was our intention to employ a FE code for the Rayleigh wave computation above. However, the DYNA code, referred to in the previous section, is restricted to grids

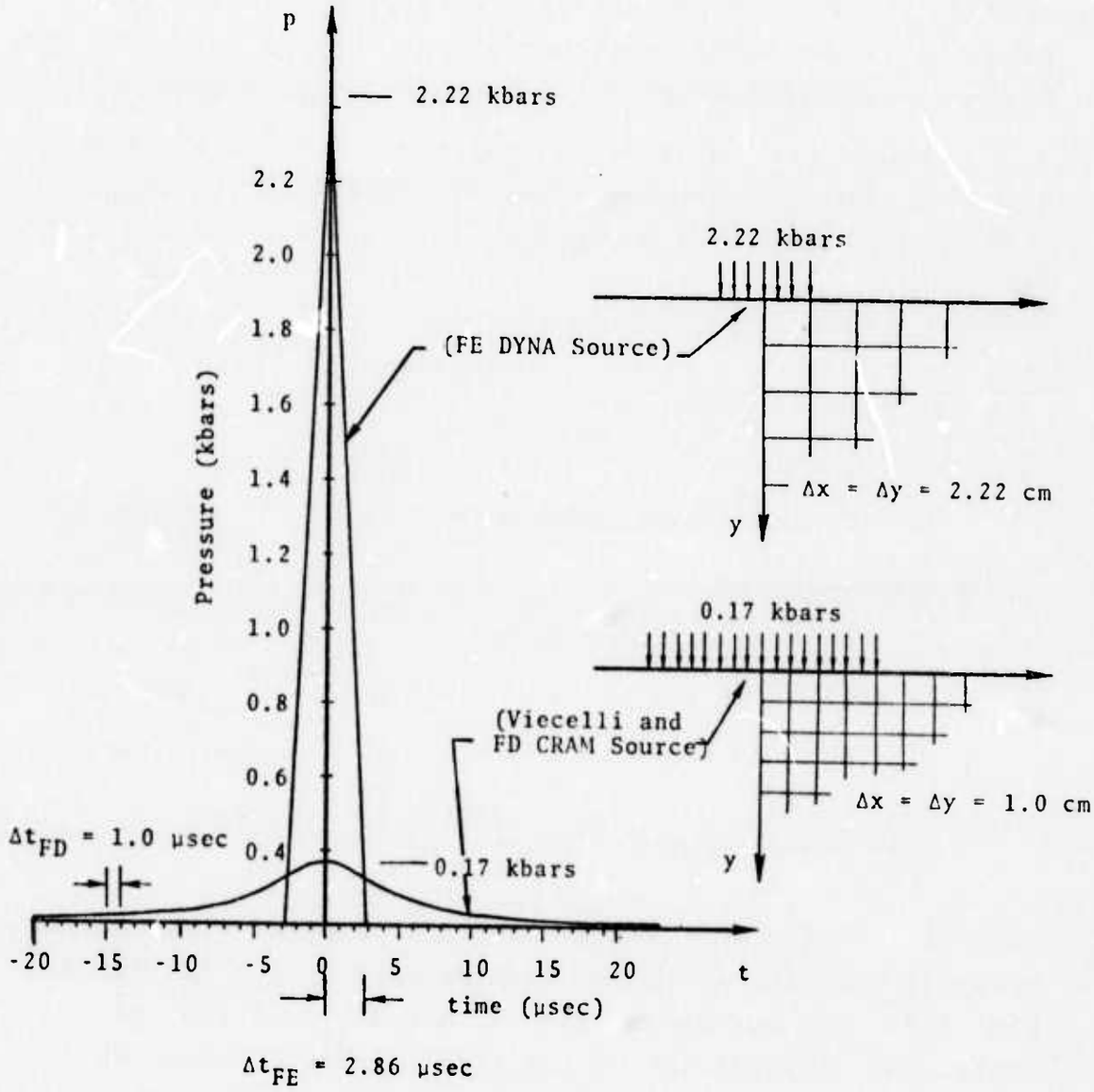


Fig. 3.2--Surface load used in the FE and FD calculations for Lamb's problem.

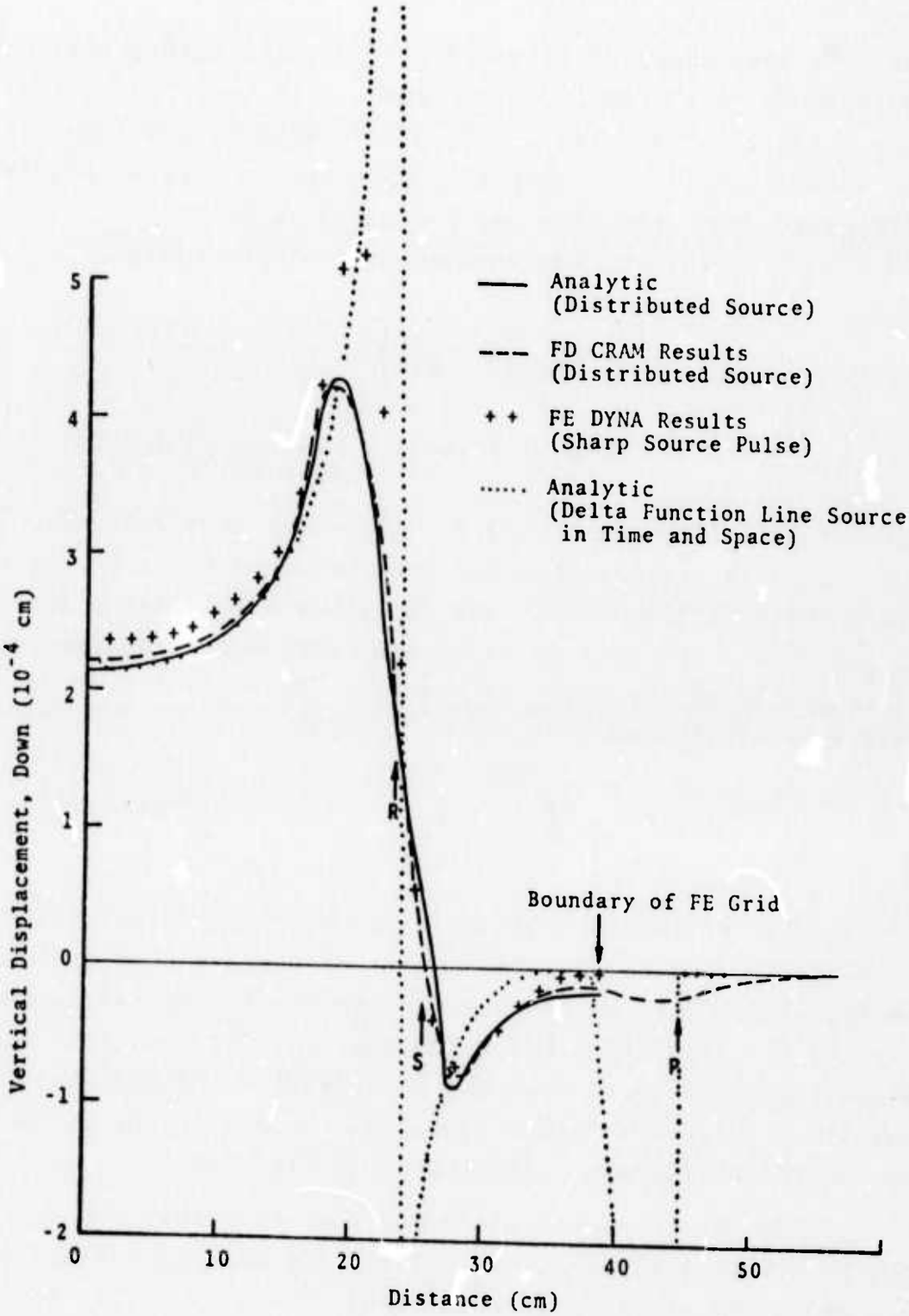


Fig. 3.3--Vertical displacement of the free surface at t = 80 μsec.

that fit into computer core. For S^3's UNIVAC 1108, this corresponds to a grid 18 zones deep and 25 zones along the free surface (S^3 now has a 2-D, 3-D dynamic FE code that is not restricted in this manner). In order to obtain results at 80 μsec, the grid size was increased to $\Delta x = \Delta y = 1.555$ cm, and the strip loading was applied as a single pulse at $t = 0$, i.e.,

$$p(t) = P_{max}\left(1 - \frac{|t|}{\Delta t}\right)$$

$$= 2.22 \times 10^9 \left(1 - \frac{|t|}{2.86\times10^{-6}}\right) \text{ dyn/cm}^2$$

for $-\Delta t \leq t \leq \Delta t$, see Fig. 3.2. The value $\Delta t = 2.86$ μsec is just slightly greater than the time required for a P-wave to cross one grid dimension. The amplitude of the strip loading $P_{max} = 2.22 \times 10^9$ dyn/cm^2 is applied over one grid dimension to each side of the plane of symmetry ($2a = 2\Delta x = 3.11$ cm) to give a total impulse

$$L = 2a \int_{-\Delta t}^{+\Delta t} p(t)dt = 1.97 \times 10^4 \text{ dyn-sec/cm}$$

The vertical displacement of the free surface that results from the FE computations at 80 μsec is also presented in Fig. 3.3. Here we see that the Rayleigh wave displacement for the FE calculation has a somewhat sharper wave form and higher amplitude peak than the corresponding FD and analytic results. This is a direct result of the difference in the two source characters, illustrated in Fig. 3.2.

The displacements and velocities of points throughout the FE grid are illustrated in Figs. 3.4 and 3.5. Results are presented at six points in time: $t = 24.9$, 34.3, 45.7, 57.2, 68.6, and 80.0 μsec. In these plots, we see the development and propagation of the P-wave, the S-wave, and the Rayleigh wave. The various wave phenomena are most effectively

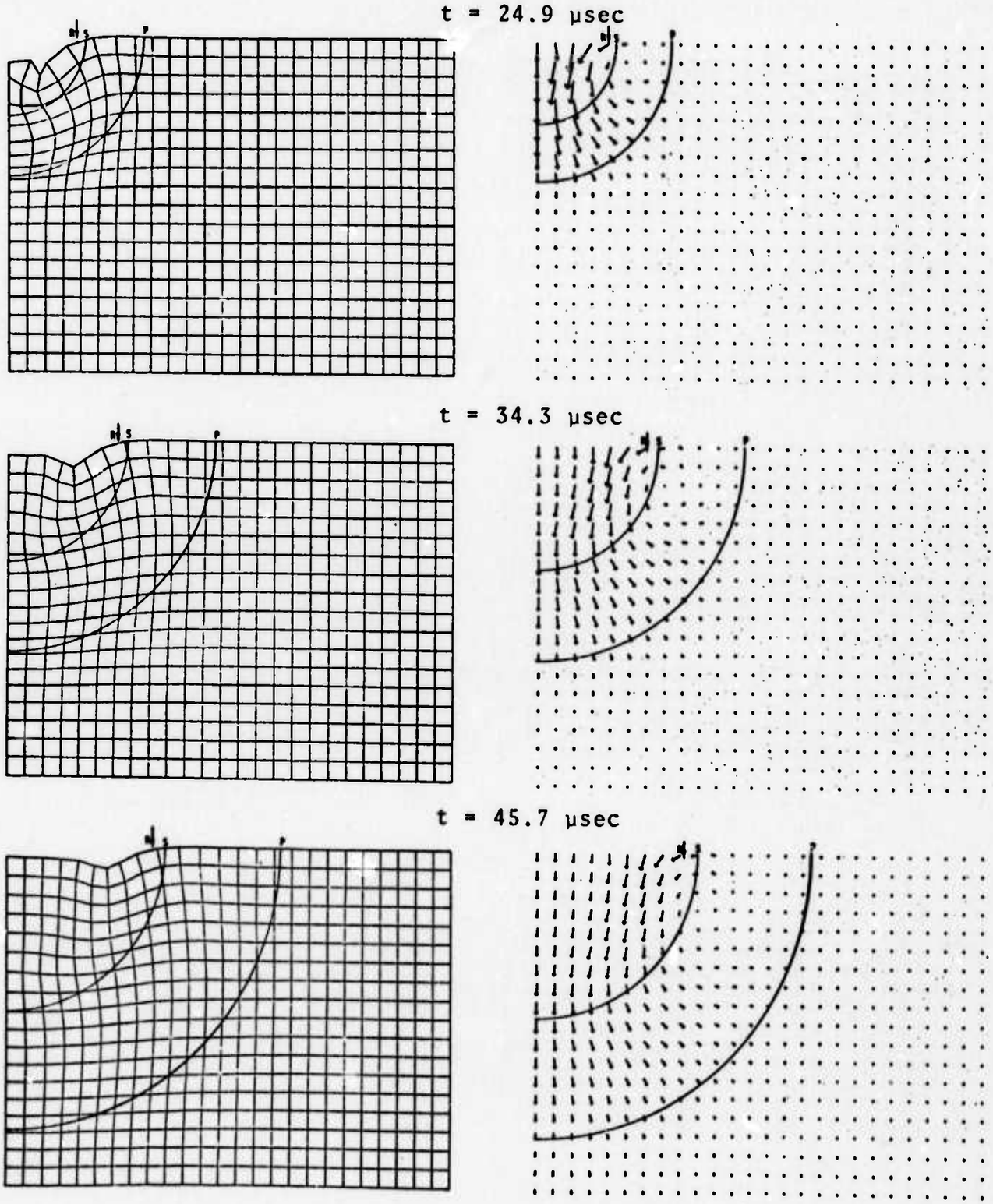


Fig. 3.4a--FE displacements generated by impulse loading applied to the free surface, 2-D Lamb's problem.

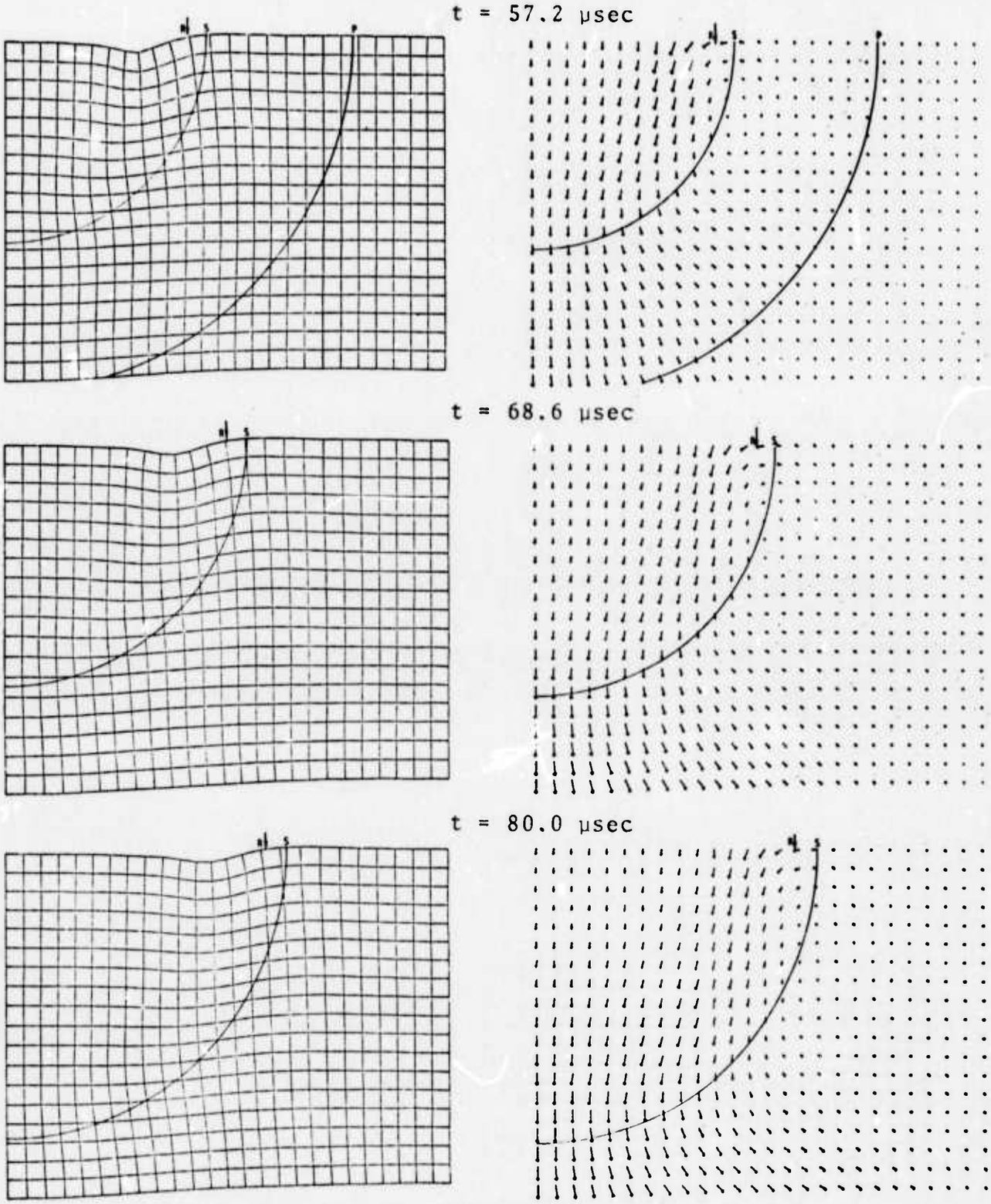


Fig. 3.4b--FE displacements generated by impulse loading applied to the free surface, 2-D Lamb's problem.

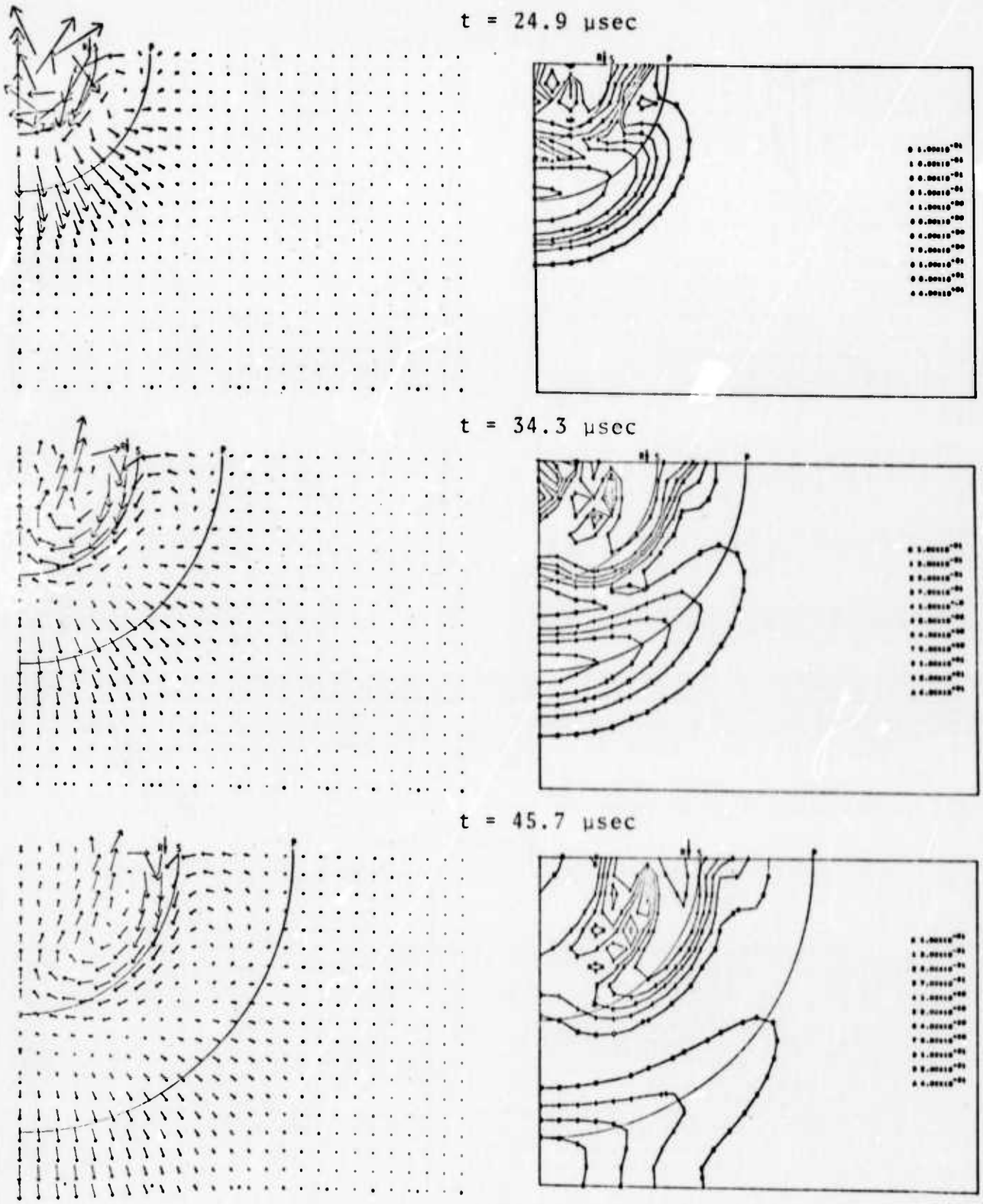


Fig. 3.5a--FE velocities and kinetic energy contours generated by impulse loading applied to the free surface, 2-D Lamb's problem.

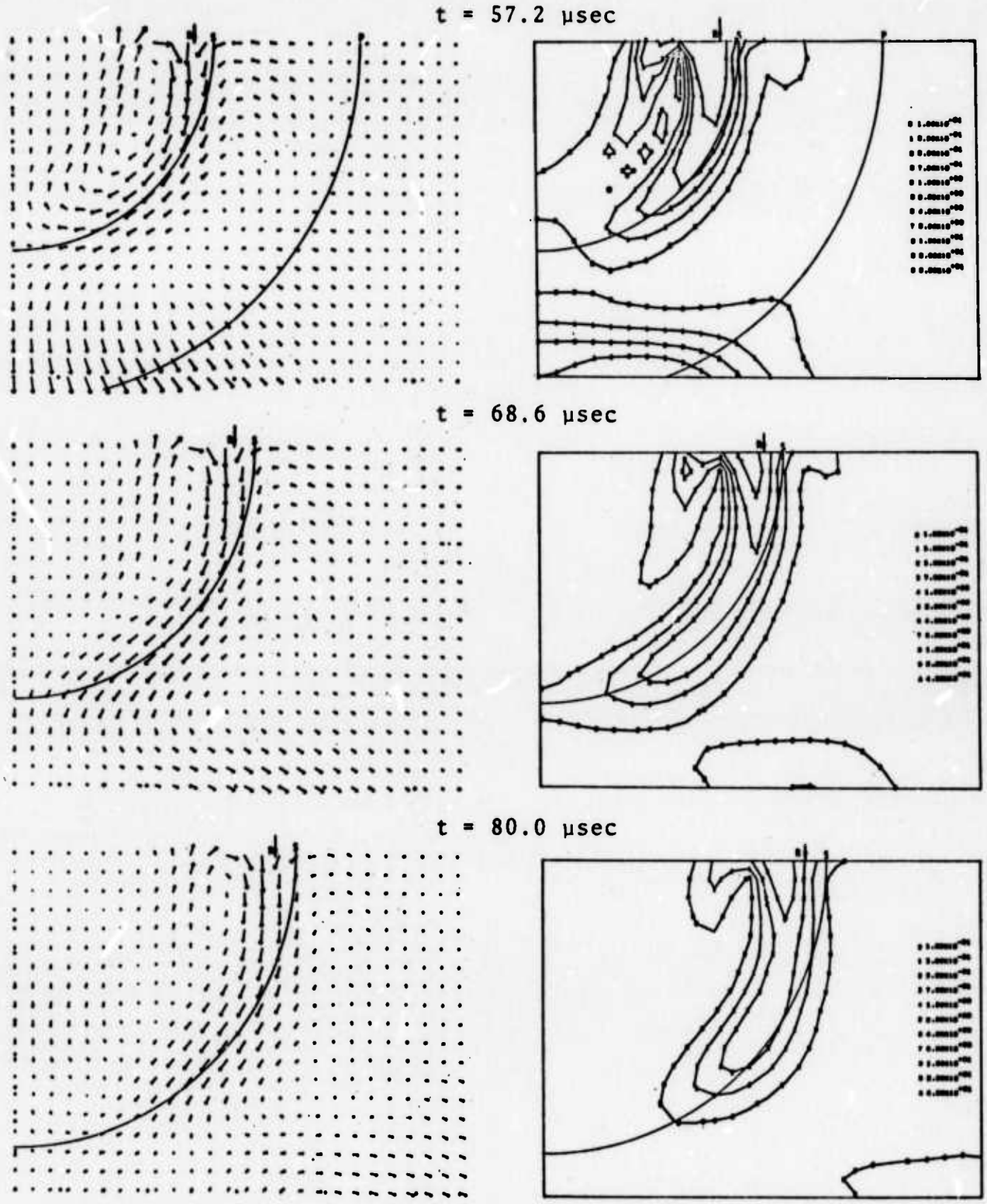


Fig. 3.5b--FE velocities and kinetic energy contours generated by impulse loading applied to the free surface, 2-D Lamb's problem.

depicted in Fig. 3.5, which contains vectorial plots of the velocity field sampled at the node points and contour plots of the kinetic energy. These techniques for presenting wave phenomena should prove valuable for depicting waves in 3-D geometry.

3.3 3-D SWIS CALCULATIONS

A series of plane wave calculations were performed on S^3's UNIVAC 1108 using the SWIS code (Stress Waves In Solids). These serial computer calculations were directed toward the following objectives:

1. To test the utility of the FE technique for performing 3-D wave calculations, particularly waves that propagate a stress jump.
2. To test the explicit time stepping algorithm and the associated artificial damping concept presented in Section 2.4 for speed and accuracy.
3. To test the nonreflecting boundary treatment presented in Section 2.5.

We consider a step planar loading p applied to the free surface ($x_1 = 0$) of a semiinfinite continuum at $t = 0$. Plane waves are generated that propagate in the x_1 direction with a discontinuity in partical velocity and stress at the wave front. Using small displacement theory and linear, homogeneous, and isotropic continuum properties, the propagating wave motion is expressed analytically as

$$u = \begin{cases} \dfrac{p}{(\lambda+2\mu)} (ct-x_1) & \text{for } x_1 \leq ct \\ 0 & \text{for } x_1 > ct \end{cases}$$

and

$$\dot{u} = \begin{cases} \dfrac{cp}{(\lambda+2\mu)} & \text{for } x_1 \leq ct \\ 0 & \text{for } x_1 > ct \end{cases}$$

where $c = V_P$ (P-wave velocity) and $u = u_1$ for normal loading ($p = \sigma_{11}$), and $c = V_S$ (S-wave velocity) and $u = u_2$ for shear loading ($p = -\sigma_{12}$).

The planar waves are modeled numerically using a string 100 uniform, cubic elements in the x_1 direction. To assure $u_2 = u_3 = 0$ for P-wave motions, the boundary nodes (all nodes for this element configuration) are constrained to move only in the x_1 direction. The following parameters were used in the FE calculations:

$V_P = 10^6$ cm/sec

$V_S = 5 \times 10^5$ cm/sec

$\rho = 2.0$ gm/cm^3

$\nu = 1/3$

$\lambda + 2\mu = 2 \times 10^{12}$ dyn/cm^2

$p = 10^8$ dyn/cm^2 = 100 bars

$\Delta x_1 = \Delta x_2 = \Delta x_3 = 1$ cm

$\Delta t = 0.5$ μsec $= 1/2(\Delta x/V_P)$

From this numerical experiment we find that:

1. Spurious high frequency (wave length less than 8 grid dimensions) motions appear in the numerical results with a signal to noise ratio just behind the wave front of about 3 for particle velocity when no artificial damping is used, see Fig. 3.6. The spurious signal does not

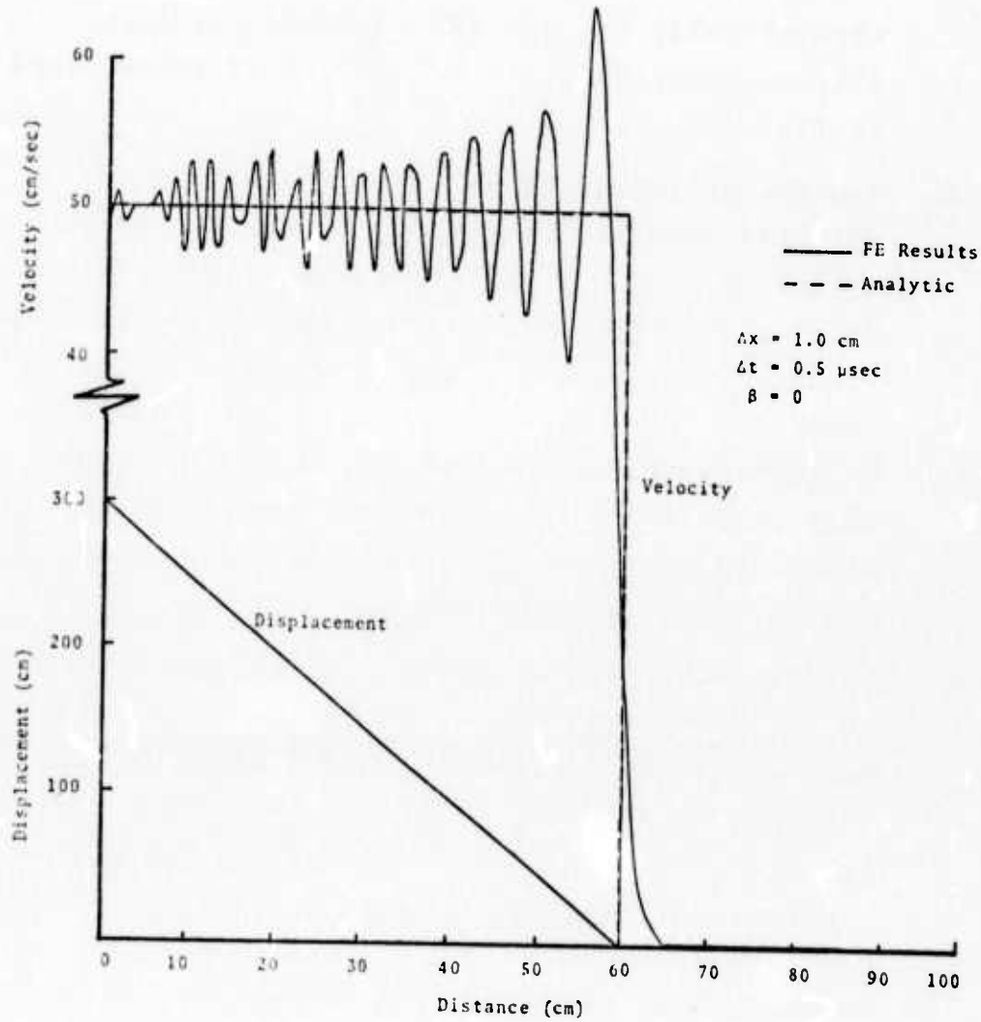


Fig. 3.6--Undamped P-wave at t = 60 μsec.

significantly degrade the computed particle displacements using $\Delta t = \frac{1}{2}\frac{\Delta x}{V_p}$, also illustrated in Fig. 3.6.

2. The discontinuity in particle velocity is suitably modeled numerically using a damping coefficient $\beta/\Delta t = 0.2$, illustrated in Fig. 3.7. Using a damping coefficient $\beta/\Delta t = 0.18$, the wave front is propagated through the grid, which is terminated by a free surface, and part way back to the source, illustrated in Fig. 3.8. From this exercise we note that the steep wave front does not appear to experience excessive deterioration as the wave propagates.

3. The explicit time stepping algorithm developed in Section 2.4 is extremely fast. Waves were propagated through the 404-node chain of 100 elements at the rate of 1/2 sec of UNIVAC 1108 CPU time per time step. Considering the disproportionately large number of constrained displacement components in this test problem, we estimate that waves passing through large 3-D meshes would be processed at only about 1/5 this rate, or 0.006 sec per node per time step. We consider this to be a very good processing rate.

Alternate boundary conditions were tested at the surface $x_1 = 100$ cm. A rigid boundary results in a reflected wave with zero particle velocity behind the reflected wave front. The SWIS calculations resulted in a particle velocity in the reflected wave five orders of magnitude below the incoming particle velocity of 50 cm/sec.

The non-reflecting boundary condition described in Section 2.5 was tested at the boundary surface $x_1 = 100$ cm. Following the incidence of the P-wave with the non-reflecting boundary, $t > 100$ μsec, the computed particle velocity remains

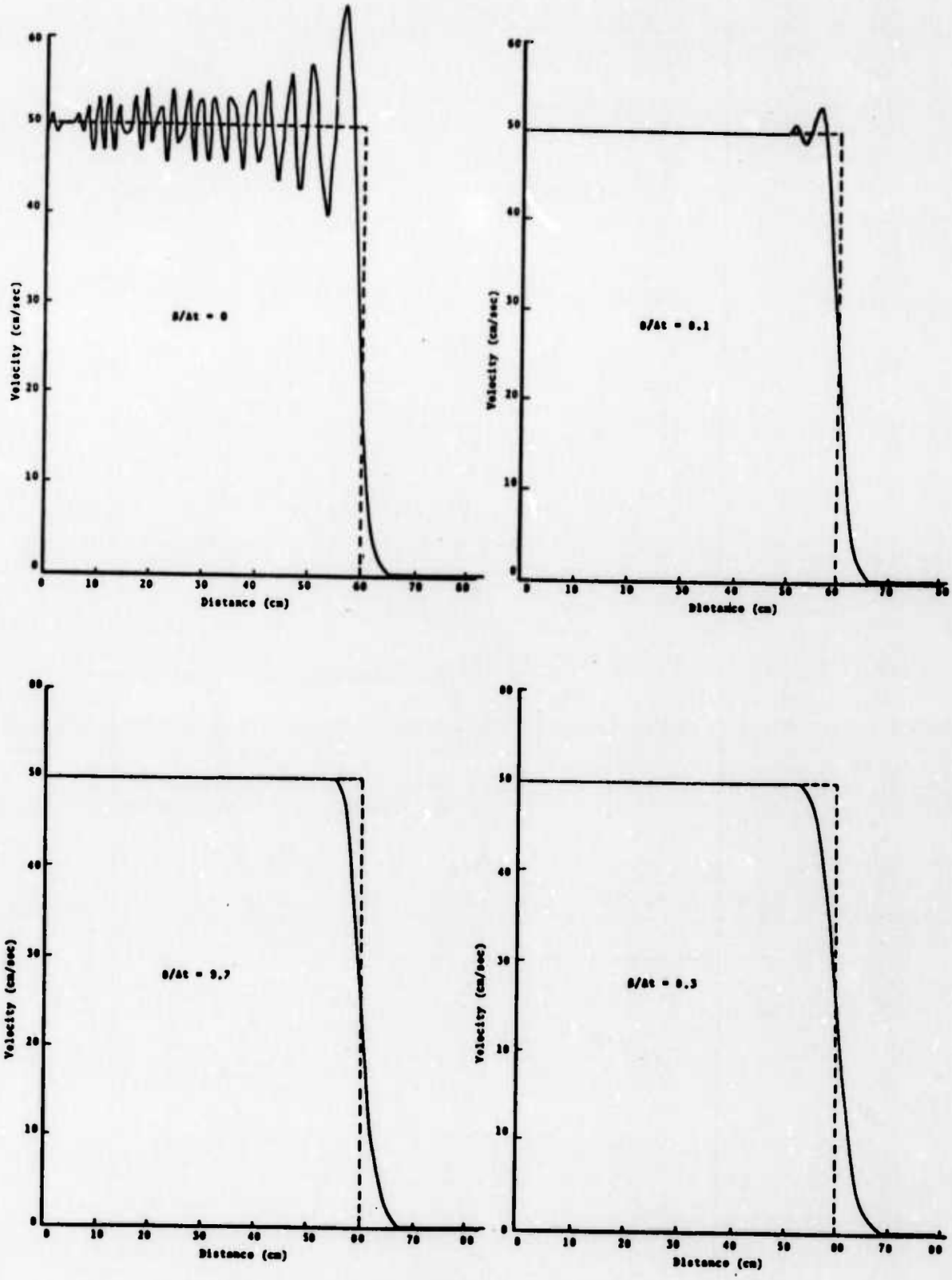


Fig. 3.7--P-wave at t = 60 μsec demonstrating the influence of artificial viscous damping on the computed particle velocity.

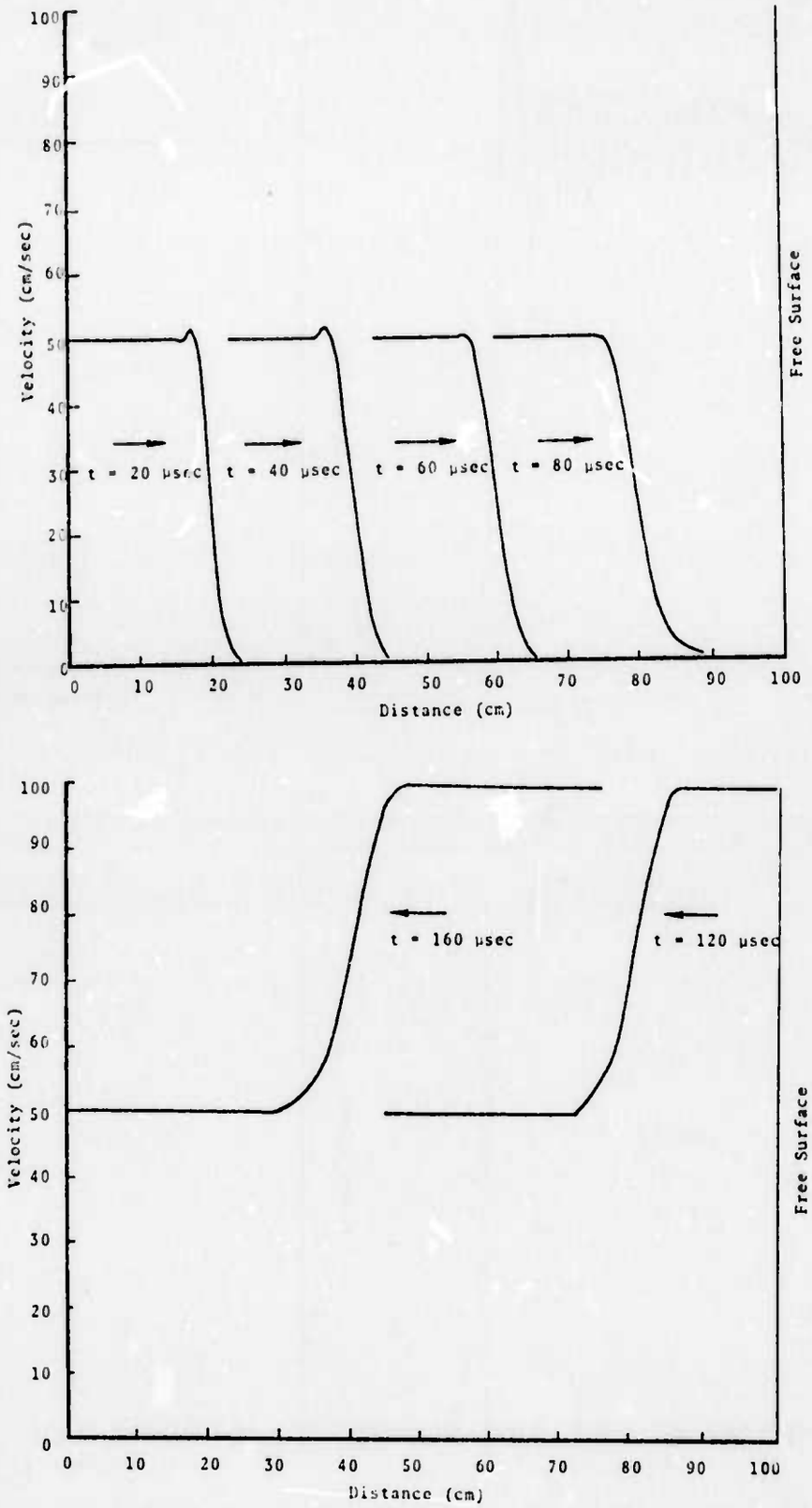


Fig. 3.8--Propagating P wave striking a free surface, damping coefficient $\beta/\Delta t = 0.18$.

at 50 cm/sec throughout the grid, Fig. 3.9. Actually a small ripple in the particle velocity (less than 1 cm/sec) was reflected from the boundary back into the grid which does not show up on the plot.

Calculations were also performed with the planar loading p applied to the surface $x_1 = 0$ in the direction x_2. This load configuration generates shear waves propagating in the x_1 direction. Using the same parameters for the S-wave calculations as for the P-wave calculations, essentially the same accuracy and speed is achieved in the S-wave calculations as in the P-wave calculations. The optimum damping coefficient ($\beta/\Delta t = 0.5$) was found to be somewhat greater than that for the P-wave calculations.

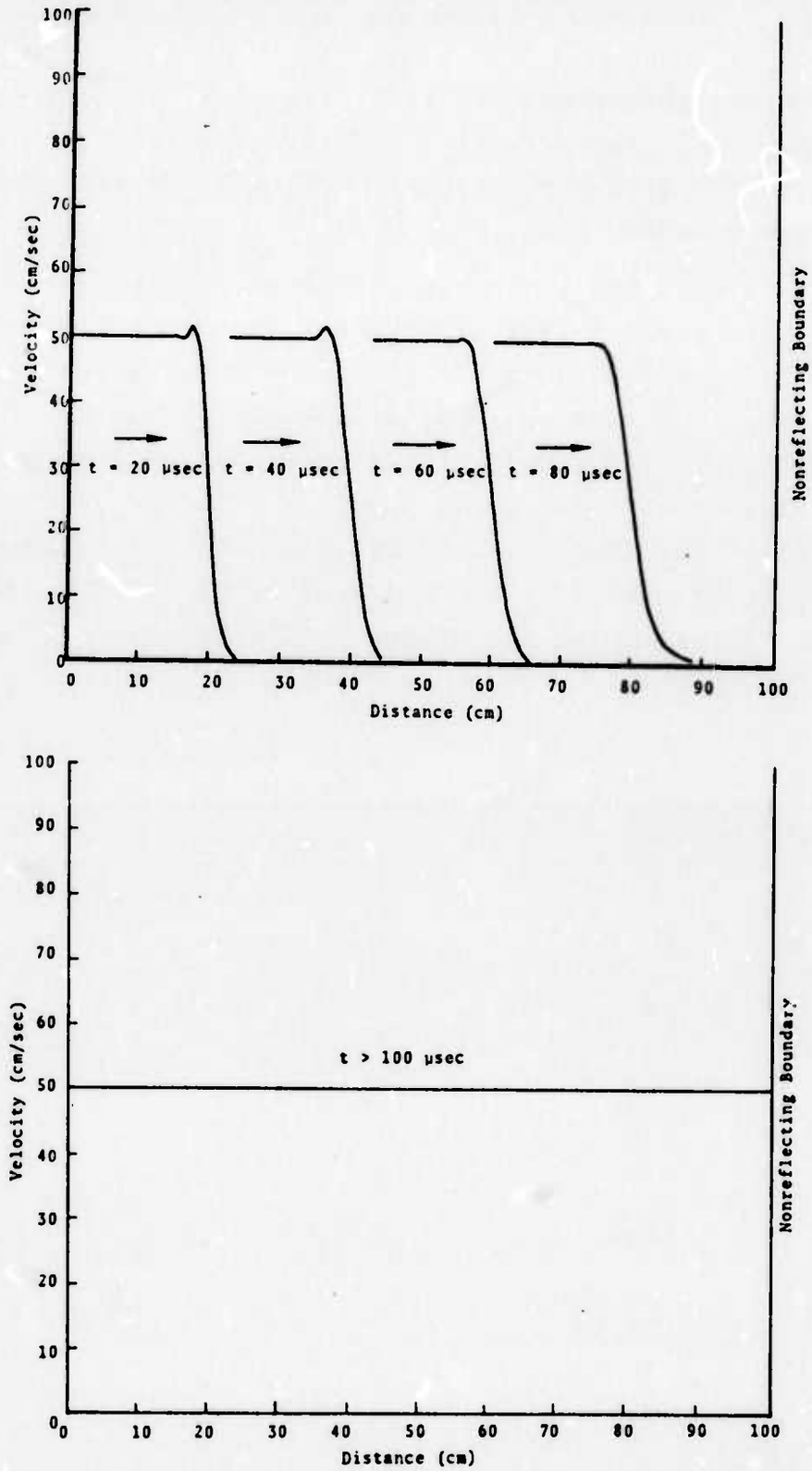


Fig. 3.9--P wave propagating into a nonreflecting boundary, damping coefficient $\beta/\Delta t = 0.18$.

IV. ILLIAC CODE FOR 3-D ELASTIC WAVES

4.1 NUMERICAL PROBLEM DEFINITION AND 3-D GRID GENERATOR

A considerable quantity of data is necessarily required for defining arbitrarily complex 3-D grid systems. In order to remove all restrictions from the node configuration, data must be provided for individually locating each node point in the grid. Since arbitrary node configurations lead to nonsystematic numbering of the nodes associated with an element, connectivity information is required in addition to the node locations. Also, arbitrarily inhomogeneous material properties will need to be specified element by element throughout the grid. In all, a completely arbitrary N-node grid will require about 17N data items. Such a requirement would make 3-D numerical calculations exceedingly tedious to set up.

The massive data requirements can be reduced to a manageable level by generating the grid data in regular ordered portions of the grid and providing detailed grid specifications in unordered portions of the grid. The more versatile the grid generating capability, the less tedious the data preparation.

In preparation for processing large 3-D grid configurations on the Illiac, a technique has been developed for automatic 3-D grid generation. A grid of skewed hexahedral elements is mapped into simple cubic shapes, all elements of identical size. The grid is then generated for the cubic geometry and mapped back into the skewed problem geometry for processing.

The versatility of this technique for generating 3-D grids depends on the type of mappings that can be systematically performed. At this time we have developed three types of mapping: spherical $\leftrightarrow$ cartesian, cylindrical $\leftrightarrow$ cartesian,

and polynomial ↔ cartesian. In each of these three mappings, the zone size can change at a controlled rate in three directions.

Of the three mappings above, the polynomial mapping is the most versatile for generating skewed 3-D grid configurations. The coordinates of twenty points around a mass of continuum are used to define the mapping, eight of the points are to form the exterior corners in the cubic element configuration and the remaining twelve points form the mid edge points in the cubic configuration. A curved grid is generated when the edge points do not lie along a straight line connecting the respective corner points. The grid spacing is reduced in the vicinity of a corner point as the adjacent edge points are moved in the direction of the corner point.

4.2 GENERATION AND STORAGE OF THE DIFFERENCE EQUATIONS

At this time the difference equations are being generated on S^3's Univac 1108 using the 3-D FE SWIS code. The influence coefficients, generated in this manner, become data for the ILLIAC time stepping code. This procedure restricts the ILLIAC code to computing stress waves in materials with linear stress-strain laws, since no provision is made for changing the influence coefficients during the time stepping calculations. This procedure of supplying the parallel processor code with predetermined influence coefficients permits the linear wave propagation calculations to be independent of the grid type. However, the transfer of large numbers of influence coefficients to the ILLIAC site for parallel processing does not appear to be desirable for the final code configuration. In the coming contract year, 1973, code will be developed for generating difference equations on the ILLIAC IV computer.

While early versions of the SWIS code will require that either the grid be fairly regular or else that the coefficients of the difference equations be generated externally on a serial machine, good progress has been made in the development of algorithms for generating coefficients in the ILLIAC for the rather general types of grids that the time stepping algorithm is equipped to handle.

The big problem is not the generation of the coefficients but rather rearranging and storing them where the time-stepper expects to find them. In order to be more specific, it is necessary to have a detailed specification for the storage layout for the variables and constants that occur in Eq. (2.18).

Equation (2.18) has the form

$$\{\underline{U}_{t+\Delta t}\} = \{\underline{V}_t\} + [\underline{\underline{A}}]\{\underline{W}_t\} \tag{4.1}$$

where

$$\{\underline{V}_t\} = \{\underline{P}_t\} - \{\underline{U}_{t-\Delta t}\} + 2\{\underline{U}_t\} \tag{4.2}$$

$$\{\underline{W}_t\} = \left\{\frac{\beta}{2\Delta t}\,\underline{P}_t - \frac{\beta}{\Delta t}\,\underline{U}_{t-\Delta t} + \left(1 + \frac{\beta}{\Delta t}\right) U_t\right\} \tag{4.3}$$

$$[\underline{\underline{A}}] = -\,\Delta t^2 [\underline{\underline{M}}]^{-1} [\underline{\underline{K}}] \tag{4.4}$$

and $\{U_{t+\Delta t}\}$ retains its previously stated significance.

Equation (4.1) can be written

$$\underline{u}_n = \underline{v}_n + \sum_m \underline{\underline{a}}_{nm}\,\underline{w}_m \tag{4.5}$$

where $\underline{u}_n$, $\underline{v}_n$, and $\underline{w}_m$ are elements of the vectors $\{\underline{U}_{t+\Delta t}\}$, $\{\underline{V}_t\}$ and $\{\underline{W}_t\}$ respectively and $\underline{\underline{a}}_{nm}$ is an element of the matrix $[\underline{\underline{A}}]$. The vector elements $\underline{u}_n$, $\underline{v}_n$ and $\underline{w}_m$ are displacement-like vectors (in fact, $\underline{u}_n$ is a displacement, that of node n at time $t + \Delta t$) and they are represented by three

floating-point numbers within the computer. Correspondingly, $\underline{\underline{a}}_{nm}$ is a 3 × 3 influence matrix that contains information about the effect of node m on the displacement of node n. It takes nine numbers to represent $\underline{\underline{a}}_{n,m}$ and it takes nine PEM words to store those numbers.

In the first parallel version of the SWIS code, space is provided in PEM for the two vectors $\{\underline{V}_t\}$ and $\{\underline{W}_t\}$ which are calculated at the beginning of each time step. The matrix $[\underline{\underline{A}}]$, which does not change from step to step, is stored in compressed form on the disk, from which it is read once each time step.

The compression of the A matrix is achieved by omitting all the zero elements. The identity of the non-zero elements is established by storing the subscript pair n, m along with the nine floating point numbers representing $\underline{\underline{a}}_{n,m}$, a strategy that adds only two half-words or one full word to the space required.

To appreciate how much storage is involved, consider a problem in which the number of nodes is N = 10,000. Such a value of N may be too small to give a good representation of many three-dimensional structures but we are limited to such a value of N by the storage requirement for $\{\underline{V}_t\}$ and $\{\underline{W}_t\}$ which are stored simultaneously in the PEM, and each of which consumes 3N words of space.

In a rectangular grid interior nodes have 26 neighbors, not counting themselves, so rows of $[\underline{\underline{A}}]$ corresponding to interior nodes will have only 27 non-zero elements. Rows corresponding to boundary nodes will have even fewer. Thus, in each row the fraction of matrix elements which are non-zero is no greater than 27/N = 0.0027. The total space required to hold the compressed A matrix is: (number of nodes) × (number of neighbors) × (number of words of storage for a matrix element) = $10^4 \times 27 \times 10 = 2.7 \times 10^6$ words when

$N = 10^4$. Thus, it takes $2.7 \times 10^6/15 \times 10^6 = 0.18$ of the disk to hold $[\underline{\underline{A}}]$ when $N = 10^4$.

In all that follows, if m is a positive integer, let $p(m)$ denote the remainder resulting from the division of m by 64. Thus, $0 \leq p(m) < 64$ and 64 divides $p(m)-m$ evenly. Further, let $PEp(m)$ denote PEr where $r = p(m)$.

The notational scheme just defined provides a compact method for describing storage arrangements in ILLIAC. In particular, the element $\underline{w}_m$ of $\{\underline{W}_t\}$ is stored in $PEp(m-1)$; i.e., $\underline{w}_1$ is stored in PE_0, $\underline{w}_2$ in PE_1, $\underline{w}_{64}$ in PE_{63} and $\underline{w}_{65}$ in PE_0. The A matrix is stored so that $\underline{\underline{a}}_{n,m}$ can be read by $PEp(m-1)$, i.e., the PE that holds $\underline{w}_m$, the element to be multiplied by $\underline{\underline{a}}_{n,m}$. The matrix elements are further arranged so that they can be read in increasing sequence on the first node number n.

4.3 SORTING THE A MATRIX

4.3.1 The Sort Problem

It would be inefficient to calculate the elements $\underline{\underline{a}}_{n,m}$ of the matrix $[\underline{\underline{A}}]$ defined in Eq. (4.1) in the order in which they are required for the time step calculation. Therefore, some means must be provided for arranging them on the disk in their proper locations. The content of the present section is a method for laying out the A matrix starting with a file of non-zero matrix elements and their associated subscript pairs in arbitrary sequence. We assume that the file is stored on the disk in the following format:

1. Each page of the file contains 64 16-word records.
2. Each record contains one matrix element $\underline{\underline{a}}_{n,m}$ consisting of 9 numbers and the indicies n and m as well.

3. The records are arranged on the page as if they had been written as 16 64-word rows, 4 records coming from each row and the 16 words of each record coming from 16 consecutive PE's.

For a problem of $N = 10^4$ nodes, as described in Section 4.2, the number of matrix elements will be $27N = 2.7 \times 10^5$. The disk will hold (300 pages/band) × (64 records/page) = 19,200 records/band and therefore the whole compressed matrix will occupy $2.7 \times 10^5/1.9 \times 10^4 = 14.2$ bands. This is more space than was estimated in Section 4.2 because here we take 16 words rather than 10 words for each non-zero matrix element.

In the final configuration, the records are to be arranged on the pages so that when a page is read as 16 64-word rows, one record will enter each PE, the 16 words of the record falling in 16 consecutive rows. Record $\underline{\underline{a}}_{n,m}$ will enter PEp(m-1), i.e., PEp will receive columns $64*\ell+p+1$ of the matrix $[\underline{\underline{A}}]$, where $0 \leq \ell \leq N/64$ and N is the order of $[\underline{\underline{A}}]$. Finally, the records are to be sorted in row-order, i.e., if $n_1 < n_2$ and $p(m_1) = p(m_2)$ then $\underline{\underline{a}}_{n_1,m_1}$ is located ahead of $\underline{\underline{a}}_{n_2,m_2}$. There are usually 27 non-zero elements per row so that a given PE will receive elements from fewer than half the rows. On the other hand, row n could have non-zero elements in columns m and $m+64$ in which case PEp(m) would receive at least two elements of row n.

4.3.2 Brief Description of the Procedures

Having completed these preliminaries, we can describe the strategy underlying the sorting procedure. The first step is to read the unsorted file, which we denote as $F_{0,0}$, and rewrite it in four new files $F_{1,j}$ where $0 \leq j < 4$. The new file $F_{1,j}$ will contain those elements $\underline{\underline{a}}_{n,m}$ for which $16j \leq p(m) < 16j + 16$, $0 \leq j < 4$. Step 2 consists in

copying each file $F_{1,j}$ into four shorter files $F_{2,4j}$, $F_{2,4j+1}$, $F_{2,4j+2}$ and $F_{2,4j+3}$ for all j, $0 \leq j < 4$. File $F_{2,i}$ will contain those elements $\underline{\underline{a}}_{n,m}$ for which $4i \leq p(m) < 4i+4$. Step 3 consists of copying each file $F_{2,j}$ into four still shorter files $F_{3,4j+k}$ where $0 \leq k < 4$ for all j, $0 \leq j < 16$. File $F_{3,p}$ will contain all those elements $\underline{\underline{a}}_{n,m}$ for which $p(m) = p$.

In step 4, each file $F_{3,p}$ from step 3 is read into the PEM, sorted on row index n and written out on file $F_{4,p}$, $0 \leq p < 64$. We assume that all files $F_{3,p}$ have no more than (1024 rows) × (4 matrix elements/row) = 4096 matrix elements. Thus, the matrix can have at most $4096 \times 64 = 2^{18}$ non-zero elements. It follows, also, that $F_{3,p}$ and $F_{4,p}$ have at most 64 pages each so they will each fit in two strips.

The last step, step 5, is unlike the preceding ones. It begins by reading the first page of each of the 64 files $F_{4,p}$ and writing their contents at the first 64 pages of the sorted file F_5. Between reading and writing there occurs a certain transposition of the data. Matrix elements from file $F_{4,p}$ fall into quarter-rows when they are read but they must be rearranged so that they occupy 16 consecutive rows of PEp when they are written on file F_5. A way of performing the transposition is described in some detail in the paragraphs below. After the first 64 pages of F_5 are written, the second page of each $F_{4,p}$ is read, for all p, and these data are transposed and written as pages 64 through 127 of F_5, and so on until F_5 contains all non-zero elements of the matrix A.

4.3.3 A Lower Bound on the Time Required

It is already possible, on the basis of this preliminary description, to estimate the I/O time T_{IO} required to perform all the data transfers between disk and PEM. It would be

of some interest to have a formula for T_{IO} in terms of the order N of the matrix A and the length L of the input file $F_{0,0}$ but for the present we will assume that $F_{0,0}$ has 2^{12} pages and that no file $F_{3,p}$ will have more than 2^7 pages. In addition, we assume just one ECU operates at a time so that reading and writing cannot be overlapped.

Under these assumptions, the first three steps each require that 2^{12} pages be read and written, so step s will take time $T_s = 2^{13}T_p$ where $1 \leq s < 3$ and T_p is the time required to transfer a page.

For step 4, assume that each $F_{3,p}$ can be read in one rotation and that $F_{4,p}$ can be written in one rotation. Then step 4 takes $T_4 = 2^7T_r$ where T_r is the rotation time.

On step 5, assume that it takes one rotation to read the $k^{\underline{th}}$ page of each file $F_{4,p}$ and one rotation to write the $k^{\underline{th}}$ segment of F_5, for $0 \leq k < 2^6$. Then $T_5 = 2^7T_r$. Using $T_p = 133\ \mu s$ and $T_r = 40$ ms, we get

$$\Sigma_s T_s = 3 \times 2^{13}\ T_p + 2 \times 2^7\ T_r = 21.9 \text{ sec}$$

The five steps outlined above fall into three phases. Phase A consists in separating the records into 64 bins, which are the files $F_{3,p}$. Phase A encompasses steps 1, 2, and 3. The records in each bin are arranged in ascending sequence in step 4 which is Phase B. Then the ordered lines are merged in step 5 which is Phase C. This 3-phase strategy has been employed to sort files of medium to large sizes on serial computers.

4.3.4 Phase A – Segmenting the File into Bins

Next, let us turn our attention to the data flow within the PE's. The three steps of Phase A are similar: The records in one file are reclassified into four others.

In the procedure for Phase A, the PE's operate in four groups of 16, group g consisting of PE(16g+h) where $0 \leq h < 16$ and $0 \leq g < 4$. The buffer area occupies 3×64 rows. This space is organized into 12 1-page buffers, 3 in each group of PE's.

The proposed procedure for separating the records into their proper bins is straightforward. Register $R will, at each cycle, hold as many as four records, one in each bin. Missing records are replaced by zeros. At the beginning of a cycle, each bin examines the contents of $R to see if the record there belongs in that bin. If so, that record is replaced by one that belongs in another bin, otherwise the record is left in $R. Then the cycle is completed by routing $R 16 words to the right. (If a record is transferred from $R to a buffer by some bin we say that the bin <u>received</u> the record and if the record was transferred from the buffer to $R we say it was <u>sent</u> from the bin.)

The attentive reader will have thought of some minor complications that are glossed over in the paragraph above. A few of them are addressed in the following list:

1. If a bin can receive the record it finds in $R but has nothing ready to send, it sends a dummy record of all zeros.
2. If a bin finds zeros in $R and has a record to send it simply sends without receiving.
3. A bin that has filled all its buffers with records belonging there just ignores $R until one of its buffers has been transmitted to the disk.

Each step is begun by reading eight pages, two per bin, and proceeds to alternate writing and reading until all pages of the input file have been read. After this, there

will be about eight pages remaining in the PEM to be binned and written out. The sequence of the data will have to determine the order in which the four output files are written.

We propose to keep a map of the files in which each page represented by a bit which is 0 or 1 corresponding to whether it is the image of a vacant page or an occupied page. When the input map is all zeros, input is complete and when output is complete, zeros remaining in the output maps correspond to unused pages in the output files.

We do not have much to say about timing but it is possible to estimate some upper bounds. Note first that the rate at which records fall into bins must average at least 4 per 3 cycles because 4 records are being moved at a time and it takes at most 3 cycles for a record to find its way home. The fact that $R might have a dummy record means either that some record was read into the bin where it belongs or that some bin has processed all its buffers. In the first case, a record reaches the output buffer in one cycle and the latter case arises only when processing gets ahead of I/O, which is of no concern. Neither case demands that we revise our pessimistic estimate of 3 cycles to process 4 records. Therefore $3/4 \times 64 = 48$ cycles per page is the upper limit on the amount of processing required. To keep up with I/O, the cycle time would have to be (266 μs/page) (16 clocks/μs)/(48 cycles/page) = 88 clocks/cycle or about 40 FINST instructions since no floating point is involved. It is hard to imagine using more than 200 instructions per cycle, which yields an upper bound of (3 Steps) × (2.18 sec for I/O) × (5 processing time/I/O time) = 60 sec for steps 1, 2, and 3.

4.3.5 Phase B – The Internal Sort

Step 4, which is the only step in Phase B, has 64 sub-steps, one for each PE. On sub-step 4_p, file $F_{3,p}$ is read in its entirety into the PEM's, sorted and written on file $F_{4,p}$.

The sorting procedure is patterned after an algorithm called quicksort (Hoare, 1961). The procedure begins with the choice of a key called the pivot which is selected so that it will fall near the middle of the file after it is sorted. In our case, the keys have numeric significance and they are fairly uniformly distributed between 0 and the maximum so we would simply take one half of the maximum as the value of the pivot first key. Sampling techniques are used in more general cases.

In either case, records at the beginning of the file with keys that exceed the pivot are interchanged with records near the end of the file with keys that do not exceed the pivot. By working from both ends toward the middle, the file is subdivided into two subfiles, the first of which consists of records with keys that are less than or equal to the pivot and the second of which falls behind the first and consists of records with keys that are greater than the pivot.

The same procedure, called partitioning, is applied to the first subfile to produce two more subfiles, and then the first of those is partitioned, and so forth, until the first subfile consists of just one record, the first. At this point, there will be about $\log_2 M$ subfiles, where M is the original file length. The next stage is not unlike the preceding ones; the first subfile that has length $n > 1$ is partitioned repeatedly, occasionally producing a one-record subfile at the front which is just appended to the ones ahead of it. The strings of records thus produced are in ascending sequence

and eventually constitute the entire file at which point the sort is complete.

The proposed method for implementing quicksort on ILLIAC IV begins by loading $F_{3,p}$ into the PEM. Recall that $F_{3,p}$ has at most 2^{12} records and will occupy no more than 2^{10} rows, or, equivalently, 2^6 pages. The four bins of up to 2^{10} records are first partitioned independently within each bin. One might well find it advantageous to use special tricks for improving PE utilization, but the method for partitioning in each bin could be very similar to that used in serial computers. Having the full 20 bit key stored in each word makes it possible for the 16 PE's in each bin to work as a unit. Between partitionings, records would be routed between bins so that there would be one pivot row in the buffer with the property that all records in lower numbered rows would have keys less than the pivot key and all records in higher numbered rows would have keys greater than the pivot key. Records on a pivot row could have keys falling on either side of the pivot key.

Having partitioned the file once, the subfile of rows numbered lower than the pivot row would be partitioned once again. The first pivot row would be included in the second partitioning because it can hold low keys. The second partitioning would again be followed by routing to produce a second pivot row. As in the serial procedure, the first subfile, i.e., the one with the lowest keys, would be repeatedly partitioned until the first row of the buffer becomes a pivot row. The partitioning process is continued further until every row in the buffer becomes a pivot row. At every stage, partitioning is performed on the first subfile consisting of a set of consecutive rows in the sequence: pivot row, one or more non-pivot row, pivot row. The number of such subfiles at any given point in the process

will seldom exceed 10 if there are 2^{10} rows in the buffer. Thus, little space is required to keep track of the pivot rows.

The file will rarely be in order after being partitioned as described above but permuted elements will always be on consecutive rows so they cannot be separated by more than six intervening records. The remaining unraveling is left to step 5 along with the merging. The pages of file $F_{4,p}$ consist of records from 16 consecutive full-length rows.

4.3.6 Phase C – Reassembling the Pages

Step 5 is started by reading the first page of each of the 64 files and rearranging the records so that those coming from file $F_{4,p}$ fall in PE_p, $0 \leq p < 64$. Supposing that the coding for this operation contains 100 instructions for manipulating one row of information, the time required for the rearranging would be (100 instructions/row) × (1024 rows/buffer load) × (2 clocks/instruction) ÷ (16 clocks/μs) = 12.8×10^3 μs.

There still remains a little sorting to do because the quicksort of step 4 left some records as much as six positions away from their final positions. A variant of the classical binary merge procedure will work well here and only four records, 64 rows, of extra storage are required. The first two steps of the binary merge would be carried out exactly as for a full sort to produce sorted sequences of 4 records. The variation comes in the third and final step. It begins normally with a merge of the first two strings of 4 to produce a string of 8, but then the last 4 records of the string of 8 are merged with the third string of 4 to produce another string of 8. The first half of the preceding string of 8 is stored just behind the first half of the first string of 8 and the second half is then merged with the next string.

The binary merge will take less time than the 13 ms it took to get records in their proper PE's so the whole processing time for step 5 should be less than 25 ms.

4.4 TIME STEPPING

4.4.1 Algorithm

A good design for stepping numerical stress waves along in time is most important for the 3-D SWIS code, since it is this process that will account for essentially all of the execution time. The time stepping process being described is the repeated calculation of Eq. (2.18) taking advantage of the parallel processing capabilities of the ILLIAC IV.

The first three terms of Eq. (2.18) involve column vector operations which are relatively minor. The matrix [M], containing the nodal masses, is diagonal and thus trivial to invert. All these operations are easily adapted to parallel processing. The terms involving β are also easily computed. The column vector resulting from this operation is multiplied by the very large banded sparse matrix $[\underline{\underline{A}}]$ (which is defined in Eq. (4.4)). This multiplication by $[\underline{\underline{A}}]$ will account for nearly all the execution time that is required to complete one numerical time step.

The following scheme for banded sparse matrix multiplies was developed by Frazier (1972). Since the matrix is of order 3×10^4, it is necessary to compress it to a manageable size. We note that in a 3-D gridwork of bricks, most nodes have just 26 immediate neighbors; consequently, for each component (i,j) there will usually be 27 non-zero terms in a single row of $[\underline{\underline{A}}]$. The unnecessary zeros are compressed out of the rows of $[\underline{\underline{A}}]$ to yield a matrix N by 27 (N being the total number of nodes in the grid). From the node numbering sequence we can deduce the column numbers for

the non-zero terms in each row, i.e.,

$$m = m_{n,k} \ , \ n = 1,2, \ldots N$$
$$k = 1,2, \ldots \approx 27 \tag{4.6}$$

where n and m are the row and column numbers of $[\underline{\underline{K}}]$, respectively, and N is the total number of node points. The array of contributing column numbers $m_{n,k}$, $k = 1,2, \ldots \approx 27$ are simply the node numbers adjacent to node n.

Only the non-zero multiplications are performed in the sparse matrix multiplication which is expressed by

$$\underline{b}_n = \sum_{k=1}^{\approx 27} \bar{\underline{\underline{a}}}_{n,k} \, \underline{u}_{m_{n,k}}(t_\alpha) \tag{4.7}$$

for $n = 1,2, \ldots N$ with

$$\bar{\underline{\underline{a}}}_{n,k} = \underline{\underline{a}}_{n,m_{n,k}}$$
$$= - \Delta t^2 \, M_n^{-1} \, \underline{\underline{K}}_{n,m_{n,k}} \tag{4.8}$$

The sparse matrix multiplication is performed at each time step with the sparse matrix remaining unchanged for the special case of linear wave simulation; consequently considerable effort can be devoted to arranging the non-zero terms of the sparse matrix on the mass storage unit in an optimal fashion for processing. The compressed matrix $[\underline{\underline{A}}]$ should be arranged on the disk so that each term arrives in the processor containing the nodal displacement for which it is to be multiplied, Eq. (4.7). Thus, $\bar{\underline{\underline{a}}}_{n,k}$ should arrive in the PEM containing $\underline{u}_{m_{n,k}}(t_\alpha)$ without requiring additional shift operations. The reordering of $[\underline{\underline{A}}]$ is done on a serial

machine independent of the ILLIAC IV in the first implementation of SWIS. For larger problems, the previous section describes how the reordering can be performed by the ILLIAC IV. The following discussion defines the sparse matrix multiply process and the flow of information from disk memory to the array in detail.

The nodal displacements $\underline{u}_m$ are vectors of three components and require three numbers for their representation. Let u_{im} be the component of $\underline{u}_m$ along the x_i axis, $i = 1,2,3$. Correspondingly, $\bar{\underline{\underline{a}}}_{n,k}$ is a 3×3 matrix that requires nine numbers in its representation. Let $\bar{a}_{in,jk}$ denote the ij element of this matrix. Then Eq. (4.7) can be written

$$b_{in} = \sum_{k=1}^{\approx 27} \sum_{j=1}^{3} \bar{a}_{in,jk}\, u_{jm_{n,k}} \tag{4.7'}$$

where b_{in} is the component of $\underline{b}_n$ along the x_i axis.

The nodal displacements at t_a are arranged on the disk to flow into the PEM's (denoted p) by PE rows (denoted r) so that $\underline{u}_{11} \rightarrow p = 0,\ r = 0$; $\underline{u}_{12} \rightarrow p = 0,\ r = 1$; $\underline{u}_{13} \rightarrow p = 0,\ r = 2$; $\underline{u}_{21} \rightarrow p = 1,\ r = 0$; ... $\underline{u}_{(64)3} \rightarrow p = 63,\ r = 2$; $\underline{u}_{(65)3} \rightarrow p = 0,\ r = 3$; etc.

In general we have

$$u_{jm} \rightarrow p(m-1) = (m-1) - 64 \left|\frac{m-1}{64}\right| ,$$
$$r = 3 \left|\frac{m-1}{64}\right| + j-1 \tag{4.9}$$

where $\left|\frac{m-1}{64}\right|$ denotes fixed point division and $p(m)$ is the remainder of $\left|\frac{m}{64}\right|$ as in Section 4.2. This storage configuration in the PE's is achieved by loading $u_{jm}(t_\alpha)$, $m = 1,2, \ldots N$, on the disk in the sequential order $\bar{\bar{u}}_S$, $S = 1,2, \ldots 3N$ where

$$S = m + 128 \frac{m-1}{64} + 64(j-1) \quad , \tag{4.10}$$

$$j = 1,2,3 \quad .$$

The condition for $\bar{a}_{in,jk}$ to arrive in the processor containing $u_{jm_{n,k}}(t_\alpha)$ is expressed, using Eq. (4.9), as

$$\bar{a}_{in,jk} \rightarrow p\ (m_{n,k}-1) \tag{4.11}$$

The PE row number r to be occupied by the various non-zero terms of the sparse matrix is somewhat more difficult to express because of the arbitrariness of the node numbering scheme. The node numbers n should increase monotonically (but not necessarily sequentially) with increasing row number r in each PE so that the row number n of the sparse matrix can be processed in ascending order. However, in general, no more than 27 of the 64 PE's will contain a $u_{jm_{n,k}}(t_\alpha)$ for which $\bar{a}_{in,jk}$ is non-zero for any particular matrix row number n. That is, only about one-third of the PE's will contain nodal displacements that are adjacent to node number n in the spatially zoned continuum. Furthermore, a single PE may, in some instances, contain more than one neighbor nodal displacement but rarely more than nine.

When performing the multiplications that contribute to matrix row number n, there is no need to make the noncontributing PE's inoperative. Each noncontributing PE can simultaneously perform multiplications for the next higher matrix row number for which the PE will have a contribution. If the compressed matrix $[\underline{\underline{A}}]$ is loaded into the PE's in the proper sequence, this work-ahead scheme can be carried out by performing multiplications in each PE in the sequence that the $[\underline{\underline{A}}]$ terms are loaded from mass storage. This desired storage configuration in the various PE's is achieved by loading $\bar{a}_{in,jk}$; $n = 1,2, \ldots N$; $k = 1,2 \ldots \approx 27$, on the disk in the sequential

order $\bar{\bar{a}}_S$, $S = 1,2, \ldots \approx 10 \times 27 \times N$ where

$$S = 64r + (p+1) + 64(3i+j-4) \ , \qquad (4.12)$$

$$i \text{ and } j = 1,2,3 \ ,$$

in which $r = 0, 1,2 \ldots \approx 10 \times 27 \times N/64$ and $p = 0, 1,2 \ldots 63$ are the row and PE number, respectively. The PE number is $p(m_{n,k}-1)$. The row number for each PE is expressed by summing all previous entries in the particular PE, i.e., the row number for PEp is expressed in terms of n and k by

$$r = 10 \sum_{n'=1}^{n-1} \sum_{\beta'=1}^{\approx 27} \delta_{pp'} + 10 \sum_{k'=1}^{k-1} \delta_{pp'} \qquad (4.13)$$

where $\delta_{pp'} = 0$ for $p \neq p'$ and $\delta_{pp'} = 1$ for $p = p'$ and where, just as in Eq. (4.11)

$$p' = p\left(m_{n',k'}^{-1}\right) \ . \qquad (4.14)$$

Every 10th term in the array $\bar{\bar{a}}_S$ starting with $S = 1$ is used to store two index numbers: $m = m_{n,k}$ to identify the nodal displacement that is to be multiplied and n to identify the matrix row to which the multiplication contributes, Eq. (4.7).

Using the storage schemes defined above, i.e., starting from a point in the computations in which $\{\underline{U}\ (t_\alpha)\}$ and $[\underline{\underline{A}}]$ appear in their prescribed sequences on the disk, the sparse matrix multiplication of Eq. (4.7) proceeds as follows:

1. $\{\underline{U}\ (t_\alpha)\}$ is loaded into core by PE rows using a single access.

2. The serially arranged version of $[\underline{\underline{A}}]$, defined Eqs. (4.11) - (4,14), is accessed and its loading is initiated. The terms flow into core by PE rows starting with row 0. After 30 rows are

filled the loading is continued, uninterrupted, back at row 0 overwriting the previously loaded terms. The computations and manipulations of the following steps are carried out before the matrix terms are overwritten.

3. Initialize $r_0 = -10$; $r_2 = 0$; $b_r = 0$ for $r = 0, 1, 2, \ldots 99$; $n_0 = 1$.

4. Three sets of three multiplications and product summation are performed simultaneously in all processors.

$$b_{r_2+i} = b_{r_2+i} + \sum_{j=1}^{3} \bar{\bar{a}}_{r_0+j+3i-3} \, \bar{\bar{u}}_{r_1+j-1}; \quad i = 1,2,3$$

where

$$r_0 = (r_0+10)\left(1-\delta_{20,r_0}\right)$$

$$r_1 = \left\lfloor \frac{m-1}{64} \right\rfloor$$

$$r_2 = \begin{cases} r_2, & \text{if } n = b_{r_2} \\ r_2+4, & \text{if } n > b_{r_2} \end{cases}$$

$$n = \bar{\bar{a}}_{r_0} \quad \text{(first 32 bits)}$$

$$m = \bar{\bar{a}}_{r_0} \quad \text{(second 32 bits)}$$

Also, store matrix row number of the product contribution

$$b_{r_2} = n$$

Set $n_0 = n_0 + 1$

5. Check for the completion of matrix row number n_0. If n_0 = MIN(n) return to step (4); otherwise, i.e., n_0 < MIN(n)) continue on to step (6). MIN(n) is the minimum b_0 among all 64 PE's.

6. Perform a row sum on b_i; i = 1,2,3 among those PE's for which $b_0 = n_0$. With only PEp operative, shift the result to $b_{r_1 + i-1}$, i = 1,2,3 where

$$p = p(n_0 - 1)$$

$$r_1 = \left\lceil \frac{n_0 - 1}{64} \right\rceil$$

Operating only those PE's for which $b_0 = n_0$ set $b_r = b_{r+4}$ for r = 0, 1, ... 99 .

7. If the next ten PE rows of the [A] matrix have been loaded, return to step (4); otherwise return to step (5).

The computations and manipulations of steps (4) - (7) are displayed in Table II.

The time stepping scheme without the damping terms involving β has been implemented in GLYPNIR and tested on the B6700 simulator. Chapter V contains a description of the test problem and results. The following section discusses schemes for outputting and representing the results from the Illiac SWIS code.

4.4.2 Ideal Timing

Since the sparse matrix multiply used in the time stepping scheme is time consuming, some consideration is given to performance of the SWIS code with parts of the logic

TABLE II

COMPUTATIONS AND MANIPULATIONS OF STEPS (4) - (7)

$u_{jm}(t_k)$, $i = 1,2,3;\quad n = 1,2,3 \ldots N$

$\downarrow$

$\bar{\bar{u}}_{pr}$, $p = 0,1,2, \ldots 63$

$\downarrow$

$$= (S-1) - 64 \left\lfloor \frac{S-1}{64} \right\rfloor = p(m-1)$$

$$r = R_0,\ R_0+1,\ R_0+1,\ \ldots\ R_0+3 \left\lfloor \frac{N+63}{64} \right\rfloor$$

$$= R_0 + \left\lfloor \frac{S-1}{64} \right\rfloor = R_0+3 \left\lfloor \frac{m-1}{64} \right\rfloor + i-1$$

$\bar{\bar{\bar{u}}}_S$, $S = 1,2,3, \ldots \approx 3N$

$$= m+128 \left\lfloor \frac{m-1}{64} \right\rfloor + 64(i-1)$$

$a_{in,jm}$, $(n,m) = 1,2,3, \ldots N;\ i = 1,2,3;\ j = 1,2,3;$

$\downarrow$

$\bar{a}_{in,jk}$, $n = 1,2,3 \ldots N;$

$\downarrow$

$k = 1,2,3, \ldots 27$

$m = m_{n,k}$ data

$\bar{\bar{a}}_{pr}$ $r = R_1,\ R_1+1,\ R_1+2,\ \ldots \approx R_1 + 10 \times 27 \times \frac{N}{64}$

$\downarrow$

$$= R_1 + 10 \sum_{n'=1}^{n-1} \sum_{k'=1}^{\approx 27} \delta_{pp'} + 10 \sum_{k'=1}^{k-1} \delta_{pp'}$$

$p = 0,1,2 \ldots 63$

$$= p(m_{n,k}-1)$$

$$p' = p(m_{n',k'}-1)$$

$\bar{\bar{\bar{a}}}_S$ $S = 1,2,3, \ldots \approx$

$$= 64r + (p+1) + 64(3i+j-4)$$

which have been coded in ASK. A preferable manner to implement SWIS would be to code the outer parts of the program in GLYPNIR for ease in maintenance or modification. The critical parts of the sparse matrix multiply in the time stepping section would be coded in ASK to speed up program execution. This discussion will compare estimated timings of various logical sections of SWIS written in ASK with the simulator generated timing for the same section coded in GLYPNIR.

GLYPNIR timing for three logical steps in the code were given in Section 5.2. The only logical step which consumed a non-trivial quantity of time was step 3, involving the partial matrix sums and rowsumming.

The time through step 3-a in GLYPNIR was 44 μs. The time estimate for the same logic in ASK is roughly 11 μs, giving a factor 4 advantage. The GLYPNIR time for a rowsum was 92 μs, while the estimate for the ASK version is 40 μs. Following the computations given in Section 5.2 for a 10^4 node problem:

1. Partial sum time

   $3 \times 11\ \mu s/\text{block} = 33\ \mu s/\text{block}$

2. $7\ \text{nodes/block} \times 40\ \mu s/\text{node} = 280\ \mu s/\text{block}$

3. Total:

$$\frac{(33\mu s + 280\mu s)/\text{block} \times 10^4 \text{nodes}}{7\ \text{nodes/block}} = 0.44\ \text{sec/time step}$$

Thus it appears possible to roughly double the execution speed of the code by programming in ASK.

4.5 COMPUTER RESULTS

If we are to take full advantage of each seismic calculation, large quantities of digital results must be stored for post ILLIAC processing and displaying. Again alluding

to our reference problem involving a 10^4-node grid, we note that five time intervals of full grid output, nodal displacements and velocities, involves 3×10^5 words of digital data or about 10^7 storage bits. Transporting these quantities of data through the ARPA net at 5×10^4 bits/sec, assuming 100 percent occupancy of the net without interrupts, would require about 5 minutes. Of course, the ARPA net communication system would be more effectively used to interrogate the computed output. If the computed results actually need to be shipped out of the Illiac site for post processing and displaying, magnetic tapes would better serve the purpose.

Grid sizes considerably greater than 10^4 nodes are anticipated. Also, stress time histories will be computed in addition to the displacement and velocity fields. Consequently, the more data reduction that can be accomplished at the Illiac site, the better. It appears to us that the UNICON mass storage device has considerable potential as a plot device due to the high resolution that is achieved in writing on the film strips. Optical techniques could be easily employed to enlarge and print the film plots after they left the Illiac site. Such a plot capability would prove an asset to many of the Illiac users.

V. TEST CALCULATIONS USING 3-D ILLIAC CODE

5.1 PROBLEM SET UP AND RESULTS

For the purpose of debugging and testing GLYPNIR coding on the UCSD simulator, a simple rectangular rod is analyzed. A step load is applied axially at one end of the rod as pictured in Fig. 5.1, and the resulting stress wave is allowed to propagate to the end of the rod. This problem is modeled as a four element grid with 20 nodes as shown in Fig. 5.2. The Fortran version of SWIS (run on S^3's UNIVAC 1108) performs the explicit time stepping developed in Section II, Eq. (2.18). With the damping coefficient β set to zero we obtain:

$$\{\underline{U}(t+\Delta t)\} = +\Delta t^2 \, [M]^{-1}\{\underline{F}(t)\} - \{\underline{U}(t-\Delta t)\} + 2\{\underline{u}(t)\} + [\underline{\underline{A}}]\{\underline{U}(t)\} \tag{5.1}$$

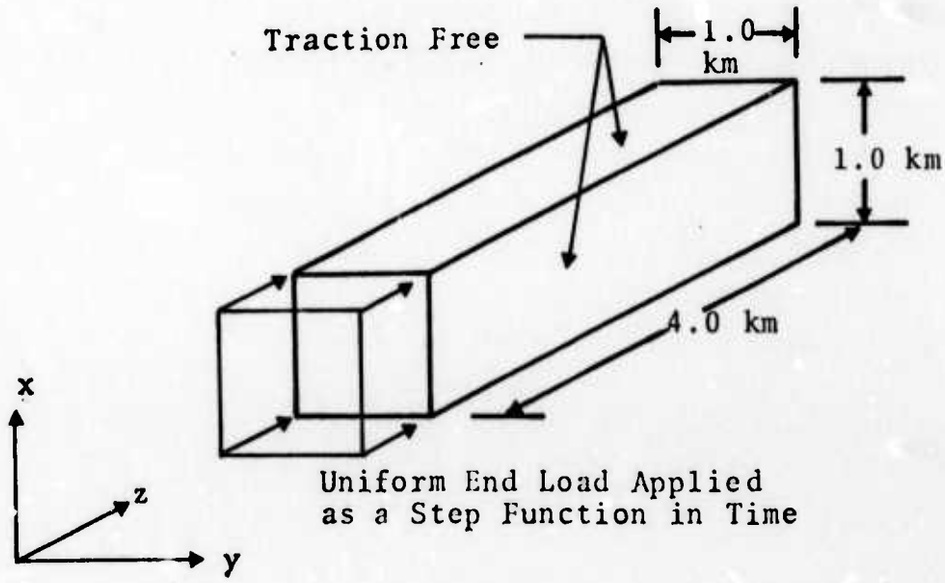
where

$$[\underline{\underline{A}}] = -\Delta t^2 [\underline{\underline{M}}]^{-1}[\underline{\underline{K}}] \tag{5.2}$$

which has been programmed for operation on the ILLIAC IV. In addition, the Fortran code also arranges the nonzero terms of the $[\underline{\underline{A}}]$ matrix according to their desired location in the ILLIAC processors in preparation for the sparse-matrix multiply $[\underline{\underline{A}}]\{\underline{U}(t)\}$ (Eq. 5.1). The arranged terms are then transferred from the UNIVAC 1108 to the UCSD simulator where parallel time stepping computations are performed.

The first simulations ran only one time step to test the sparse-matrix multiply algorithm. Two node numbering schemes were used to confirm the generality of the code. The number of operative processors was reduced to eight for this test problem in order to generate a nontrivial

Preceding page blank

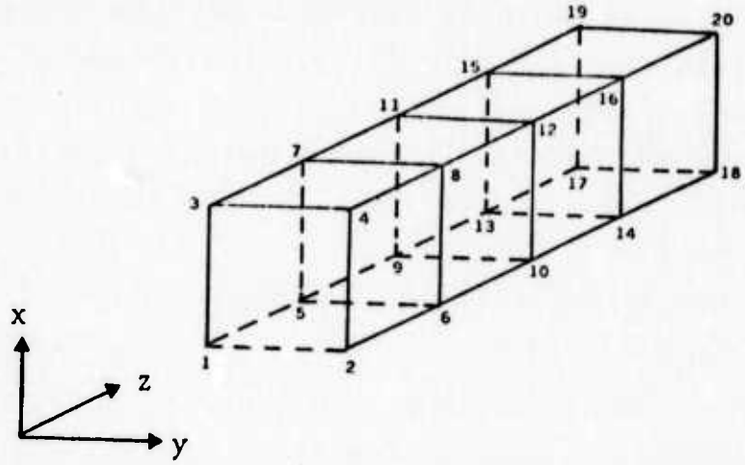


P-wave Velocity = 10.0 km/sec

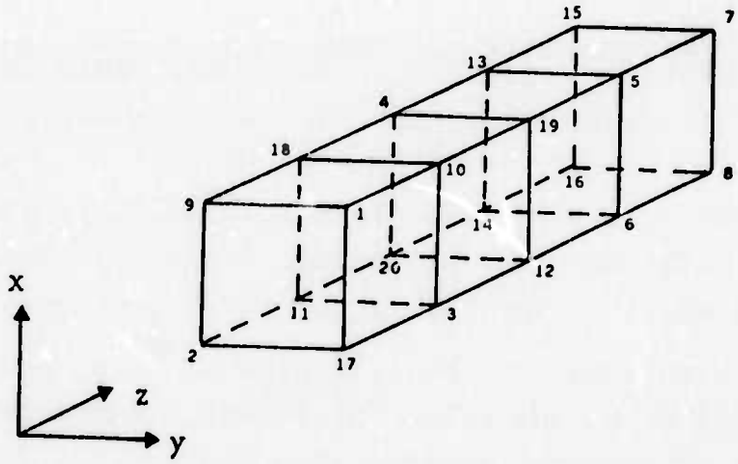
S-wave Velocity = 5.0 km/sec

Density = 2.0 g/cm^3

Fig. 5.1--Test Problem 1: Uniaxial wave propagation in a rectangular rod.



(a) Node Numbering Scheme A



(b) Node Numbering Scheme B

Fig. 5.2--Two node numbering schemes for Test Problem 1: uniaxial wave propagation in a rectangular rod.

simulation with more than one nodal displacement associated with each processor.

As depicted in Table 5.1, the two node numbering schemes result in a markedly different arrangement of the nonzero terms of the $[\underline{\underline{A}}]$ matrix in the eight processors; however, we note that matrix row numbers (n) increase monotonically within each processor. This is to assure the completion of lowest uncompleted row of the sparse matrix multiplication at the earliest sequential step possible within the constraints of the node numbering scheme.

In node numbering scheme A, we note that each of the eight processors contributes to row one of the sparse matrix multiplication, and the processor rowsum for matrix row one is performed directly following the first set of parallel multiplications and accumulations. Whereas for scheme B, only the first three processors contribute to matrix row one. This causes the processor rowsum for matrix row one to be delayed a few steps. However, this results in no loss of efficiency since the remaining five processors are working ahead on matrix row numbers 3, 4, 5 and 6. A loss of efficiency does occur near the completion of the matrix multiplication since some of the processors have finished while other processors are still computing. This effect would be negligible for large problems.

The simulation of the full time stepping process was performed with node numbering scheme B in Table 5.1 on a subset of the problem described above. The simulation was initialized with the wave part way down the rod and run for four time steps.

The results of this run are depicted in Fig. 5.3. Solid lines on the graph indicate the four time steps run on the simulator. The numerical results of the simulation were in exact agreement with the results from the UNIVAC run

TABLE 5.1

ARRANGEMENT OF SPARSE MATRIX INFLUENCE COEFFICIENTS FOR TEST PROBLEM 1 IN AN EIGHT-PROCESSOR SIMULATION

Node Numbering Scheme A

Processor Number	Level	1	2	3	4	5	6	7	8
Displacement Node No. m ($\underline{U}_m$)	0	1	2	3	4	5	6	7	9
	1	9	10	11	12	13	14	15	16
	2	17	18	19	20				
Influence Coefficient Node Nos. n,m ($\underline{A}_{n,m}$)	0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8
	1	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8
	2	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8
	3	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8
	4	5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8
	5	5,9	5,10	5,11	5,12	6,5	6,6	6,7	6,8
	6	6,1	6,2	6,3	6,4	7,5	7,6	7,7	7,8
	7	6,9	6,10	6,11	6,12	8,5	8,6	8,7	8,8
	8	7,1	7,2	7,3	7,4	9,5	9,6	9,7	9,8
	9	7,9	7,10	7,11	7,12	9,13	9,14	9,15	9,16
	10	8,1	8,2	8,3	8,4	10,5	10,6	10,7	10,8
	11	8,9	8,10	8,11	8,12	10,13	10,14	10,15	10,16
	12	9,9	9,10	9,11	9,12	11,5	11,6	11,7	11,8
	13	10,9	10,10	10,11	10,12	11,13	11,14	11,15	11,16
	14	11,9	11,10	11,11	11,12	12,5	12,6	12,7	12,8
	15	12,9	12,10	12,11	12,12	12,13	12,14	12,15	12,16
	16	13,9	13,10	13,11	13,12	13,13	13,14	13,15	13,16
	17	13,17	13,18	13,19	13,20	14,13	14,14	14,15	14,16
	18	14,9	14,10	14,11	14,12	15,13	15,14	15,15	15,16
	19	14,17	14,18	14,19	14,20	16,13	16,14	16,15	16,16
	20	15,9	15,10	15,11	15,12	17,13	17,14	17,15	17,16
	21	15,17	15,18	15,19	15,20	18,13	18,14	18,15	18,16
	22	16,9	16,10	16,11	16,12	19,13	19,14	19,15	19,16
	23	16,17	16,18	16,19	16,20	20,13	20,14	20,15	20,16
	24	17,17	17,18	17,19	17,20				
	25	18,17	18,18	18,19	18,20				
	26	19,17	19,18	19,19	19,20				
	27	20,17	20,18	20,19	20,20				

Node Numbering Scheme B

Processor Number	Level	1	2	3	4	5	6	7	8
Displacement	0	1	2	3	4	5	6	7	8
Node No. m	1	9	10	11	12	13	14	15	16
($\underline{U}_m$)	2	17	18	19	20				
Influence	0	1,1	1,2	1,3	3,4	4,5	4,6	5,7	5,8
Coefficient	1	1,9	1,10	1,11	3,12	4,13	4,14	5,15	5,16
Node Nos.n,m	2	1,17	1,18	2,3	3,20	5,5	5,6	6,7	6,8
($\underline{A}_{nm}$)	3	2,1	2,2	2,11	4,4	5,13	5,14	6,15	6,16
	4	2,9	2,10	3,3	4,12	6,5	6,6	7,7	7,8
	5	2,17	2,18	3,11	4,20	6,13	6,14	7,15	7,16
	6	3,1	3,2	3,19	5,4	7,5	7,6	8,7	8,8
	7	3,9	3,10	4,3	5,12	7,13	7,14	8,15	8,16
	8	3,17	3,18	4,11	5,20	8,5	8,6	13,7	13,8
	9	9,9	4,10	4,19	6,4	8,13	8,14	13,15	13,16
	10	9,1	4,18	5,19	6,12	12,5	12,6	14,7	14,8
	11	9,17	9,2	6,19	6,20	12,13	12,14	14,15	14,16
	12	10,1	9,10	9,3	1[illegible],4	13,13	13,6	15,15	15,8
	13	10,9	9,18	9,11	10,12	13,5	13,14	15,7	15,15
	14	10,17	10,10	10,3	10,20	14,5	14,14	16,7	16,16
	15	11,1	10,2	10,11	11,4	14,13	14,6	16,15	16,8
	16	11,9	10,18	10,19	11,12	15,5	15,6		
	17	11,17	11,2	11,11	11,20	15,13	15,14		
	18	17,17	11,10	11,3	12,12	16,5	16.6		
	19	17,1	11,18	11,19	12,4	16,13	16,14		
	20	17,9	12,10	12,3	12,20	19,5	19,6		
	21	18,1	12,18	12,11	13,4	19,13	19,14		
	22	18,9	17,2	12,19	13,12	20,5	20,6		
	23	18,17	17,10	13,19	13,20	20,13	20,14		
	24		17,18	14,19	14,4				
	25		18,18	17,3	14,12				
	26		18,2	17,11	14,20				
	27		18,10	18,3	18,4				
	28		19,10	18,11	18,12				
	29		19,18	18,19	18,20				
	30		20,10	19,19	19,4				
	31		20,18	19,3	19,12				
	32			19,11	19,20				
	33			20,3	20,20				
	34			20,11	20,4				
	35			20,19	20,12				

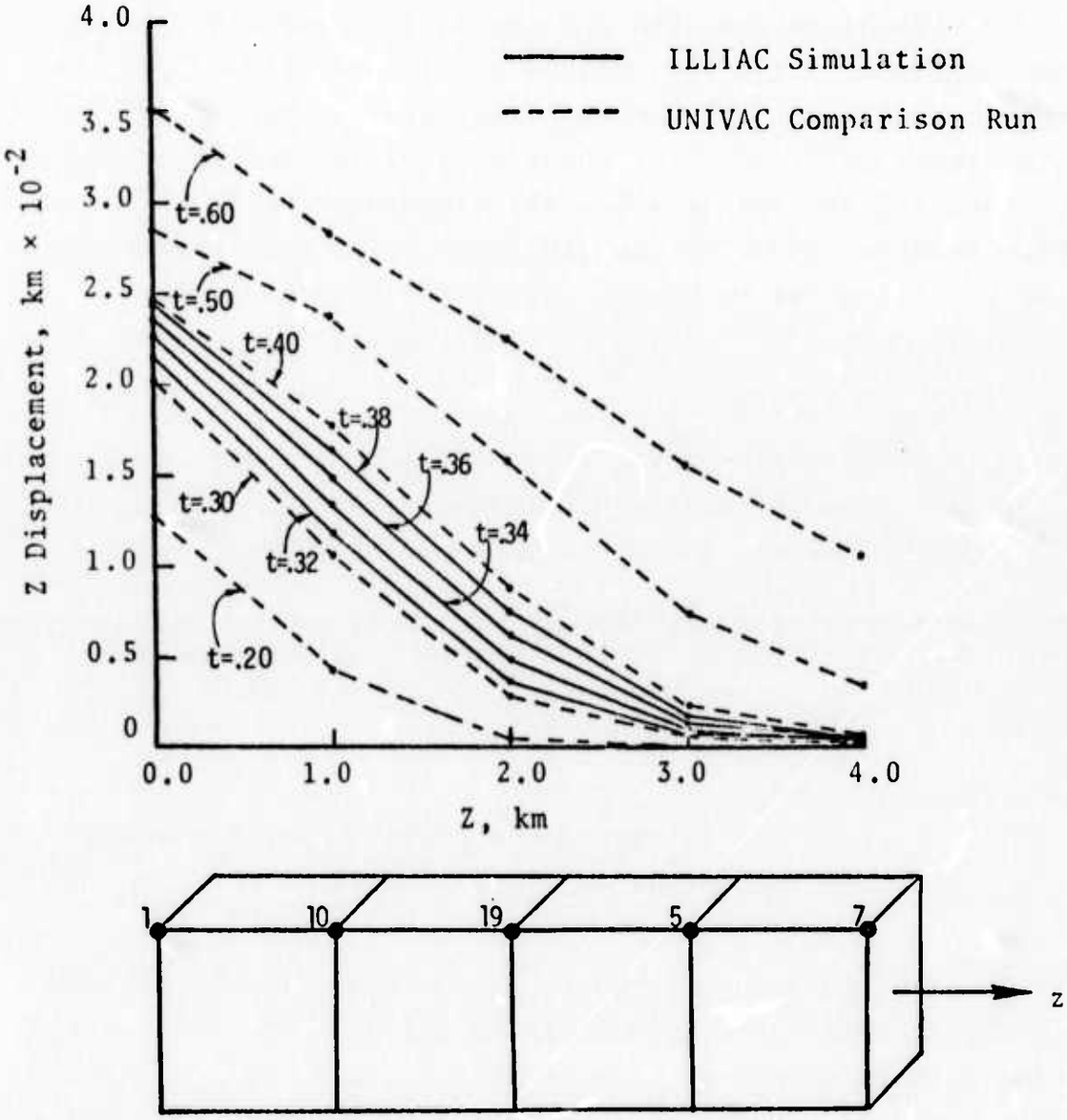


Fig. 5.3--Plot of nodal displacements in Z direction for test problem. The nodes plotted are indicated on the solid.

for the same times. Thus, the curves from the comparison run exactly superimpose the curves of the simulation.

5.2 TIMING RESULTS

Timing information for the GLYPNIR version of SWIS was provided by the simulations on the UCSD B6700. As the sparse-matrix multiply accounts for over 97 percent of the time spent per time step, the timing of the matrix multiply is analyzed in some detail. All times are ILLIAC IV execution times as given by the simulator. The computations for one time step can be broken into three logical sections.

1. Time for loop control and bookkeeping:
   $\sim$ 0.05 ms
2. Time for adding terms of Eq. (5.1) not involving the sparse-matrix multiply:
   0.14 ms
3. Matrix multiply time:
   a. Time to perform the partial sum consisting of a 3 × 3 submatrix of $[\underline{\underline{A}}]$ times three components of $\{\underline{U}(t)\}$:
      44 μs
   b. Time to rowsum the partial sums for a completed row of $[\underline{\underline{A}}] \times \{\underline{U}(t)\}$:
      92 μs

The sparse-matrix multiply performed in the GLYPNIR simulation does a 10 row simulated ILLIAC disk read for each time through step 3-a. The 10 rows contain one 3 × 3 submatrix of $[\underline{\underline{A}}]$ for each PEM. Since the minimum read available on the ILLIAC disk is 16 rows (1 page), a simple extension of the code will be used on the ILLIAC in which two pages are read at one time. One block of two pages (32 rows) just contains the 30 rows necessary for three

3 × 3 submatrices per PEM. This will allow three partial sums (step 3-a) for each disk read. The times given for the partial sums in the matrix multiply may simply be multiplied by three to give the computation time for a two page read.

These figures can now indicate the overall machine time for a typical problem run on the GLYPNIR code. Consider a grid containing 10^4 nodes. This is a large problem for common serial machines, but is small enough to contain all nodal displacements $\{\underline{U}(t)\}$ and $\{\underline{U}(t+\Delta t)\}$ in the PEM's (6 nodal displacements/node).

The influence coefficients $[\underline{\underline{A}}]$ for seven nodes will fit into a two page block read from disk:

$$\frac{(64 \text{ PE's}) \times (3\ 3\times3 \text{ matrices/block/PE})}{(27 \text{ neighbors/node})} = \frac{64 \times 3}{27} = 7$$

The total sparse-matrix multiply time for a 10^4 node problem for one time step is then:

1. Partial sum time:
   3 × 44 μs/block = 132 μs/block
2. Time for summing partial sums:
   7 nodes/block × 92 μs/node = 644 μs/block
3. Total:
   $$\frac{(132\ \mu s + 644\ \mu s)/\text{block} \times 10^4 \text{ nodes}}{7 \text{ nodes/block}} = 1.1 \text{ sec/time step}$$
   (the execution time for other equation terms and o erhead are negligible)

The time given is for GLYPNIR implementation of the code without consideration of the time taken for I/O. Using a double buffering scheme, the only extra time consumed would be execution of code for I/O control and bookkeeping. The additional code should not increase the estimated time beyond

two seconds/time step for the 10^4 node problem. It is also possible to speed up the code by implementing the inner loop of the matrix multiply in ASK.

VI. SUMMARY AND CONCLUSIONS

A long-standing controversy between FE and FD methods has been partially resolved. Both test calculations and theoretical comparisons bear evidence that the two numerical methods are very similar in nature. The spatial difference coefficients for the simplest FD scheme are identical to those of the FE method using tetrahedral elements in a uniform rectalinear grid. The difference coefficients obtained using trilinear FE bricks are considered to be superior to those obtained from the cell-centered-stress FD method because of an instability that exists in this FD scheme. The difference coefficients for the FE brick reduce to those for the cell-centered-stress FD method when a one-point integration procedure (implying uniform strain energy density within an element) is employed in evaluating the element stiffness matrix for the FE brick. And, conversely, a FD method is described that results in the difference coefficients for the FE brick using exact integration for the element stiffness matrix.

The major difference in conventional FE and FD computer codes is in the processing of the numerical difference equations. We are adopting an explicit time stepping procedure, which is generally associated with FD codes and a type of artificial damping, which is almost exclusively associated with FE codes. Also, we have developed and tested a simple non-reflecting boundary condition which is equally applicable to both FE and FD codes.

Test calculations have been performed on S^3's UNIVAC 1108 to compare existing FE and FD codes. Comparisons were made for two problems: a spherically symmetric explosion with an exponentially decaying step pressure and Lamb's problem--an impulse loading applied to the free surface of

a homogeneous half space. Neither method displayed noteworthy superiority on the two problems presented. Further comparisons may uncover advantages of one method over the other for special problem conditions.

Several machine computations have also been performed using the SWIS code to test certain aspects of the 3-D time stepping algorithm such as accuracy, artificial damping, computing economy, and the nonreflecting boundary condition. Very satisfactory results were obtained for the case of a step loading applied to a chain of 100 brick elements.

A parallel processing time stepping algorithm for propagating elastic waves has been developed and tested on the UCSD simulator. A stress wave generated by a step loading applied to a four-element chain of brick elements was propagated part way down the element chain using S^3's UNIVAC computer. At an intermediate point in the serial machine calculations, the stress wave calculations were transferred to the parallel processing algorithm where four time steps were performed. The results from the parallel algorithm were compared with the serial machine results to verify the GLYPNIR code.

Other major accomplishments that were completed during the period April 1972 through December 1972 and reported on in this report include: the development of a 3-D grid generator, the preliminary development of 3-D plot capability, and the development of sparse matrix multiply and sort algorithms for processing on the ILLIAC computer.

The adaptation of the ILLIAC IV system for processing elastic wave calculations is just beginning. Based on timing estimates, the ILLIAC code is expected to be extremely fast, requiring about 0.4 seconds per time step for a 10^4-node grid. In addition, there are virtually no geometric restrictions of the type that would preclude local mesh refinements or the treatment of odd-shaped solids.

Further development is needed to extend the elastic stress wave code for treating nonlinear materials. Also, imaginative graphical techniques need to be employed for displaying results from stress wave calculations.

VII. REFERENCES

Lamb, H., "On The Propagation of Tremors Over the Surface of an Elastic Solid", Phil. Trans. Roy. Soc. (London) A, Vol. 203, pp. 1-42, 1904.

Ewing, W. M., and W. S. Jardetzky, F. Press, Elastic Waves in Layered Media, McGraw-Hill, 1957.

Washizu, K., Variational Methods in Elasticity and Plasticity, Pergamon Press, 1968.

Frazier, G. A., "Vibrational Characteristics of Solids with Applications to Earth Dams", Ph.D. Thesis, Civil Engineering Department, Montana State University, Bozeman, Montana, August 1969.

Frazier, G. A., "A 3-D Code to Simulate Stress Waves in Solids", Proceedings of the ARPA/NASA ILLIAC Symposium, Monterey, California, March 1972.

Blake, F. G., "Spherical Wave Propagation in Solid Media", Jour. Acoust. Soc. Am., Vol. 24, pp. 211-215, 1952.

Cherry, J. T., C. B. Archambeau, G. A. Frazier, A. J. Good, K. G. Hamilton, and D. G. Harkrider, "The Teleseismic Radiation Field From Explosions: Dependence of Seismic Amplitudes upon Properties in the Source Region", Systems, Science and Software Report SSS-R-72-1193, DASA 01-71-C-0156, July 1972.

Wilson, E. L., "Elastic Dynamic Response of Axisymmetric Structures", report to Waterways Experiment Station, U. S. Army Corps of Engineers, Report No. 69-2, Structural Engineering Laboratory, University of California, Berkeley, California, January 1969.

Newmark, N. M., "A Method of Computation for Structural Dynamics", Proc. ASCE 85, EM3, July 1959.

Goudreau, G. L., "Evaluation of Discrete Methods for Linear Dynamic Response of Elastic and Viscoelastic Solids", Report No. 69-15, Structural Engineering Laboratory, University of California, Berkeley, California, June 1970.

Halda, E. J., and J. T. Cherry, "Numerical Simulation of Stress Wave Propagation in Two Space Dimensions", Transactions, Amer. Geo. Union, Vol. 53, No. 11, November 1972.

Preceding page blank

APPENDIX A

FINITE ELEMENT FORMULATION

A.1 VIRTUAL WORK

Conservation of momentum for an arbitrarily nonlinear material is expressed at an instant in time by the virtual work expression

$$\int_V \left(\rho \frac{d\dot{u}_i}{dt} \delta u_i + \sigma_{ij} \frac{\partial \delta u_i}{\partial x_j} - \bar{f}_i \delta u_i \right) dV - \int_{S_\sigma} \bar{\tau}_i \, \delta u_i \, ds = 0 \tag{A.1}$$

in which δu_i is a virtual displacement, $\bar{f}_i$ is a specified body force, and $\bar{\tau}_i$ is a specified traction applied to the surface S_σ. This is the energy principle used to develop nonlinear procedures by the FE displacement method. If we restrict ourselves to the small displacement theory of linear elasticity, the virtual expression reduces to

$$\int_V \left(\rho \ddot{u}_1 \delta u_i + E_{ikjl} \frac{\partial u_j}{\partial x_l} \frac{\partial \delta u_i}{\partial x_k} - \bar{f}_i \, \delta u_i \right) dv - \int_{S_\sigma} \bar{\tau}_i \, \delta u_i \, ds = 0 \tag{A.2}$$

where E_{ikjl} contains the elastic modulii. For isotropic materials, we have

$$E_{ikjl} = \mu \, \delta_{ij} \delta_{kl} + \mu \, \delta_{il} \delta_{kj} + \lambda \, \delta_{ik} \delta_{jl}$$

A.2 FINITE ELEMENT SPATIAL REDUCTION

A single approximation is employed in conventional finite element procedures regarding the behavior of the dependent variable within the region of each element. In this development we approximate the displacement field in an element by a

Preceding page blank

strains,

$$\varepsilon_{ij}(\underset{\sim}{x},t) = \frac{1}{2}\left\langle \frac{\partial \phi^e}{\partial x_j}(\underset{\sim}{x}) \right\rangle \left\{ U_i^e(t) \right\} + \frac{1}{2}\left\langle \frac{\partial \phi^e}{\partial x_i}(\underset{\sim}{x}) \right\rangle \left\{ U_j^e(t) \right\} \quad (A.8)$$

and stresses,

$$\sigma_{ij}(\underset{\sim}{x},t) = E_{ijkl} \frac{\left\langle \partial \phi^e(\underset{\sim}{x}) \right\rangle}{\partial x_l} \left\{ U_k^e(t) \right\} \quad . \quad (A.9)$$

Virtual fields are also expressed in terms of the polynomial interpolation field. For example, the virtual displacement field appearing in Eq. (A.2) is simply denoted

$$\delta u_i(\underset{\sim}{x},t) = \left\langle \phi^e(\underset{\sim}{x}) \right\rangle \left\{ \delta U_i^e(t) \right\} \quad . \quad (A.10)$$

In order to assure continuity in the displacement field across the interface of adjacent elements, the interpolation functions must satisfy the condition that displacements along that portion of an element boundary common to an adjacent element only depend on the nodal displacements located on that boundary. That is, a nodal displacement must have no influence on the surface displacements at the opposite end of the element. When this requirement is fulfilled the displacement field on an element surface, as deduced from Eq. (A.4), becomes

$$u_i(\underset{\sim}{x},t) = \sum_{n=1}^{N^b} \phi_n^b(\underset{\sim}{x})\, U_{in}^b(t) \quad (A.11)$$

or in matrix form

$$u_i(\underset{\sim}{x},t) = \left\langle \phi^b(\underset{\sim}{x}) \right\rangle \left\{ U_i^b(t) \right\} \quad (A.12)$$

for $\underset{\sim}{x}$ on element boundary b. The following section, "Isoparametric Elements", describes how this requirement is fulfilled, even when using curved elements.

The polynomial approximation to the element displacement field is incorporated into the governing virtual work equation, Eq. (A.2), to yield

$$\sum_{e=1}^{E} \left\{\delta U_i^e\right\}^T \int_{V^e} \left\langle \phi^e \right\rangle^T \rho \left\langle \phi^e \right\rangle dV \left\{\ddot{U}_i^e\right\} =$$

$$+ \sum_{e=1}^{E} \left\{\delta U_i^e\right\}^T \int_{V^e} \left\langle \phi^e \right\rangle^T \bar{f}_i \, dV$$

$$- \sum_{e=1}^{E} \left\{\delta U_i^e\right\}^T \int_{V^e} \frac{\langle \partial \phi^e \rangle}{\partial x_j}^T E_{ijkl} \frac{\langle \partial \phi^e \rangle}{\partial x_l} dV \left\{U_k^e\right\}$$

$$+ \sum_{b=1}^{B} \left\{\delta U_i^b\right\}^T \int_{S_\sigma^b} \left\langle \phi^b \right\rangle^T \bar{\tau}_i \, dS \tag{A.13}$$

in which volume integrals are carried out element by element, E denoting the total number of elements, and similarly the surface integral is carried out surface by surface, B denoting the total number of element surfaces on S_σ.

The elements are now assembled by joining the nodes common to various elements. Symbolically this can be accomplished by expressing element and boundary nodes, in terms of the global system, i.e.,

$$\left\{U_i^e\right\} = [T^e] \left\{U_i\right\} \tag{A.14}$$

where U_i is a column listing of all of the nodal displacements that appear in the entire continuum. Each row of $[T^e]$ has just one nonzero term which is unity. The term is located to pick out the node from the global system that corresponds to the element node associated with that row. Similarly, the

transformation from the global numbering scheme to the surface numbering scheme is expressed

$$\left\{U_i^b\right\} = [T^b]\left\{U_i\right\} \tag{A.15}$$

Equation (A.13) is then expressed globally using Eqs. (A.14) and (A.15) to obtain

$$\left\{\delta U_i\right\}^T \left([M]\left\{\ddot{U}_i\right\} + [K_{ij}]\left\{U_j\right\} = \left\{F_i\right\}\right) \tag{A.16}$$

where

$$[M] = \sum_{e=1}^{E} [T^e]^T [M^e] [T^e]$$

$$[M^e] = \int_{V^e} \left\langle \phi^e \right\rangle^T \rho \left\langle \phi^e \right\rangle \, dV \tag{A.17}$$

$$[K_{ij}] = \sum_{e=1}^{E} [T^e]^T [K_{ij}^e] [T^e] \, dV$$

$$[K_{ij}^e] = \int_{V^e} \frac{\langle \partial \phi^e \rangle}{\partial x_k}^T E_{ikjl} \frac{\langle \partial \phi^e \rangle}{\partial x_l} \, dV \tag{A.18}$$

$$\{F_i\} = \sum_{e=1}^{E} [T^e]^T \int_{V^e} \left\langle \phi^e \right\rangle^T \bar{f}_i \, dV$$

$$+ \sum_{b=1}^{B} [T^b]^T \int_{S_\sigma^b} \left\langle \phi^b \right\rangle^T \bar{\tau}_i \, dS \tag{A.19}$$

Since the virtual displacements are arbitrary, the discretized version of the virtual work expression, Eq. (A.16)

reduces to

$$[M]\{\ddot{U}_i(t)\} + [K_{ij}]\{U_j(t)\} = \{F_i(t)\} \tag{A.20}$$

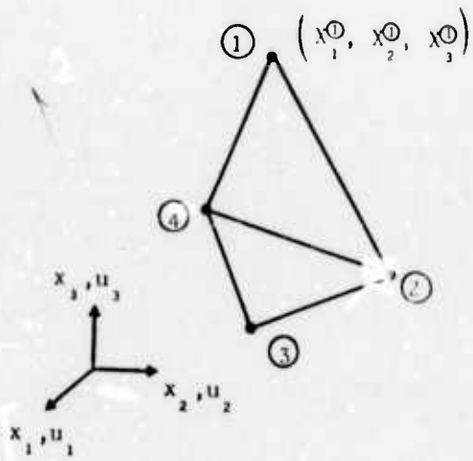
from which we obtain Eq. (2.10) in Section 2.4.

A considerable reduction in the computing effort can be achieved by diagonalizing the mass matrix [M], i.e., by lumping the element masses at the node points. One way of diagonalizing [M] is to set each diagonal term equal to the sum terms in the corresponding row. Many studies have been conducted to examine the effects that lumping the masses at the nodes has on the resulting calculations. The general concensus is that little if any advantage is gained by carrying the nondiagonal mass matrix, and considerable computational advantage is gained by diagonalizing [M].

A.3 ISOPARAMETRIC ELEMENTS

Numerical refinement and computational efficiency is of the utmost importance in treating stress waves in three-dimensional solids. For this reason, highly sophisticated three-dimensional elements are being considered. Figure A.1 illustrates the basic tetrahedral element that is presently coded into STATIC, one of S^3's static finite element analyzers. The element is of comparable accuracy to a low order differencing scheme, with properties similar to the basic triangular element used for two-dimensional analysis. The tetrahedral shape is ideal for simulating irregular geometries as has been demonstrated by Frazier (1969).

Both two- and three-dimensional isoparametric elements are presently being coded into STATIC. The remaining portion of this section briefly describes how these higher order elements are being developed in three-dimensional space.



Inner-element Displacements:

$$u_i(\underset{\sim}{x}) = C_i^{(\beta)} + C_i^{(1)} x_1 + C_i^{(2)} x_2 + C_i^{(3)} x_3$$

$$= \langle \phi(\underset{\sim}{x}) \rangle \{U_i\}$$

Fig. A.1--Basic tetrahedron element that permits linear displacements within the region of the element and gives continuous displacements across the element boundaries.

As indicated by Figs. A.2 and A.3, a skewed hexahedron representing a typical finite element is mapped into a two-unit cube. This mapping, which is not necessarily conformal, is defined by the expression

$$x_i = \langle \phi^e(\underset{\sim}{\eta}) \rangle \{X_i^e\} \tag{A.21}$$

in which $\{X_i^e\}$ is a column listing of the coordinates of the node points associated with element e expressed in the global Cartesian system x_i, and $\langle \phi^e(\underset{\sim}{\eta}) \rangle$ is a row listing of spatial interpolating functions expressed in the local coordinate system η_i. Specifically, the spatial interpolation functions for the linear isoparametric element of Fig. A.2 are given by

$$\phi_1^e(\underset{\sim}{\eta}) = (1+\eta_1)\ (1+\eta_2)\ (1+\eta_3)/8$$

$$\phi_2^e(\underset{\sim}{\eta}) = (1-\eta_1)\ (1+\eta_2)\ (1+\eta_3)/8$$

$$\phi_3^e(\underset{\sim}{\eta}) = (1+\eta_1)\ (1-\eta_2)\ (1+\eta_3)/8$$

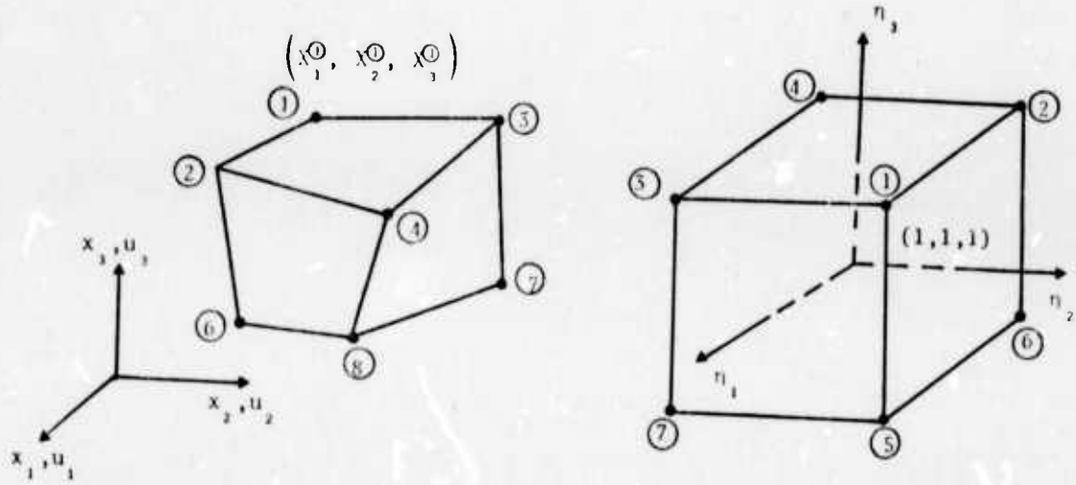
with similar expressions for $\phi_4^e(\underset{\sim}{\eta}) \ldots \phi_8^e(\underset{\sim}{\eta})$, (A.22)

and the spatial interpolation functions for the quadratic isoparametric element of Fig. A.3 are given by

$$\phi_1^e(\underset{\sim}{\eta}) = (1+\eta_1)\ (1+\eta_2)\ (1+\eta_3)\ (\eta_1+\eta_2+\eta_3-2)/8$$

$$\phi_2^e(\underset{\sim}{\eta}) = (1-\eta_1)\ (1+\eta_2)\ (1+\eta_3)\ (-\eta_1+\eta_2+\eta_3-2)/8$$

with similar expressions for $\phi_3^e(\underset{\sim}{\eta}) \ldots \phi_8^e(\underset{\sim}{\eta})$



(a) Global coordinates of the assembled solid.

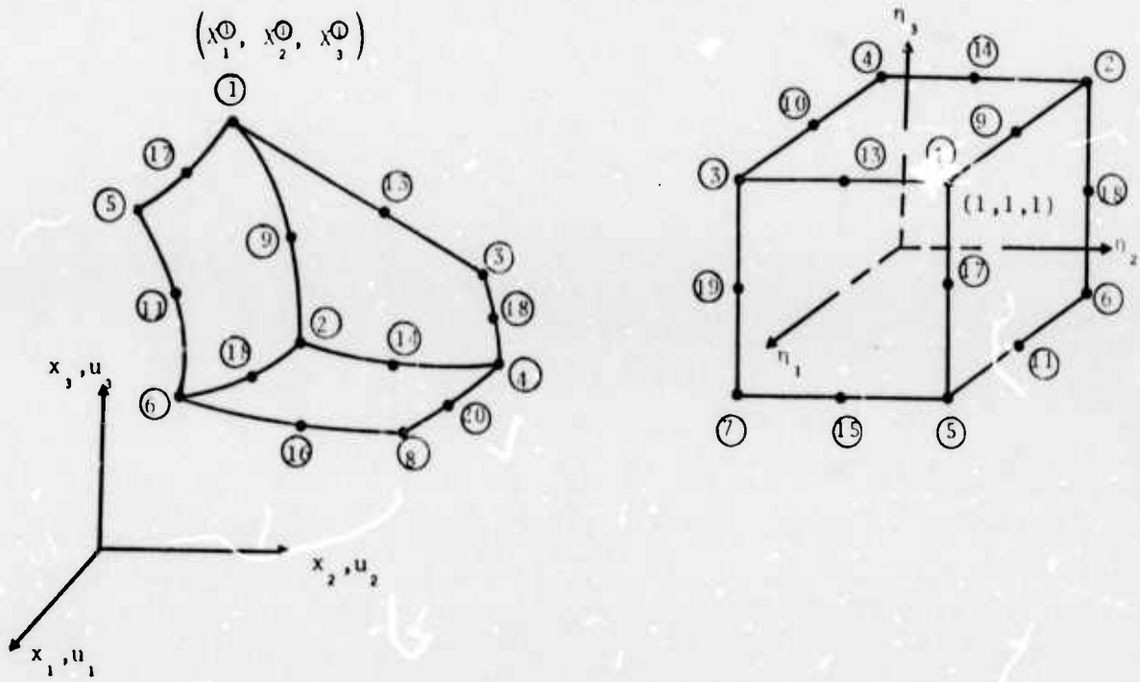
(b) Local natural coordinates of the element.

Inner-element Displacements (linear):

$$u_i(\underset{\sim}{\eta}) = \sum_{\alpha=0}^{1} \sum_{\beta=0}^{1} \sum_{\gamma=0}^{1} c_i^{(\alpha,\beta,\gamma)} \eta_1^{\alpha} \eta_2^{\beta} \eta_3^{\gamma}$$

$$= \langle \phi(\underset{\sim}{\eta}) \rangle \{ U_i \}$$

Fig. A.2--Basic isoparametric hexahedron element that permits linear displacements within the region of the element and gives continuous displacements across element boundaries.



(a) Global coordinates of the assembled solid.

(b) Local natural coordinates of the element.

Inner-element Displacements (quadratic):

$$u_i(\underset{\sim}{\eta}) = \sum_{\alpha=0}^{2} \sum_{\beta=0}^{2} \sum_{\gamma=0}^{2} C_i^{(\alpha,\beta,\gamma)} \eta_1^{\alpha} \eta_2^{\beta} \eta_3^{\gamma}$$

$$= \langle \phi(\underset{\sim}{\eta}) \rangle \{U_i\}$$

Fig. A.3--Curved isoparametric element that permits quadratic displacements within the region of the element and gives continuous displacements across element boundaries.

$$\phi_9^e(\underset{\sim}{\eta}) = (1-\eta_1)^2(1+\eta_2)\ (1+\eta_3)/4$$

$$\phi_{10}^e(\underset{\sim}{\eta}) = (1-\eta_1)^2(1-\eta_2)\ (1+\eta_3)/4$$

with similar expressions for $\phi_{11}^e(\underset{\sim}{\eta}) \ldots \phi_{20}^e(\underset{\sim}{\eta})$ (A.23)

Thus Eq. (A.21) yields, point by point, the global coordinate value x_i corresponding to each local coordinate value η_j. When η_j is equal to an element node point $(\pm 1, \pm 1, \pm 1)$, x_i will take on the value of the corresponding node point in the global system. The same requirement is placed on the displacement field: When the displacement field is evaluated at an element node point, the corresponding nodal displacement is obtained. Consequently, the same spatial interpolation functions are used for the displacement field.

$$u_i(\underset{\sim}{\eta}) = \left\langle \phi^e(\underset{\sim}{\eta}) \right\rangle \left\{ U_i^e \right\} \tag{A.24}$$

Spatial derivatives with respect to the global Cartesian coordinates are expressed in terms local coordinate derivatives using the chain rule for differentiation,

$$\left\langle \frac{\partial \phi^e}{\partial x_k}(\underset{\sim}{\eta}) \right\rangle = \frac{\partial \eta_m}{\partial x_k} \left\langle \frac{\partial \phi^e}{\partial \eta_m}(\underset{\sim}{\eta}) \right\rangle . \tag{A.25}$$

We note that $\partial \eta_m / \partial x_k$ is not known explicitly, but its inverse, $\partial x_k / \partial \eta_m$, is computed directly from Eq. (A.21),

$$\frac{\partial x_k}{\partial \eta_m} = \left\langle \frac{\partial \phi^e}{\partial \eta_m}(\underset{\sim}{\eta}) \right\rangle \left\{ X_k^e \right\} \tag{A.26}$$

The volume integration that is involved in defining element properties, Eqs. (A.17), (A.18), and (A.19), is carried out in the local coordinate system over the two-unit cube using

the property

$$dx_1\ dx_2\ dx_3 = J(\underset{\sim}{\eta})d\eta_1\ d\eta_2\ d\eta_3 \tag{A.27}$$

where the transformation Jacobian J is simply the determinate of $\partial x_k/\partial \eta_m$ of Eq. (A.26), i.e.,

$$J(\underset{\sim}{\eta}) = \mathrm{Det}\left|\left\langle \frac{\partial \phi^e}{\partial \eta_j}(\underset{\sim}{\eta})\right\rangle \{X_i^e\}\right| . \tag{A.28}$$

Using the appropriate spatial interpolation functions given by Eqs. (A.22) or (A.23), the element mass matrix, as defined in Eq. (A.17), is given by

$$[M^e] = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} \left\langle \phi^e(\underset{\sim}{\eta})\right\rangle^T \rho \left\langle \phi^e(\underset{\sim}{\eta})\right\rangle J\ d\eta_1 d\eta_2 d\eta_3 \tag{A.29}$$

and the element stiffness matrix, as defined by Eq. (A.18), is given by

$$[K_{ij}^e] = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} \left(\frac{\partial x_k}{\partial \eta_m}\right)^{-1} \left\langle \frac{\partial \phi^e}{\partial \eta_m}(\underset{\sim}{\eta})\right\rangle^T E_{ikjl} \left\langle \frac{\partial \phi^e}{\partial \eta_n}(\underset{\sim}{\eta})\right\rangle$$

$$\left(\frac{\partial x_\ell}{\partial \eta_n}\right)^{-1}$$

$$J\ d\eta_1\ d\eta_2\ d\eta_3 \tag{A.30}$$

The actual integration is carried out numerically using the Gaussian quadrature integration scheme.

APPENDIX B

ARTIFICIAL DAMPING

B.1 CAUSES OF NUMERICAL OSCILLATIONS

In the absence of numerical damping, computer calculations for the propagation of discontinuous wave forms contain oscillations that die out away from the propagating discontinuity, as illustrated in Figs. 3.6 and 3.7. These nuisance oscillations are the result of two phenomena. One is not related to the propagating nature of the wave but is simply due to the truncation error of approximating a discontinuous function by piecewise polynomials of low order. The result is quite analogous to the Gibb's phenomena of Fourier Analysis – oscillations in the vicinity of discontinuities. Just as with the oscillations from a truncated Fourier series, a nonoscillating signal can be obtained by suppressing the participation of the higher wave numbers (short wave length) in a transitional manner so that the highest wave number is suppressed the most.

The other source of the numerical oscillations results from numerical dispersion – the higher wave numbers tend to propagate at a velocity slightly different from the low wave numbers, which propagate at the velocity of the medium. When the consistent mass matrix of the FE method is employed, the higher wave numbers propagate at a higher velocity than that of the medium. This fact is the direct result of the eigenvalues (natural frequencies) of the discrete system being bounded from below according to the Rayleigh quotient, Washizu (1968). The result of the high wave numbers propatating at a slightly higher velocity is seen in work by Goudreau (1970).

The dispersion from the lumped-mass approximation of FE, which is equivalent to the FD method, is not entirely

one-sided. The eigenvalues obtained by using the lumped mass approximation can lie both above and below the corresponding eigenvalues for the bounded continuum. The preponderance of dispersion, however, is of the type where the higher wave numbers propagate at a slightly lower velocity than the velocity for the medium.

The elimination of the dispersion-induced oscillations is similar to that for the nondynamic oscillations – suppress the higher wave numbers in a transitional manner. The suppression of the higher modes needs to be more pronounced as time increases in the calculations, because numerical dispersion causes the higher modes to become out of phase with the wave front by an ever-increasing amount. Therefore, a steep wave front will necessarily deteriorate somewhat at late times. In Fig. 3.8, we observe that a velocity step (particle velocity) spreads over about 10 zones after propagating through 160 zones.

It is appropriate to note that earth materials display damping properties. Low amplitude seismic waves are attenuated in a frequency dependent manner – high frequency waves (short wave lengths) are attenuated more rapidly than low frequency waves (long wave lengths). Also, high amplitude waves passing through a small block of material trace out a hysteresis loop when stress is plotted against strain. The area of the loop gives the energy density that is lost in the nonlinear dynamic process. The type of artificial damping that is developed below to remove numerical oscillations also results in a hysteresis loop, which is an ellipse. The numerical viscous damping parameter can be adjusted to cause the area of the elliptical hysteresis loop to be equal to the area that results from the nonlinear earth material so that the energy loss of the linear system becomes equal to the energy loss from the nonlinear material. Care must be exercised in this procedure to assure the two systems are

made equivalent at the predominant frequency of the propagating waves, because the energy that is dissipated per cycle of ground motion may have a different frequency dependence for the two systems.

B.2 MATHEMATICAL ANALYSIS OF ARTIFICIAL DAMPING

In this development we will treat three types of artificial damping; each has a different frequency dependence. The damping matrix of Section 2.4 is set to

$$[\underline{\underline{C}}] = \alpha[\underline{\underline{M}}] + 2\zeta[\underline{\underline{KM}}]^{\frac{1}{2}} + \beta[\underline{\underline{K}}] \tag{B.1}$$

where $[\underline{\underline{M}}] = \delta_{ij}[M]$. The matrix square root $[\underline{\underline{KM}}]^{\frac{1}{2}}$ is defined in terms of the eigenvectors (mode shapes) $\{\underline{\Phi}\}_m$ and eigenvalues (natural frequencies) ω_m, $m = 1,2, \ldots 3N$ for the discrete system

$$[\underline{\underline{KM}}]^{\frac{1}{2}} = [M] \sum_{m=1}^{3N} \omega_m M_m^{-1} \{\underline{\Phi}\}_m \{\underline{\Phi}\}_m^T [M] \tag{B.2}$$

where the 3N eigenvalues and corresponding eigenvectors are computed from the equation

$$[\underline{\underline{K}}]\{\underline{\Phi}\}_m = \omega_m^2 [M]\{\underline{\Phi}\}_m , \quad m = 1,2, \ldots 3N \tag{B.3}$$

and M_m is the normalization constant for scaling the $m^{\underline{th}}$ eigenvector. Orthogonality between the various eigenvectors is expressed

$$\{\underline{\Phi}\}_n^T [M] \{\underline{\Phi}\}_m = M_m \delta_{nm}$$

The eigenvalues are conveniently arranged in ascending order, i.e., $\omega_m \leq \omega_{m+1}$.

The expression for the artificial damping matrix above, Eq. (B.1), is substituted into the linearized discrete equation of motion, Eq. (2.10), to obtain

$$[M]\{\ddot{\underline{U}}(t)\} + [\alpha\underline{\underline{M}} + 2\zeta\sqrt{\underline{\underline{K}}\underline{\underline{M}}} + \beta\underline{\underline{K}}]\{\dot{\underline{U}}(t)\} + [\underline{\underline{K}}]\{\underline{U}(t)\} = \{\underline{F}(t)\}. \tag{B.4}$$

For the purpose of conciseness, we will not develop modal decomposition for systems in which nodal displacements are specified;* we will restrict our development to systems in which surface tractions are specified on the boundaries. For the case involving no specified nodal displacements, the eigenvectors form a complete set of basic functions for expressing the nodal displacements at an instant in time. That is, the nodal displacements can be expressed as a time varying linear combination of the eigenvectors for the discrete system

$$\{\underline{U}(t)\} = \sum_{k=1}^{3N} T_k(t)\ \{\underline{\Phi}\}_k \tag{B.5}$$

where $T_k(t)$ is the time varying participation coefficient for the $k^{\underline{th}}$ eigenvector.

The above eigenvector expansion for the nodal displacements is substituted into Eq. (B.4). The resulting simultaneous equations, expressed in matrix form, are then decoupled by premultiplying through by $M_n^{-1}\{\underline{\Phi}\}_n^T$ to obtain

*The more general case involving specified nodal displacement time histories has been developed elsewhere by Frazier (1969), also by Przemieniecki (1968). The more general development requires additional notation which does not add to our understanding the influence of the various types of damping.

$$\ddot{T}_n(t) + 2c_n\dot{T}_n(t) + \omega_n^2 T_n(t) = Q_n(t) \tag{B.6}$$

where

$$c_n = \alpha/2 + \zeta\omega_n + \beta/2\omega_n^2$$

and

$$Q_n(t) = M_n^{-1}\{\underline{\Phi}\}_n^T \{\underline{F}(t)\} .$$

In obtaining the decoupled equation above, we have used the substitution

$$M_n^{-1}\{\underline{\Phi}\}_n^T[\underline{\underline{K}}M]^{\frac{1}{2}} \sum_{\beta=1}^{3N} \dot{T}_k(t)\{\underline{\Phi}\}_k = \omega_n\dot{T}_n(t)$$

from Eq. (B.2). Also, we have substituted

$$M_n^{-1}\{\underline{\Phi}\}_n^T[\underline{\underline{K}}] \sum_{k=1}^{3N} \dot{T}_k(t)\{\underline{\Phi}\}_k = \omega_n^2\dot{T}_n(t)$$

using Eq. (B.3).

The general solution for the $n^{\underline{th}}$ participation coefficient is now expressed using a convolution integral in time

$$\begin{aligned} T_n(t) = {} & e^{-c_n t} \cos\omega_{dn} t \; T_n(0) \\ & + \frac{1}{\omega_{dn}} e^{-c_n t} \left(\dot{T}_n(0) + c_n T_n(0)\right) \sin\omega_{dn} t \\ & + \frac{1}{\omega_{dn}} \int_0^t e^{-c_n(t-\tau)} \sin\omega_{dn}(t-\tau) Q_n(\tau) d\tau \end{aligned} \tag{B.7}$$

where

$$\omega_{dn} = \sqrt{\omega_n^2 - c_n^2} \ ,$$

$$T_n(0) = M_n^{-1}\{\underline{\Phi}\}_n^T[M]\{\underline{U}(0)\}$$

and

$$\dot{T}_n(0) = M_n^{-1}\{\underline{\Phi}\}_n^T[M]\{\dot{\underline{U}}(0)\}$$

$\{\underline{U}(0)\}$ and $\{\dot{\underline{U}}(0)\}$ being the initial conditions for the nodal displacements and velocities, respectively.

From Eq. (B.7), we can see how the various types of damping influence the dynamic response to a linear system. First, we see that the natural frequencies are reduced by the factor $\sqrt{1 - c_n^2/\omega_n^2} \leq 1$ which may result in additional dispersion, particularly in the higher nodes where c_n is the greatest. More importantly, however, we find that the free vibrations that result from excitation at $t = 0$ are damped by the factor

$$e^{-c_n t} = e^{-\frac{\alpha}{2}t} \, e^{-\zeta\omega_n t} \, e^{-\frac{\beta}{2}\omega_n^2 t} \tag{B.8}$$

Thus, we have shown that the term α damps all eigenvectors equally, ζ damps eigenvectors an amount proportional to their respective natural frequencies, and β damps as the square of the natural frequencies. Incidentally, the β damping, which has been incorporated into the ILLIAC time stepping algorithm, results in damping only in elements undergoing a strain rate; those elements experiencing rigid body deformations experience no damping.

B.3 ESTIMATING AN OPTIMUM DAMPING COEFFICIENT

The β-type damping is most appropriate for the numerical calculations and is used in the time stepping algorithm of Section 2.4. We will now develop a technique for predicting values of β for use in numerical calculations.

Equation (B.8) permits us to set β to damp a specified eigenvector any desired amount. First, let us relate eigenvectors, eigenvalues, and wave lengths. For 1-D geometry (or plane waves), an eigenvector, characterized by a wave length λ_n, has an associated eigenfrequency given by

$$\omega_n = 2\pi \frac{c}{\lambda_n} \tag{B.9}$$

where $c = V_p$ for propagating P waves and $c = V_s$ for propagating S waves. We note that this expression is used only to estimate the natural frequencies of the discrete system. Our estimate for ω_n is substituted into (B.8) with $\alpha = \zeta = 0$ to obtain a damping factor

$$e^{-c_n t} = e^{-2\pi^2 c^2 \beta\, t/\lambda_n^2} \quad . \tag{B.10}$$

Thus, to select an optimum β for a particular calculation we must establish three quantities:

1. The wave length of the spurious oscillations that are to be damped. From Fig. 3.6, we find spurious oscillations with wave lengths up to about six grid dimensions, i.e., $\lambda_n = 6\Delta X$.
2. The time at which the oscillations should disappear. This presupposes that we are willing to tolerate some oscillations in the very early stages of the calculations. Let us consider the case where $\Delta t = \frac{\Delta X}{2V_p}$ and the oscillations

are to be removed by the time a P wave crosses 50 grids, i.e.,

$$t = 100\ \Delta t = 50\ \frac{\Delta X}{V_p}$$

3. A damping factor

$$e^{-c_n t} = e^{-\frac{\beta}{2}\omega_n^2 t} = e^{-2} = 0.135$$

is sufficient to remove the oscillations.

The optimum value of β for the job is then calculated from Eq. (B.10)

$$e^{-2\pi c^2 \beta\ t/\lambda_n^2} = e^{-2} \tag{B.11}$$

so that

$$\beta_p/\Delta t \cong 0.15 \tag{B.12}$$

for P waves. Similarly, we compute an optimum β for eliminating spurious oscillations after S waves have traversed 50 grid dimensions to obtain $\beta_s = 3\beta_p$ for $V_p/V_s = \sqrt{3}$. It is interesting to note that the test calculations presented in Section 3.3, Fig. 3.7, indicate $0.1 < \beta/\Delta t < 0.2$ for the P-wave calculations. For propagating S waves we got $\beta/\Delta t \cong 0.5$.