To the Graduate Council:

I am submitting herewith a thesis written by William W. Murphy entitled "Efficient Integration of an Isoparametric Finite Element Matrix." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Steven M. Serbin

We have read this thesis
and recommend its acceptance:

John E Akin

Julius Smith

Accepted for the Council:

Hilton A. Smith
Vice Chancellor for
Graduate Studies and Research

EFFICIENT INTEGRATION OF AN ISOPARAMETRIC FINITE ELEMENT MATRIX

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee

William W. Murphy

June 1974

ACKNOWLEDGMENT

The author is grateful for the assistance of Dr. Steven M. Serbin, who directed the preparation of this thesis.

The author also wishés to express his appreciation to Dr. John E. Akin, Associate Professor of Engineering Science and Mechanics at The University of Tennessee, who furnished the computer program, related material and guidance for this thesis.

A special thanks goes to the Computing Services staff of the Tennessee Valley Authority, Knoxville, Tennessee, for providing the computer time, printing services, and gracious assistance.

1147141

ABSTRACT

The purpose of this investigation is to explore programming procedures for improving the numerical integration techniques presently being used to evaluate the element matrices for isoparametric finite element analysis.

A general two-dimensional isoparametric finite element computer program developed by Akin is used as a starting basis for investigation. Certain proposed modifications are applied to this program. In addition, several FORTRAN optimization techniques are used to obtain the final version. Additional programs are developed to further test the resulting integration efficiency.

For analysis a steady state heat conduction problem with specified temperature boundary conditions is chosen.

An "efficiency factor" is introduced to measure the efficiency of each modification and corresponding run times, storage, and efficiencies for two, three, and four point quadrature are summarized.

Results demonstrate that substantial improvement in computational time and efficiency is achieved.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
II.	BRIEF REVIEW OF FINITE ELEMENT THEORY	3
	General Description of the Method	3
	The Extremum Problem	6
	Variational Principle	7
III.	THE ISOPARAMETRIC FINITE ELEMENT CONCEPT	15
	Introduction	15
	Formulation of Isoparametric Element Matrix	17
	Numerical Integration	27
IV.	PROGRAMMING CONSIDERATIONS	32
	Introduction	32
	Common Programming Procedures	32
	Proposed Programming Procedures	35
	Fortran Optimization	38
V.	MODIFICATIONS AND NUMERICAL RESULTS	42
	Introduction	42
	Program Efficiency	47
	Integration Efficiency	58
	Concluding Remarks	58
VI.	CONCLUSIONS	63
BIBLIOGRAPHY		66

CHAPTER PAGE

APPENDICES. 68

APPENDIX A. 69

APPENDIX B. 75

VITA. 116

LIST OF TABLES

TABLE		PAGE
I.	Estimated Effects of First Modification	37
II.	Estimated Effects of Second Modification.	39
III.	Relationship Between Symbols and FORTRAN Names.	46
IV.	Finite Element Program (200 Elements)--Run Times, Original Version.	48
V.	Finite Element Program (200 Elements)--Run Times, First Modification.	49
VI.	Finite Element Program (200 Elements)--Run Times, Second Modification	52
VII.	Finite Element Program (200 Elements)--Run Times, Final (Optimized) Modification.	56
VIII.	Finite Element Program (200 Elements)--Core Storage, Average Run Times and Efficiency Factors (E).	57
IX.	Element Matrix Integration (1001 Elements)--Run Times, Original Version	59
X.	Element Matrix Integration (1001 Elements)--Run Times, Final (Optimized) Version.	60
XI.	Element Matrix Integration (1001 Elements)--Core Storage, Average Run Times and Efficiency Factors (E)	61
A-1.	Abscissae and Weight Coefficients of the Gaussian Quadrature Formula.	74

LIST OF FIGURES

FIGURE		PAGE
1.	Two-Dimensional Region Represented as an Assemblage of Triangular Elements.	4
2.	Isometric View of a Triangular Element Showing a Typical "Linear" Interpolation Function $N_i(x,y)$.	11
3.	Region Having Same Number of Degrees of Freedom with "Parabolic" and "Linear" Elements.	16
4.	The Curvilinear Mapping of Some Elements	19
5.	Two-Dimensional Mapping of an Isoparametric Quadrilateral with Four Nodes.	23
6.	Steady State Heat Conduction in a Unit Square Plate. . . .	43
7.	One-Dimensional Gaussian Integration	70
8.	Integration Points for n = 3 in a Square Region (Exact for Polynomial of Fifth Order in Each Direction)	73

CHAPTER I

INTRODUCTION

In finite element analysis, the evaluation of element matrices by numerical integration can represent a significant proportion of the total computational time. Thus efficient programming techniques are of great practical importance.

Gaussian quadrature, which requires the least number of evaluations for a certain required accuracy is used extensively. However, some economic limit on this accuracy must obviously be imposed. The determination of an adequate minimum number of Gauss points is of some importance.

To proceed in the evaluation of the element properties it is necessary first to specify certain "interpolation" functions and their derivatives and second, to specify the order of integration. For a given class of problems with the same form of matrices, it is only necessary to change the prescription of the interpolation functions to achieve a variety of possible elements.

Present computational procedures require that the evaluation of the Gaussian integration data and the prescription of the interpolation functions (and derivatives) be done within the element routine, and thus be determined for <u>each</u> element. Akin proposed two modification procedures to improve the computational efficiency. The first suggests that the Gaussian data be determined initially,

stored, and passed to the element routine for use in numeral integration. A second, similar procedure was proposed for the evaluation of the "local" derivatives of the element interpolation functions.

It is the purpose of this paper to examine these two procedures and their effect on computation time, storage requirements, and accuracy. Two, three, and four Gauss points in each dimension will be used. Additional improvements in computational time will be attempted by FORTRAN optimizing techniques.

In Chapter II we review the governing equations, the associated functional and the minimization of the functional for the finite element model. Interpolation functions are introduced which must satisfy certain specified convergence criteria. The general three dimensional case with a scalar unknown is examined.

The isoparametric finite element concept is introduced in Chapter III. Certain features of an isoparametric quadrilateral element of the type used in this investigation are outlined. Gaussian quadrature techniques are also discussed, with an example of the two dimensional steady state heat conduction problem used for analysis in this study.

Chapter IV outlines the present and proposed programming procedures, as well as various FORTRAN optimizing techniques. In Chapter V the actual programming changes are discussed and there is a summary of numerical results. Chapter VI contains the conclusions; the Appendices include Gaussian formulas, tabulated Gaussian data, and program listings.

CHAPTER II

BRIEF REVIEW OF FINITE ELEMENT THEORY

I. GENERAL DESCRIPTION OF THE METHOD

In brief, the basis of the finite element method is the representation of a body, structure, specific quantity of solid material, fluid, or other media of interest by an assemblage of subdivisions called finite elements, Figure 1. For some problems the relevant continuum may consist simply of a region of space in which a phenomenon occurs. In cases such as fluid mechanics the assemblage represents a spatial subdivision rather than a material subdivision.

These elements are considered interconnected at joints which are called nodes or nodal points. The number of nodal parameters necessary to specify completely the unknown variables of the finite element are the degrees of freedom of the element. Each is specifically identified with a single nodal point and usually represents an unknown having a clear physical meaning. A combination of certain simple functions is chosen to approximate the unknown function over each element. Such assumed functions are called interpolation or shape functions. A unique interpolation function must be chosen for each nodal degree of freedom. The unknown magnitudes or amplitudes of the interpolation functions determine the value of the unknown variable at the nodal points. Hence, the final solution will yield the approximate values of the unknown variable at discrete locations in the

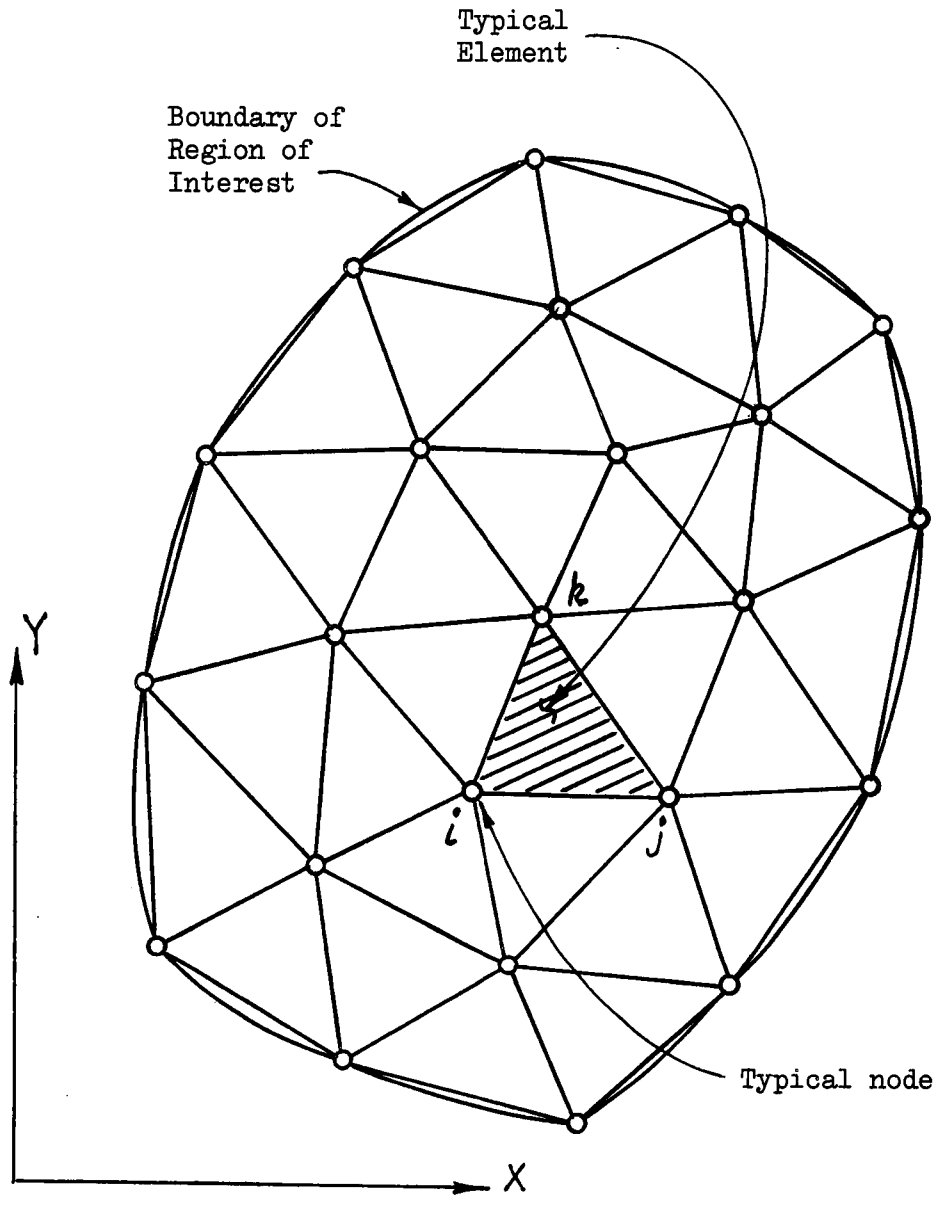


Figure 1. Two-dimensional region represented as an assemblage of triangular elements.

body, the nodal points. An interpolation function can be expressed in various simple forms, such as polynomial and trigonometric functions. Since polynomials offer ease in mathematical manipulations, they have been employed commonly in finite element applications.

A variational principle is usually employed to obtain the set of governing equations for each element. These equations relate the nodal unknowns to the nodal applied parameters through an element characteristic (or square) matrix which generally characterizes the properties of the matter or material considered in the problem. Examples are conductivity, stiffness, flexibility, and permeability matrices. These are also generally referred to simply as element matrices.

When all element matrices have been determined, the assembly of the algebraic equations for the overall discretized continuum must be determined. In general, the basis for an assembly method is that the nodal interconnections require the values at a node to be the same for all elements adjacent to that node. The overall equilibrium relations will again be expressed as a set of simultaneous equations.

These equations cannot be solved until the geometric boundary conditions are taken into account by appropriate modification of the equations. A geometric boundary condition arises from the fact that nodal values may be prescribed at the boundaries or edges of the body or structure.

The algebraic equations thus assembled are solved for the unknown values.

II. THE EXTREMUM PROBLEM

Governing Equations

The general Poisson "quasi-harmonic" equation governing the behavior of some unknown physical quantity, ϕ, can be written as

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial \phi}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial \phi}{\partial z}\right) + Q = 0 \tag{1}$$

in which ϕ is the unknown function, assumed to be single valued within the region or domain and k_x, k_y, k_z and Q are known, specified, functions of x, y, and z. If Q = 0, (1) is known as the Laplace equation.

For heat conduction (steady state) the functions k_x, k_y, and k_z are the anisotropic conductivity coefficients, the function Q is the rate of heat generation, and the unknown function ϕ is the temperature (provided the coordinates coincide with the principle axes of the material). But no matter what the physical interpretation, the mathematical problem remains the same.

Boundary Conditions

The physical conditions of the particular problem will impose certain boundary conditions. The two most commonly encountered cases are those in which

1. The value of ϕ is specified on the boundary, B

$$\phi = \phi_B \tag{2}$$

2. $$k_x \frac{\partial \phi}{\partial x} l_x + k_y \frac{\partial \phi}{\partial y} l_y + k_z \frac{\partial \phi}{\partial z} l_z + q + \alpha\phi = 0 \tag{3}$$

on the boundary in which l_x, l_y, and l_z are the direction cosines of the outward normal to the boundary surface.

In a conduction problem q represents the (heat) flux per unit surface and $\alpha\phi$ the convection loss.

III. VARIATIONAL PRINCIPLE

Associated Functional

Equation (1) together with the boundary conditions specifies the problem in a unique manner. However, an alternate formulation is possible with the aid of the calculus of variations. The well-known Euler theorem states that the problem defined above is identical to that of finding a function ϕ which minimizes the functional χ

$$\chi = \int_V \left[1/2 \left\{ k_x \left(\frac{\partial\phi}{\partial x}\right)^2 + k_y \left(\frac{\partial\phi}{\partial y}\right)^2 + k_z \left(\frac{\partial\phi}{\partial z}\right)^2 \right\} - Q\phi \right] dxdydz \tag{4}$$

subject to ϕ obeying the same boundary conditions, and integration being carried out over the whole region [9].

While condition (1) of prescribed boundary conditions is easily implemented, condition (2) presents some difficulty. This is overcome by adding to the functional, the integral

$$\int_S (q\phi + 1/2\alpha\phi^2)\, dS \tag{5}$$

in which S is the surface where condition (2) applies, the minimization automatically yields the boundary condition [9].

The details of derivation of this fairly general form of Euler equation may be found in [1] and [9].

Minimization of the Functional-The Finite Element Model

When minimizing the functional defined by (4) it is necessary only to derive the necessary conditions for a typical element. The influence of other elements on the general minimum equations will follow an identical pattern, and to obtain the general equations it is simply necessary to add the contributions of all elements.

To minimize the functional χ with respect to the total number of parameters $\{\phi\}$ associated with the whole domain we must write a system of equations

$$\frac{\partial \chi}{\partial \{\Phi\}} = \left\{ \begin{array}{c} \dfrac{\partial \chi}{\partial \{\phi\}_1^e} \\ \dfrac{\partial \chi}{\partial \{\phi\}_2^e} \\ \vdots \end{array} \right\} = 0 \qquad (6)$$

where $\{\phi\}_\mu^e$ are the nodal values associated with the element "μ." For element μ we have

$$\frac{\partial \chi}{\partial \{\phi\}_\mu^e} = \left\{ \begin{array}{c} \dfrac{\partial \chi}{\partial \{\phi\}_{\mu,1}} \\ \dfrac{\partial \chi}{\partial \{\phi\}_{\mu,2}} \\ \vdots \\ \dfrac{\partial \chi}{\partial \{\phi\}_{\mu,n}} \end{array} \right\}^e \qquad (7)$$

hwere n are the number of nodes and $\{\phi\}_{\mu,j}$ the parameters associated with the nodal degrees of freedom at node j.

The value of the functional is equal to the sum of the contributions of each element, that is

$$\chi = \sum \chi^e \tag{8}$$

and a typical equation becomes

$$\frac{\partial \chi}{\partial \{\phi\}_\mu^e} = \sum \frac{\partial \chi^e}{\partial \{\phi\}_\mu^e}$$

with summation over all elements. The rule for the assembly of a general minimization set of equations is thus available.

In the special case where χ is a quadratic functional of $\{\phi\}$ and its derivatives we shall find that we can always write the element derivatives as

$$\frac{\partial \chi^e}{\partial \{\phi\}^e} = [s]^e \{\phi\}^e + \{t\}^e \tag{10}$$

in which $[s]^e$ and $\{t\}^e$ are matrices of constants. Now the minimizing set of equations (6) can be simply written as

$$\frac{\partial \chi}{\partial \{\Phi\}} = [S] \{\Phi\} + \{T\} = 0 \tag{11}$$

in which the submatrices are

$$[S_{ij}] = \sum [s_{ij}]^e \tag{12}$$

$$\{T_i\} = \sum \{t_i\}^e \tag{13}$$

with the summations over all elements [9].

The matrices $[s]^e$ and $[S]$ are commonly known as the element and system matrices respectively; and the matrices $\{t\}^e$ and $\{T\}$ as the element column and system column matrices.

Interpolation Functions

Consider the case of an unknown scalar function ϕ. At any point within a typical element, ϕ is prescribed in terms of the appropriate nodal values as follows.

Let

$$\phi(x,y,z) = [N(x,y,z)]\{\phi\}^e = [N_1, N_2, \ldots] \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \vdots \end{Bmatrix}^e \tag{14}$$

in which the components of $[N]$ are in general functions of position. The functions $[N]$ must be so chosen such that $\phi = \phi_i$ at node i. N_i must assume the value of unity at node i and zero at all other nodes, Figure 2.

The functions $[N]$ are called interpolation or shape functions and play a paramount role in finite element analysis.

Once the choice of appropriate interpolation functions has been made the determination of the element matrix and other relevant matrices follows well-known standard rules.

The selection of appropriate interpolation functions is, however, not arbitrary. To insure convergence to the correct results certain simple requirements have to be satisfied. Three conditions must be satisfied, [1] and [9].

1. The interpolation functions must be continuous within the element and must coincide with those of adjacent elements at the common boundaries.

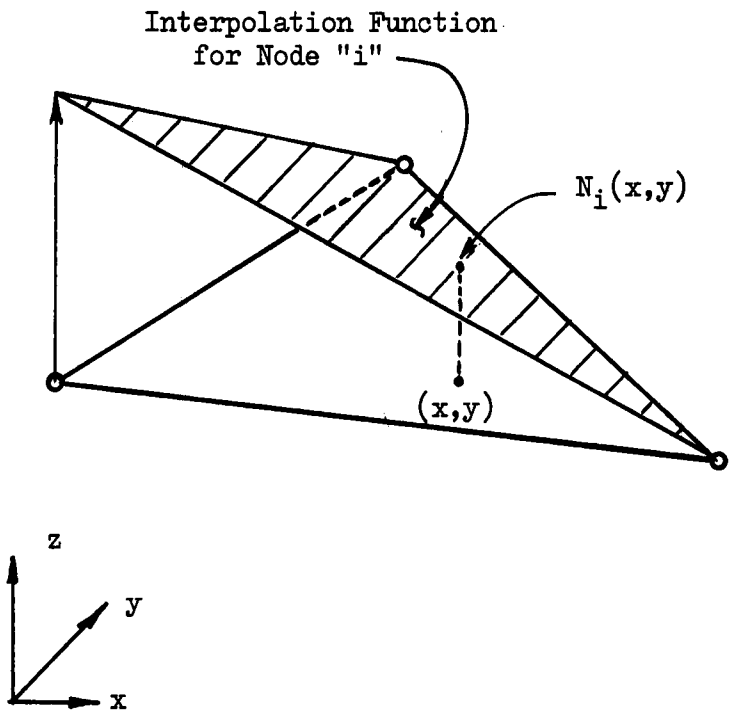


Figure 2. Isometric view of a triangular element showing a typical "linear" interpolation function $N_i(x,y)$.

2. The interpolation functions must allow all points on the element to experience the same variation of the unknown. This must be true for each of the possible nodal variations.

3. In the limit, the interpolation functions must include the possibility of satisfying any required state of constant first derivatives of the unknown parameter.

In finite element literature, formulations which satisfy the first condition above are called compatible or conforming. Elements which meet both the second and third requirements are known as complete, [1].

If polynomial expressions are used, it can be generally stated that the omission of higher order terms in favor of lower order terms is desirable, [9].

Functional Convergence Criteria

The function approximation given by (4) must obey certain completeness rules to enable convergence of results to be achieved as the subdivision into even smaller elements is attempted.

The element interpolation functions [N] must be such that, [9]

1. With a suitable choice of $\{\phi\}^e$ any constant values of ϕ or its derivatives present in the functional χ should be able to be represented in the limit as the element size decreases to zero.

2. At element interfaces ϕ and its derivatives of one order less than that occurring in the expressions which define the functional are continuous.

Finite Element Discretization--General Three Dimensional Case

If the unknown scalar function ϕ is defined, element by element, in the usual manner by (14), approximate minimization can be carried out.

Proceeding generally, we evaluate the element contribution on using the definitions of (4), (5), and (14).

For any node i, we can write by differentiating (4) and (5)

$$\frac{\partial \chi^e}{\partial \phi_i} = \int_{V^e} \left\{ k_x \frac{\partial \phi}{\partial x} \frac{\partial}{\partial \phi_i} \left(\frac{\partial \phi}{\partial x} \right) + k_y \frac{\partial \phi}{\partial y} \frac{\partial}{\partial \phi_i} \left(\frac{\partial \phi}{\partial y} \right) + k_z \frac{\partial \phi}{\partial z} \cdot \frac{\partial}{\partial \phi_i} \left(\frac{\partial \phi}{\partial z} \right) - Q \frac{\partial \phi}{\partial \phi_i} \right\} dxdydz \qquad (15)$$

$$+ \int_{S^e} \left(q \frac{\partial \phi}{\partial \phi_i} + \alpha \phi \frac{\partial \phi}{\partial \phi_i} \right) dS$$

where the second integral only applies if the element has an external boundary on which conditions of type (2) are specified.

Noting that

$$\frac{\partial \phi}{\partial x} = \left[\frac{\partial N_i}{\partial x}, \frac{\partial N_j}{\partial x}, \ldots \right] \left\{ \phi \right\}^e$$

and that

$$\frac{\partial}{\partial \phi_i} \left(\frac{\partial \phi}{\partial x} \right) = \frac{\partial N_i}{\partial x}$$

$$\frac{\partial \phi}{\partial \phi_i} = N_i$$

we have immediately for the element

$$\frac{\partial \chi^e}{\partial \{\phi\}^e} = [s]^e \left\{ \phi \right\}^e + \left\{ t \right\}^e \qquad (16)$$

in which the element matrix, $[s]^e$, is obtained from (14) and (15). A typical term of $[s]^e$ is

$$s_{ij}{}^e = \int_{V^e} \left\{ k_x \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + k_y \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + k_z \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} dxdydz \qquad (17)$$

and, with $dV = dxdydz$

$$t_i{}^e = - \int_{V^e} Q N_i \, dV + \int_{S^e} q N_i \, dS + \left(\int_{S^e} [N] \alpha N_i \, dS \right) \{\phi\}^e$$

Assembly of the whole set of minimization equations follows the usual rules.

CHAPTER III

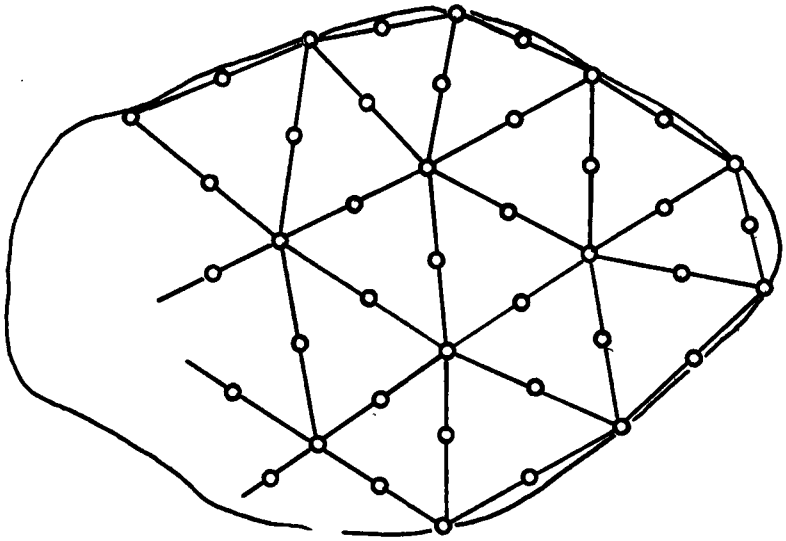
THE ISOPARAMETRIC FINITE ELEMENT CONCEPT

I. INTRODUCTION

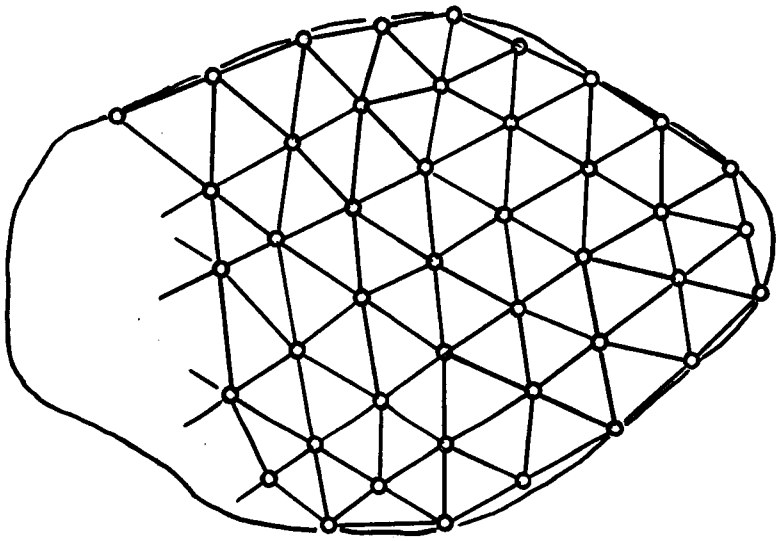
It has been frequently demonstrated in the past that for a given total number of degrees of freedom in a region, accuracy is increased by use of larger elements with a greater number of degrees of freedom [5]. Thus, for instance, consider the two subdivisions shown in Figure 3. The first, Figure 3 (a), is subdivided into so-called "parabolic" triangular elements and the second, Figure 3 (b) shows simpler "linear" triangles. If both regions have the same total degrees of freedom, in the majority of cases, the accuracy of the finite element method with "parabolic" elements will be superior.

This process of increasing the element size and degrees of freedom creates an obvious difficulty. The larger triangles can no longer follow the boundary to the same extent as the smaller ones. One advantage of the finite element formulation appears to have been forsaken.

To overcome the geometric difficulties presented by large elements, curved sides are essential. If complex elements with sides capable of taking on the boundary curvatures are possible then indeed progress can be made. Such a step forward has been achieved by the introduction of distorted, curvilinear elements [9]. However, the



a. "Parabolic" Triangular Elements



b. "Linear" Triangular Elements

Figure 3. Region having same number of degrees of freedom with "parabolic" and "linear" elements.

introduction of curvilinear elements presents some difficulties. First it is difficult to determine interpolation functions in cartesian coordinates that will satisfy the necessary convergence criteria and also be applicable to the elements of curvilinear shapes. Also, the development of the element equations usually requires the use of numerical integration; yet the use of cartesian coordinates is (usually) not well suited for numerical integration over complicated geometric shapes.

The formulation of interpolation functions and the calculation of element matrices have been both simplified and generalized by the introduction of isoparametric elements [9]. In this technique the same interpolation functions are used to define the element coordinates that are used to define the variation of the unknown.

II. FORMULATION OF ISOPARAMETRIC ELEMENT MATRIX

The Parent Element

Curvilinear elements utilize a local coordinate system that is usually not cartesian. These local coordinates (ξ, η, and ζ) are usually dimensionless and range from 0 to 1 or from -1 to 1. The latter range is usually preferred since it is directly compatible with the definition of absicssae used in standard numerical integration routines such as Gaussian quadrature that must be used to evaluate the element matrix.

Elements of the basic one, two, or three-dimensional types will be mapped into distorted forms in the global space as indicated in Figure 4. The basic element in undistorted, local coordinates is referred to as a "parent" element.

Interpolation functions in local coordinates. Consider a single scalar variable, ϕ, that is to be modeled in global space, (x, y, z) by a set of curvilinear elements with local coordinates (ξ, η, ζ). Recall that at a point within a typical element e one assumes ϕ is related to the nodal values of the element $\{\phi\}^e$ by

$$\phi(\xi, \eta, \zeta) \equiv [N(\xi, \eta, \zeta)]\{\phi\}^e \tag{18}$$

where [N] are the interpolation functions as previously defined (14).

Local derivatives of ϕ. The local derivatives of ϕ can be related to the nodal parameters by

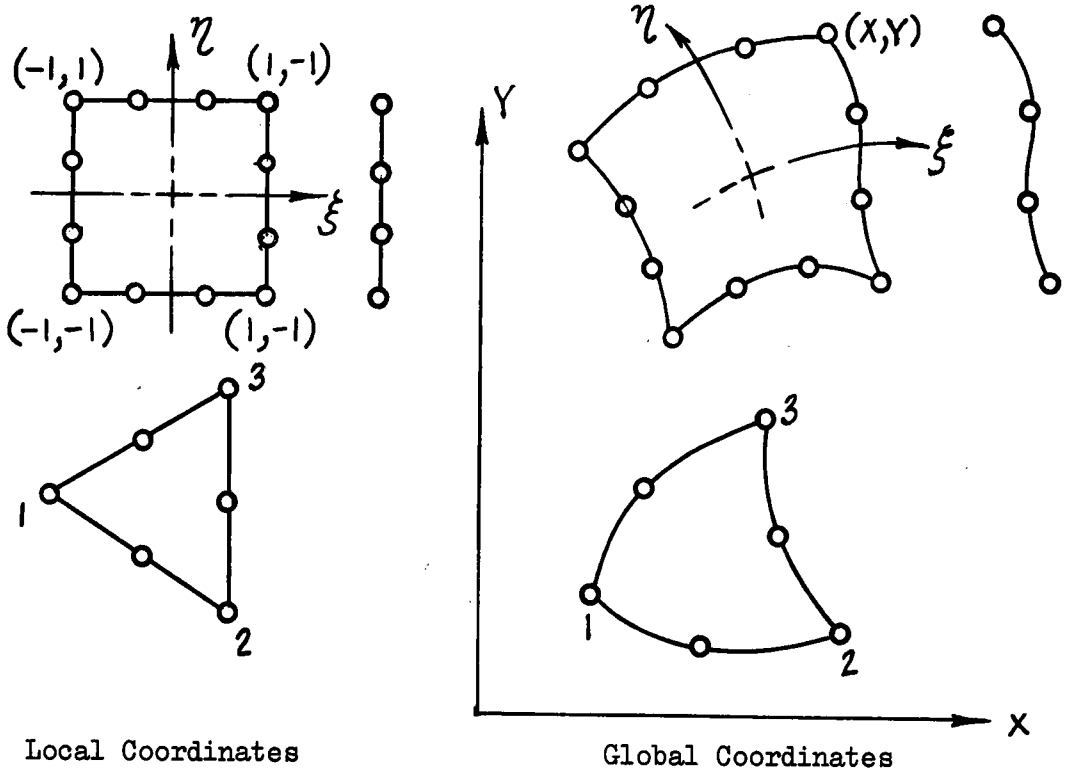
$$\frac{\partial \phi}{\partial \xi} = \left[\frac{\partial N}{\partial \xi}\right]\{\phi\}^e \tag{19}$$

Performing similar operations for the other dimensions, we obtain

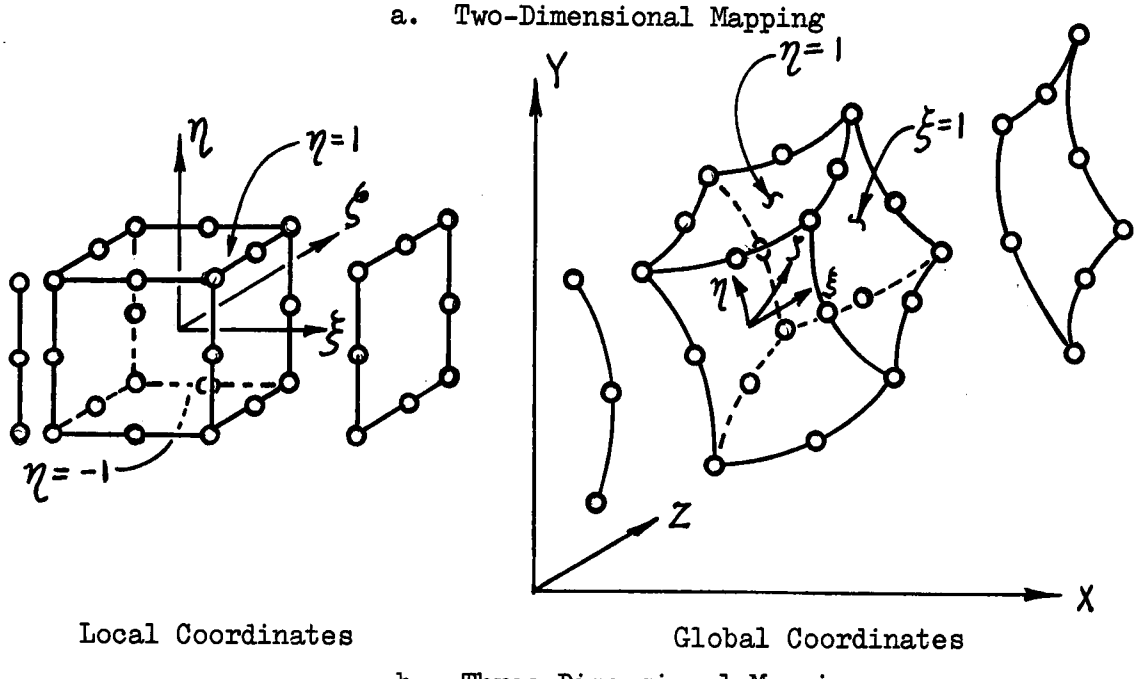
$$\{\partial \phi_1\} = [\Delta(\xi, \eta)]\{\phi\}^e \tag{20}$$

where

$$[\Delta] = \begin{bmatrix} \frac{\partial N}{\partial \xi} \\ \cdots \\ \frac{\partial N}{\partial \eta} \\ \cdots \\ \frac{\partial N}{\partial \zeta} \end{bmatrix} \tag{21}$$



a. Two-Dimensional Mapping



b. Three-Dimensional Mapping

Figure 4. The curvilinear mapping of some elements.

and the subscript 1 denotes local derivatives. Note that the matrices $[N]$ and $[\Delta]$ could be the same for all the elements in the system since they depend only on the local coordinates.

The Curvilinear, Isoparametric Element

With parent elements defined we shall now proceed to distort the shapes so that a closer representation of boundaries may be attempted.

The position within the new elements is determined by new curvilinear coordinates x, y, z, taking place of ξ, η, ζ. Just as the latter coordinates took constant values on element faces in the original elements, so will the new variables in the distorted elements. This by itself does not define the mapping relationship needed. This will be provided by the basic isoparametric definition.

Coordinate transformations. A most convenient method of establishing the coordinate transformations is to use the interpolation functions that are used to represent the variation of the unknown function.

That is, the global coordinates (x, y, z) within an isoparametric element are evaluated from the element nodal coordinates (x_i, y_i, z_i) as

$$x(\xi,\eta,\zeta) = \sum_{i=1}^{n} N_i^e(\xi,\eta,\zeta)\, x_i = [N^e]\{x^e\} \qquad (22)$$

and similiarly for y and z, so that we have

$$[x, y, z] = [N^e] [x^e, y^e, z^e] = [N^e] [X^e] \quad (23)$$

In other words, the geometry and the unknowns of the element are described in terms of the same parametric equations and are of the same order. Elements such as these are called isoparametric elements.

Zienkiewicz [9] states the following two theorems for the compatibility requirements of the isoparametric element.

Theorem 1. If two adjacent elements are generated from parents in which the same interpolation functions satisfy continuity requirements then the distorted elements will be contiguous.

Theorem 2. If the interpolation functions are such that continuity of ϕ is preserved in the parent coordinates, then continuity requirements will be satisfied in distorted elements.

Thus if the chosen interpolation function satisfies the compatibility requirements for the parent element, this necessary convergence condition will be preserved for the transformed, isoparametric element.

A wide choice of functions which satisfies the compatibility criteria could be made to define the geometry and variation of ϕ. However, the satisfaction of the constant derivative criteria imposes some limitation.

Using the relation which states that the sume of the interpolation functions is unity, completeness is shown to be preserved [9].

The "linear" isoparametric quadrilateral. As an example of some of the features of an isoparametric element consider the mapping of a quadrilateral with nodes at the four corners, as shown in Figure 5.

The following interpolation functions have been developed in local coordinates for any point i of this element [9].

$$N_i = (1 + \xi\,\xi_i)\,(1 + \eta\,\eta_i)/4 \qquad (24)$$

Note that along an edge of the element these interpolation functions become linear.

Since ϕ varies linearly on the edges, the element is called a "linear" isoparametric quadrilateral although the interpolation functions are not linear in general. It can also be shown that the element has straight line boundaries in the global system [2].

For future reference note that the local derivatives of the interpolation functions are

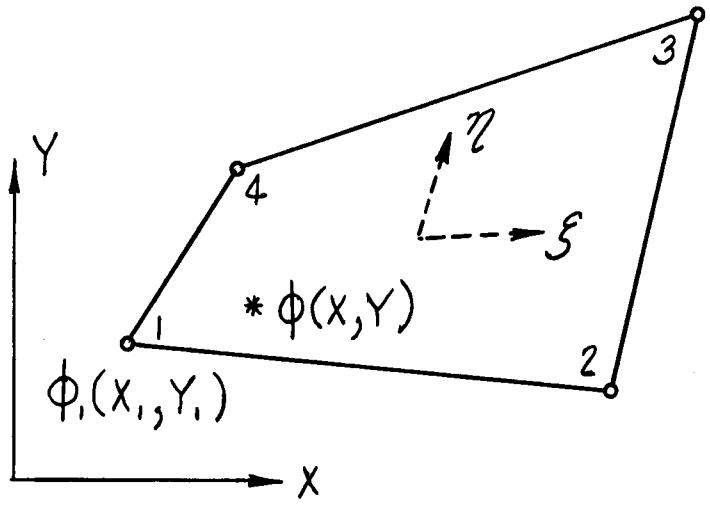
$$\frac{\partial N_i^e}{\partial \xi}(\xi, \eta) = \xi_i\,(1 + \eta\,\eta_i)/4$$

and (25)

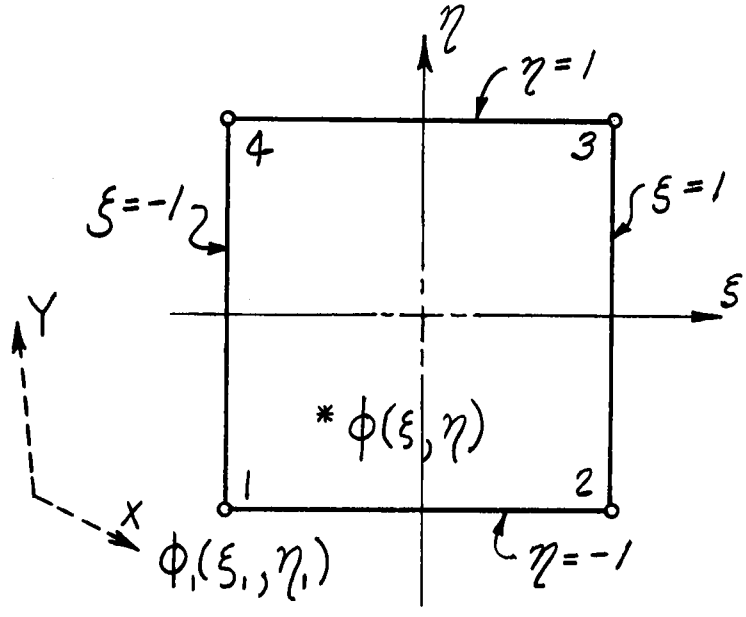
$$\frac{\partial N_i^e}{\partial \eta}(\xi, \eta) = (1 + \xi\,\xi_i)\,\eta_i/4$$

which are the necessary relations required to establish the relationships between the local and global coordinate systems.

Transformation of derivatives. To perform the finite element analysis the matrices defining the element properties have to be found. We have shown these to be of the form



a. Global Coordinates



b. Local Coordinates

Figure 5. Two-dimensional mapping of an isoparametric quadrilateral with four nodes.

$$\int_V [G]\, dV \tag{26}$$

in which the expression $[G]$ is usually formulated in terms of the derivatives of the unknown variable ϕ with respect to the global system.

To evaluate such matrices, two transformations are necessary. Firstly, as $[N]$ is defined in terms of local coordinates it is necessary to devise some means of expressing the global derivatives in terms of local derivatives.

Secondly the element of volume (or surface) over which the integration has to be carried out needs to be expressed in terms of the local coordinates with an appropriate change of limits of integration.

Transformation of the derivatives can be accomplished in the following manner. By the usual rules of partial differentiation we can write

$$\frac{\partial \phi}{\partial \xi} = \frac{\partial \phi}{\partial x} \cdot \frac{\partial x}{\partial \xi} + \frac{\partial \phi}{\partial y} \cdot \frac{\partial y}{\partial \xi} + \frac{\partial \phi}{\partial z} \cdot \frac{\partial z}{\partial \xi} \tag{27}$$

Performing the same differentiation with respect to the other two coordinates, we have

$$\begin{Bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \\ \frac{\partial \phi}{\partial \zeta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \begin{Bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \\ \frac{\partial \phi}{\partial z} \end{Bmatrix} \tag{28}$$

or

$$\left\{\partial\phi_l\right\} = [J(\xi, \eta, \mathcal{J})]\ \left\{\partial\phi_g\right\} \tag{29}$$

where $[J]$ is known as the Jacobian matrix and where the subscript g is introduced to represent global derivatives.

The global derivatives are thus related to the local derivatives

$$\left\{\partial\phi_g\right\} = [J]^{-1}\left\{\partial\phi_l\right\} \tag{30}$$

In practice one can evaluate the Jacobian matrix at a point i within a typical element by differentiating the expression given by (23) with respect to the local coordinates. That is

$$\left[\frac{\partial x_i}{\partial \xi}, \frac{\partial y_i}{\partial \xi}, \frac{\partial z_i}{\partial \xi}\right] = \left[\frac{\partial N^e}{\partial \xi}\right]\left[x^e, y^e, z^e\right] \tag{31}$$

and similarily for η and $\mathcal{J}$, so that

$$\begin{bmatrix} \frac{\partial x_i}{\partial \xi} & \frac{\partial y_i}{\partial \xi} & \frac{\partial z_i}{\partial \xi} \\ \frac{\partial x_i}{\partial \eta} & \frac{\partial y_i}{\partial \eta} & \frac{\partial z_i}{\partial \eta} \\ \frac{\partial x_i}{\partial \mathcal{J}} & \frac{\partial y_i}{\partial \mathcal{J}} & \frac{\partial z_i}{\partial \mathcal{J}} \end{bmatrix} = \begin{bmatrix} \frac{\partial N^e}{\partial \xi} \\ \frac{\partial N^e}{\partial \eta} \\ \frac{\partial N^e}{\partial \mathcal{J}} \end{bmatrix} \left[x^e, y^e, z^e\right] \tag{32}$$

or

$$[J_i] = [\Delta]\ [x^e] \tag{33}$$

Having (numerically) evaluated the Jacobian one has a practical means for evaluating the important relation that defines the global derivatives of $\{\phi\}^e$ in terms of the nodal values. From (20) and (30) we can write

$$\left\{\partial\phi_g\right\} = [J]^{-1}\ [\Delta]\ \left\{\phi\right\}^e = [D^e]\left\{\phi\right\}^e \tag{34}$$

where the matrix

$$[D^e] = [J]^{-1} [\Delta] \tag{35}$$

is the desired expression that defines the global derivatives of $[N]$ at any point in the element. That is

$$[D^e] = \begin{bmatrix} D_x{}^e \\ \cdots \\ D_y{}^e \\ \cdots \\ D_z{}^e \end{bmatrix} = \begin{bmatrix} \frac{\partial N}{\partial x} \\ \cdots \\ \frac{\partial N}{\partial y} \\ \cdots \\ \frac{\partial N}{\partial z} \end{bmatrix} \tag{36}$$

In practice the Jacobian, $[J]$, exist in a numerical form and thus $[D^e]$ is also defined numerically. Equations (18) and (34) are important as they usually appear in an integral definition of an element matrix.

To transform the variables and the region with respect to which the integration is made, a standard process will be used which involves the determinant of $[J]$. Thus for instance a volume element

$$dxdydz = \det [J] \, d\xi \, d\eta \, d\zeta \tag{37}$$

This type of transformation is valid irrespective of the number of coordinates used [9].

Element Matrix--The General Three-Dimensional Case

The element matrix for an isoparametric element will now be established.

Recalling (17), we have for the first term of $s_{ij}{}^e$

$$\int_{V^e} \left\{ k_x \frac{\partial N_i}{\partial x} \cdot \frac{\partial N_j}{\partial x} \right\} \, dxdydz \tag{38}$$

where nodes i and j denote the subscripts of the element matrix coefficient.

The first term of the element matrix $[s]^e$ is thus

$$\int_{V^e} [k_x] \left[\frac{\partial N}{\partial x}\right]^T \left[\frac{\partial N}{\partial x}\right] \, dxdydz \tag{39}$$

Applying (22), (36), and (37), the above relation expressed in the local coordinate system becomes

$$\int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} k_x(\xi, \eta, \zeta) \left[D_x^e(\xi, \eta, \zeta)\right]^T \left[D_x^e(\xi, \eta, \zeta)\right] \det \left[J(\xi, \eta, \zeta)\right] d\xi d\eta d\zeta \tag{40}$$

Similar expressions may be stated for the other terms of $[s]^e$.

The element matrix is now in a form suitable for Gaussian integration.

III. NUMERICAL INTEGRATION

Using relations developed in the preceding sections, the formulation and procedures for the numerical evaluation of the isoparametric element matrix will be outlined for the general three-dimensional case.

The principles developed here will then be applied to the specific problem analyzed for this investigation; the two-dimensional,

steady state, heat conduction problem utilizing linear isoparametric quadrilateral elements.

Some principles of numerical integration for one dimensional and rectangular regions together with a table showing the abscissae and weight coefficients for two, three, and four point Gaussian quadrature are found in Appendix A.

General Three-Dimensional Case

We have seen that for an isoparametric element a typical term of the element matrix is defined by (40).

We will integrate using Gaussian quadrature. It is well-known [9] that Gaussian quadrature integrates a polynomial of degree 2Q - 1 exactly where Q is the number of Gauss points (see Appendix A).

Thus by using Q Gauss points in each dimension, we have for the approximate solution of (40),

$$\sum_{i=1}^{Q}\sum_{j=1}^{Q}\sum_{k=1}^{Q} k_x(\alpha_i,\alpha_j,\alpha_k)\ \left[D_x^e(\alpha_i,\alpha_j,\alpha_k)\right]^T \left[D_x^e(\alpha_i,\alpha_j,\alpha_k)\right]\ \det\left[J(\alpha_i,\alpha_j,\alpha_k)\right]\omega_i\,\omega_j\,\omega_k \qquad (41)$$

where $\alpha_i, \alpha_j, \alpha_k$ and $\omega_i, \omega_j, \omega_k$ are the tabulated abscissae and weight coefficients for Gaussian quadrature, at the integration point denoted by the indices i, j and k [2].

This can be written as a single sum.

$$\sum_{n=1}^{I} k_x \left(\begin{matrix} \vdash\!\dashv \\ - \\ \vdash\!\dashv \end{matrix} \right) \left[D_x{}^e \left(\begin{matrix} \vdash\!\dashv \\ - \\ \vdash\!\dashv \end{matrix} \right) \right]^T \left[D_x{}^e \left(\begin{matrix} \vdash\!\dashv \\ - \\ \vdash\!\dashv \end{matrix} \right) \right] \det \left[J^e \left(\begin{matrix} \vdash\!\dashv \\ - \\ \vdash\!\dashv \end{matrix} \right) \right] W_n \tag{42}$$

where the number of integration points is

$$I = Q^s \tag{43}$$

with s being the spatial dimension (s = 3 for the general case under investigation), and the integration point n is indexed by

$$n = j + (i - 1)\, Q + (k - 1)\, Q^2$$

with

$$W_n = \omega_i\, \omega_j\, \omega_k \tag{44}$$

as the calculated <u>weight function</u>, and

$$\left[\begin{matrix} \vdash\!\dashv \\ - \\ \vdash\!\dashv \end{matrix} \right] = [\alpha_i, \alpha_j, \alpha_k] \tag{45}$$

the local coordinate matrix.

Later it will be useful to note that the latter two quantities are submatrices of

$$\{W\} = \begin{Bmatrix} W_1 \\ W_2 \\ \cdot \\ \cdot \\ \cdot \\ W_I \end{Bmatrix} \tag{46}$$

and

$$\begin{bmatrix} \vec{r} \\ - \\ \vec{r} \end{bmatrix} = \begin{bmatrix} \vec{r} \\ - \\ \vec{r}_1 \\ \cdots \\ \vec{r} \\ - \\ \vec{r}_2 \\ \cdots \\ \vdots \\ \vdots \\ \vdots \\ \cdots \\ \vec{r} \\ - \\ \vec{r}_I \end{bmatrix} \tag{47}$$

From (43), (35) and (33) it is noted that at each integration point, one must evaluate the local coordinates, $\begin{bmatrix} \vec{r} \\ - \\ \vec{r}_n \end{bmatrix}$, the weighting function, W_n, and the local derivatives matrix, $\left[\Delta\left(\begin{matrix} \vec{r} \\ - \\ \vec{r}_n \end{matrix}\right)\right]$, in order to evaluate the element matrices.

Example--Two-Dimensional Steady State Heat Conduction

Consider a two-dimensional steady state heat conduction problem where (only) the temperature is given on parts of the boundary. The general governing equation (1) becomes the Laplace equation

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial \phi}{\partial y}\right) = 0 \tag{48}$$

where as before, ϕ is the temperature and k_x and k_y are the material thermal conductivities.

The associated functional (4) is

$$\chi = 1/2 \int_V \left[k_x \left(\frac{\partial \phi}{\partial x}\right)^2 + k_y \left(\frac{\partial \phi}{\partial y}\right)^2\right] dxdy \tag{49}$$

and by (40), the element matrix for an isoparametric element becomes

$$[s]^e = \int_{-1}^{1}\int_{-1}^{1} k_x(\xi,\eta)\,[D_x(\xi,\eta)]^T\,[D_x(\xi,\eta)]\,\det[J(\xi,\eta)]$$

$$d\xi d\eta + k_y(\xi,\eta)[D_y(\xi,\eta)]^T\,[D_y(\xi,\eta)]\,\det[J(\xi,\eta)]\,d\xi\,d\eta \tag{50}$$

For an isoparametric quadrilateral with four nodes, we note from (33) that each of $[J]$, $\det[J]$, and $[\Delta]$ are at most, linear functions of ξ and η. Thus assuming k_x and k_y constant, the integrand involves polynomials which are, at most, fifth order in ξ and η. Therefore for exact integration [9], three Gauss points in each dimension are required, which gives nine integration points (44).

The element matrix can thus be approximated as

$$[s]^e \cong \sum_{n=1}^{9} W_n(k_x[D_{x_n}]^T\,[D_{x_n}] + k_y\,[D_{y_n}]^T\,[D_{y_n}])\,\det[J_n] \tag{51}$$

where

$$[D_{x_n}] = [D_x(\alpha_i,\alpha_j)] \tag{52}$$

and

$$n = j + (i-1)3 \qquad [i \leq i,\ j \leq 3] \tag{53}$$

with

$$W_n = \omega_i\,\omega_j \tag{54}$$

CHAPTER IV

PROGRAMMING CONSIDERATIONS

I. INTRODUCTION

The general purpose of the element routine is the creation of the element matrix appropriate to the problem. It usually has all necessary data transmitted to it through common storage and passes the element matrix back to the calling program, or writes it on peripheral storage. It is highly dependent upon the basic mathematical formulation required to develop the element matrix, and in particular the numerical integration required for the isoparametric element.

This chapter is therefore concerned with improvements of the basic programming techniques commonly used in the evaluation of the isoparametric element matrix.

We will first examine these programming procedures and then propose possible improvements to the computational efficiency. Finally, we will discuss various FORTRAN optimization techniques used to increase the computational speed.

Estimated storage and calculation requirements for each modification will be compared with those required by the common programming procedures. The results will be tabulated for 200 and 1001 elements.

II. COMMON PROGRAMMING PROCEDURES

The following is an outline of the programming steps that are commonly used to numerically evaluate the matrices for an isoparametric element.

Steps Required to Call the Element Routine

1. Define subroutines to evaluate the interpolation functions, $[N]$, and their local derivatives $[\Delta]$ for any given values of the local coordinates.

2. Select the number of Gauss points Q to be used in each dimension, S, and calculate the total number of integration points, $I = Q^S$.

Steps Required for Each Element Routine

Initial steps. 1. Pass into the routine the global coordinates of the element, $[X^e]$, the total number of integration points I, and the tabulated Gauss weight and abscissae coefficients $\{\omega\}$ and $\{\alpha\}$.

2. Using the Gaussian coefficients, calculate the weighting function, W_m, and local coordinate abscissae $\begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix}_m$, of each integration point m, $1 \leq m \leq I$.

3. Initialize (zero) the element matrices, $[s]^e$ and $\{t\}^e$.

Steps required for each integration point. 1. Utilizing the local coordinates of the integration point, $\begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix}_m$, evaluate the local derivatives of the interpolation functions $\left[\Delta\left(\begin{matrix} \xi \\ \eta \\ \zeta \end{matrix}\right)_m\right]$, and (if needed) also evaluate the interpolation functions, $\left[N\left(\begin{matrix} \xi \\ \eta \\ \zeta \end{matrix}\right)_m\right]$.

2. Utilize the global coordinates of the nodes, $[X^e]$ to evaluate the Jacobian matrix at the integration point.

$$\left[J^e\left(\begin{matrix}\rightarrow\\ -\\ \rightarrow\end{matrix}\right)\right] = \left[\Delta\left(\begin{matrix}\rightarrow\\ -\\ \rightarrow\end{matrix}\right)\right]_m [X^e] \tag{55}$$

3. Evaluate the inverse, $[J_m^e]^{-1}$ and determinant, $|J_m^e|$ of the Jacobian.

4. Calculate the relation, $[D_m^e]$ between the global derivatives and the element nodal parameters

$$[D_m^e] = [J_m]^{-1} [\Delta_m]$$

$$\text{(Recall } \left\{\partial \phi_g^e \left(\begin{matrix}\rightarrow\\ -\\ \rightarrow\end{matrix}\right)\right\}_m = [D_m^e] \{\phi\}^e \tag{56}$$

5. Evaluate the matrix products and summations within the integrand. Generally this will involve products of $[N_m^e]$ and/or the partitioned sub-matrices of $[D_m^e]$ (e.g., the three-dimensional case (41) would give

$$[\bar{s}_m] \equiv [D_{x_m}]^T [D_{x_m}] + [D_{y_m}]^T [D_{y_m}] + [D_{x_m}]^T [D_{z_m}] \tag{57}$$

6. Multiply the matrix integrand by the determinant of the Jacobian and by the weighting function, adding the result to the element matrix.

(e.g. $[s]^e = [s]^e + [\bar{s}_m] |J_m^e| W_m$)

After all the above steps have been performed, the numerically integrated element matrix is obtained.

III. PROPOSED PROGRAMMING PROCEDURES

Two suggested methods of improving the computational efficiency will be outlined in this section.

In order to estimate the modification effects, we define a typical "operation" or "calculation" as one of the following:

1. a x b
2. a + b
2. a = b

where a, b are constants or any subscripted variables. A "storage location" will be the same as a conventional core storage location or register. Here no distinction is made between SINGLE or DOUBLE PRECISION storage.

First Modification--Calculation of Gaussian Coefficients

Note that since the weights and abscissae of the integration points are calculated from tabulated data these quantities will be the same for all elements that have the same number of integration points in each dimension. Usually this will be all of the elements in the system.

Thus the matrices $\{W\}$, and $\begin{bmatrix} \not\vdash \\ - \\ \not\vdash \end{bmatrix}$ could be calculated initially and passed to each element routine for use in the numerical integration. If $I = Q^S$ denotes the total number of element integration points, S the dimension of the space, and K the total number of elements, then the proposed procedure would require (I x 1 + I x S)

storage locations and approximately (S! x I + S x I) operations. By way of comparison, the present procedure of calculating each W_m and $\begin{bmatrix} \\ - \\ \\ \end{bmatrix}_m$ for each element requires (1 + S) storage locations and ((S! x I + S x I) x K) operations. Therefore this change saves approximately (K - 1) Q^S (S + S!) calculations at the expense of (Q^S - 1) (1 + S) storage locations.

Note that the storage cost is fixed but the computational savings is directly proportional to the number of elements. Table I shows some typical effects of this modification.

Second Modification--Evaluation of Local Derivatives

The second proposed modification is concerned with the first step at each integration point, the evaluation of the local derivatives of the interpolation function. The local derivative matrix $[\Delta]$ is needed to calculate the Jacobian matrix and the global derivative matrix. Recall that $[\Delta]$ depends only on the local coordinates of the point of interest. In the present procedure this matrix is evaluated at every integration point in every element. However, it was pointed out in the first modification proposal that the local coordinates of the integration points $\begin{bmatrix} \\ - \\ \\ \end{bmatrix}$ need only to be determined once in in the initial phase of the program (or stored in a data block).

Thus an alternate procedure is to evaluate and store (or place in a data block) a matrix $[\bar{\Delta}]$ (and/or $[\mathbf{N}]$) where

TABLE I

ESTIMATED EFFECTS OF FIRST MODIFICATION

Dimension of Space	No. of Gauss Points	Storage Locations Penalty	Calculation Savings	
			200 Elements	1001 Elements
2	2	9	3,184	16,000
2	3	24	7,164	36,000
2	4	45	12,736	64,000
3	2	28	6,368	72,000
3	3	104	31,492	243,000
3	4	252	50,944	576,000

$$[\bar{\Delta}] \equiv \begin{bmatrix} \Delta_1 \\ \cdots \\ \Delta_2 \\ \cdots \\ \vdots \\ \cdots \\ \Delta_I \end{bmatrix} \tag{58}$$

contains one sub-matrix, $[\Delta_m]$, for each integration point, where $m = (1, 2, \ldots I)$. This procedure calculates each local derivative matrix once, but it requires that all the matrices be stored.

Let C_L denote the number of calculations required to evaluate $[\Delta_m]$ at a single point, and L the number of degrees of freedom per element. Then the proposed procedure would require $C_L \times I$ calculations and $(S \times I) \times L$ storage locations. By way of comparison the present procedure requires $C_L \times I \times K$ calculations and only $S \times L$ storage locations. Thus an additional $(I - 1)(S \times L)$ storage locations save $(K - 1)(C_L \times I)$ calculations.

Again the storage cost is fixed but the computational savings is directly proportional to the number of elements, K. The estimated effects of this modification are shown in Table II for typical situations.

IV. FORTRAN OPTIMIZATION

Even the most efficient finite element computer codes require a relatively large amount of computer memory and time. Increased

TABLE II

ESTIMATED EFFECTS OF SECOND MODIFICATION

Dimension of Space	No. of Gauss Points	L[a]	Storage Locations Penalty	Calculation Savings[b] for 200 and 1001 Elements
2	2	4	24	4
2	3	4	64	9
2	4	4	120	16
2	2	8	48	4
2	3	8	128	9
2	4	8	240	16
3	2	8	168	8
3	3	8	624	27
3	4	8	1512	64
3	2	20	420	8
3	3	20	1560	27
3	4	20	3780	64

[a]Degrees of freedom per element.

[b]Multiply by $199C_L$ and C_L10^3 for 200 and 1001 elements respectively.

accuracy resulting in decreasing the size of the element and/or increasing the number of nodal points is offset by an increase in computation time. As the element routine (and usually other routines) must be called for <u>each</u> element, it is essential that FORTRAN optimization techniques be employed. Such optimizing techniques will be applied as a final effort to improve the computational efficiency.

Note that as a result of the previous modifications, most specialized subprograms will no longer be called by the element routine. Included are those required to evaluate $\{W\}$, $\left[\overleftrightarrow{=}\right]$ and $[\Delta]$. The remaining subprograms used to determine $[J]$, $\det [J]$, $[J]^{-1}$, $[D^e]$ and $[s]^e$ are usually of a "standard" type such as matrix addition, multiplication, inversion, etc. When the order of the matrix is small (≤ 4) it is convenient and usually more efficient to list these subprograms as in-line source code.

We will therefore investigate the effect of removing <u>all</u> remaining SUBROUTINES invoked by the element routine.

A further extension of this scheme would be the in-line expansion of all subprograms called by the assembly SUBROUTINE for <u>each</u> element. This would, of course, eliminate the element routine. This proposal will also be investigated.

Other optimizing techniques will be utilized such as the use of EQUIVALENCE statements, minimization of the number of evaluations of an expression, and the reference to an N - 1 dimensional array rather than one having N dimensions. Several other miscellaneous optimizations will be made.

The above mentioned techniques and others are described in [4] and [6]. They may also be found in most FORTRAN manuals.

CHAPTER V

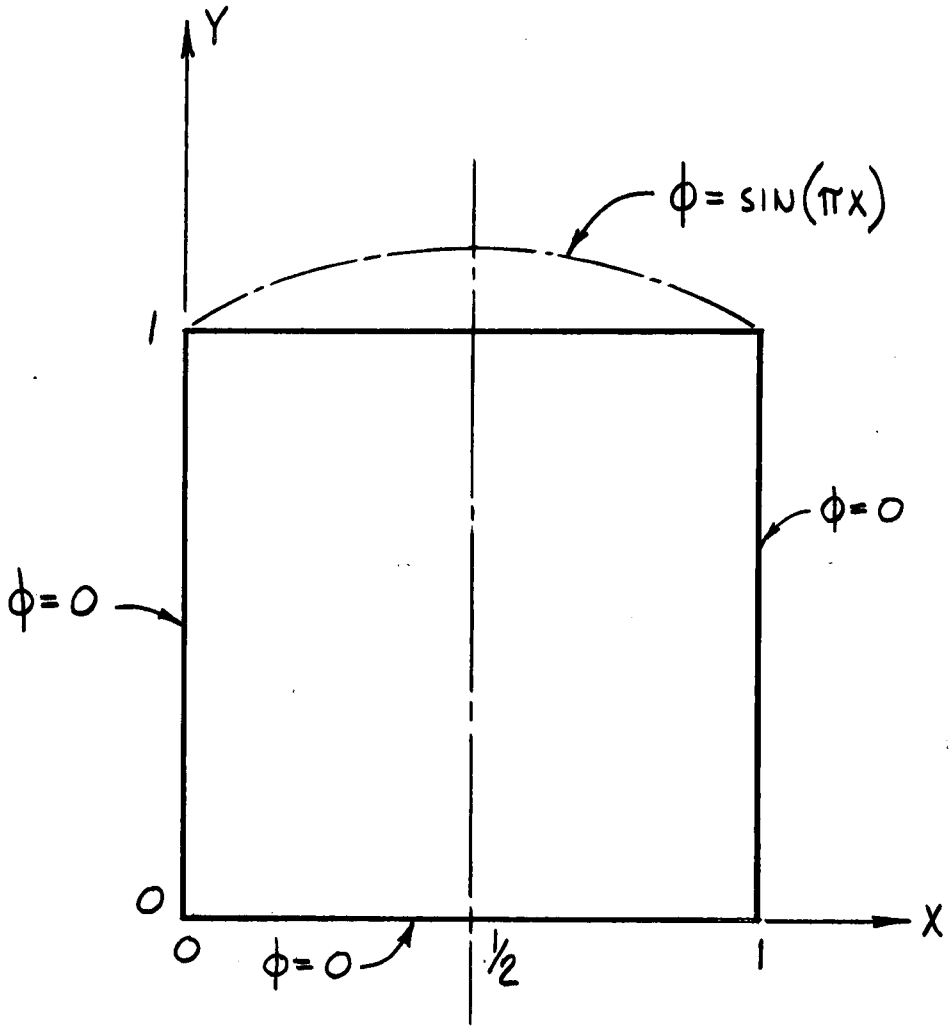
MODIFICATIONS AND NUMERICAL RESULTS

I. INTRODUCTION

The effects of the modifications proposed in Chapter IV were tested by two methods. The first studied the effect on the overall computational efficiency of a finite element program; the second was concerned with the efficiency of element matrix integration.

All modifications occurred in the first or program testing phase and were performed on the element and assembly routines and their subprograms.

For analysis a temperature profile was made of a steady state, two-dimensional heat conduction problem in a unit square plate (Figure 6). The temperature was specified by the function $\phi = \sin(\pi x)$ on the edge $y = 1$ and zero on the other boundaries. The anisotropic conductivity factors k_x and k_y were assumed to be unity, and the heat generation function Q was zero. Due to symmetry, only the left half of the plate was used. The temperature flux at the top surface was specified as temperatures at the nodal points and therefore the boundary conditions were such that the element column matrix $\{t\}^e$ (10) was zero. The region was subdivided into 200 linear isoparametric quadrilateral elements, each having four corner nodes and numbered such



$k_x = 1$

$k_y = 1$

$Q = 0$

Exact solution:

$$\phi(x,y) = \sin(\pi x)\ \sinh(\pi y)/\sinh \pi$$

Figure 6. Steady state heat conduction in a unit square plate.

that the resulting half band width of the final system of algebraic equations was thirteen [2]. The exact solution of this problem is $\phi(x,y) = \sin(\pi x) \sinh(\pi y)/\sinh\pi$ and was used to check the program [2].

Since the results obtained in the above study reflected the overall program performance, it was desirable to investigate the integration performance alone. This resulted in the development of two programs. The first made reference to the element routine used in the original program; the second used that portion of the final version used exclusively for element integration. In order to obtain a better representation of the effects, an arbitary element was used 1001 times.

For the purpose of this investigation, two, three and four point Gaussian quadrature was used. Three point quadrature, which was shown to be required for the exact integration of the subject element, will be used in the discussions.

Programs were prepared in FORTRAN for the IBM System/360-50. A FORTRAN IV (G) compiler was used and the observed run times correspond to the compiler GO step; the link, edit, GO (FORTGCL9) procedure. The GO step executes the load module (program) and thus represents the actual execution time.

However this time can vary for multiple runs. This is due primarily to a variety of possible paths available for compiler Microinstructions. Therefore, three runs for each change were made and the average times obtained. All observed times and averages are

tabulated, and are expressed in minutes and seconds (i.e. 1-54.49 would read 1 min. 54.49 sec.). Unless otherwise specified all times are compared with the original version of the programs.

The required and assigned core storages are noted for each program revision and compared with earlier estimates. DOUBLE PRECISION floating point arithmetic, which requires eight bytes of storage per word, was used exclusively. The printout did not specify the number of operations so no attempt was made to correlate the previous calculation estimates.

As a means of establishing the efficiency of a programming change, an "efficiency factor," E, is introduced. Expressed as a fraction

$$E = \frac{\tau_o}{\tau} \times \frac{C_o}{C} \tag{59}$$

where τ, and C are the observed run time and core requirement respectively. τ_o and C_o correspond to the original values used for comparison. Thus the datum (original) efficiency is unity.

Table III shows the relationship between formula symbols and FORTRAN names appearing in the assembly routine ASYMBL and/or the element routine ELSQ. Note that in some cases, several names represent the same symbol.

A listing and description of all SUBROUTINES and developed programs involved may be found in Appendix B.

In order to determine the influence of each program change, run times and storages were established for the original program.

TABLE III

RELATIONSHIP BETWEEN SYMBOLS AND FORTRAN NAMES

Formula Symbol	FORTRAN Name	Description
S	NSPACE	Spatial dimension
i	N	Number of element degrees of freedom
Q	NGP, NITG	Number of Gauss points in each dimension
n	IP, K	Integration point
I	NIP, NUMPTS	Total number of integration points
α_k	A, GPT	Gaussian abscissa coefficients
ω_k	W, GWT	Gaussian weight coefficients
W_n	WT	Weighting function
α_i	XI, ZETAPT	Local ξ coordinate of integration point
α_j	ETA, ETAPT	Local η coordinate of integration point
$[X^e, Y^e]$	COORD	Element global coordinates
$[\bar{\Delta}]$	DEL	Local derivatives for all integration points
$[\Delta_n]$	DELTA	Local derivatives
J_n, J_n^{-1}	AJ, AJJ	Jacobian and Jacobian inverse
$\det[J_n]$	DET	Determinant of Jacobian
$[D^e]$	DE, GLOBAL	Global derivatives
k_x, k_y	XXK, YYK	Anisotropic conductivity factors
$[s]^e$	S	Element matrix

The times are shown in Table IV, and the required and assigned core storages were 122,760 and 132,000 bytes respectively.

II. PROGRAM EFFICIENCY

First Modification

To apply the first proposed modification, it was necessary to place the following Gaussian data expression in the assembly routine, ASYMBL,

```
      NGP = 3
      CALL GAUSCO (NGP,A,W)
      IP = 0
      DO 1 I = 1, NGP
      DO 1 J = 1, NGP
      IP = IP + 1
      WT(IP) = W(I)*W(J)
      XI(IP) = A(I)
      ETA(IP) = A(J)
1     CONTINUE
```
(60)

and remove the equivalent from the element routine ELSQ. This information was passed to ELSQ by adding NGP, WT, XI, and ETA to the arguments, thereby increasing the calling time.

This version required 122,984 bytes of storage, an increase of 224 bytes which agrees somewhat with an earlier maximum estimate of 360 (based on four point quadrature). The assigned storage increased by 2000 bytes.

Run times for this modification are shown in Table V.

The average time savings for two and three point integration were 1.81 and 1.04 seconds respectively. An <u>increase</u> of 1.98 seconds was evidenced by four point quadrature.

TABLE IV

FINITE ELEMENT PROGRAM (200 ELEMENTS)--RUN TIMES, ORIGINAL VERSION

No. of Gauss Points	Run 1	Run 2	Run 3	Average Time
2	0-47.22	0-48.04	0-51.80	0-49.02
3	1-23.67	1-12.45	1-15.37	1-17.16
4	1-59.49	1-54.45	1-50.69	1-54.88

TABLE V

FINITE ELEMENT PROGRAM (200 ELEMENTS)--RUN TIMES, FIRST MODIFICATION

No. of Gauss Points	Run 1	Run 2	Run 3	Average Time
2	0-49.00	0-46.39	0-45.95	0-47.11
3	1-15.10	1-12.14	1-21.12	1-16.12
4	2-2.34	1-57.90	1-50.34	1-56.86

The resulting time changes were rather inconclusive, and yet additional storage was required. It is presumed that the additional arguments required to call the element routine nullified the expected time savings.

Second Modification

The second proposal requires that the local derivatives, DEL, be determined in routine ASYMBL, stored, and passed to the element routine.

As both modifications are concerned with properties common to each of the elements, it was convenient to list both in a newly created SUBROUTINE named ELCOM and pass the information to the element routine ELSQ.

Thus the expression (60) (except the statement NGP = 3) and the following code comprised the body of SUBROUTINE ELCOM.

```
      NIP = IP
      DO 2 IP = 1, NIP
      CALL DER4Q(XI(IP), ETA(IP), DELTA)
      DO 2 J = 1, 2
      DO 2 M = 1, 4                                    (61)
      DEL(J,M,IP) = DELTA (J,M)
    2 CONTINUE
      RETURN
      END
```

To obtain the local derivatives, DELTA, at each integration point, IP, the following was added to routine ELSQ.

```
      DO 5 I = 1, NSPACE
      DO 5 J = 1, N
      DELTA (I,J) = DEL (I,J,IP)                       (62)
    5 CONTINUE
```

The storage allotment was the same as before but an additional 1784 bytes were used. Note that in Table II, page 37, the core space required by DEL for expression (62) was not considered. Taking this into account we have an additional 128 words of storage resulting in a total estimated penalty of 1984 bytes, an amount slightly higher than that used.

The effects of the second modification are shown in Table VI.

In contrast with the first modification, two and three point Gauss quadrature showed an increase in time and four point quadrature showed a decrease, but in both cases the time changes discernable were minimal and additional storage was required.

Although the arguments of ELSQ have been reduced in number, the addition of (62) apparently offset any gain in efficiency.

Final Optimized Version

The FORTRAN optimizing techniques suggested in Chapter III were applied for the final revision. Due to the nature and complexity of the changes, no attempt was made to estimate either the storage or calculation requirements.

The first and perhaps major step was the elimination of most of the SUBROUTINES called for each element by the assembly routine ASYMBL. Those excluded were routines LPTPRT and ELPOST which were not used for this investigation. Note also that ELCOM was left undisturbed, as it was only called once. The element column matrix

TABLE VI

FINITE ELEMENT PROGRAM (200 ELEMENTS)--RUN TIMES, SECOND MODIFICATION

No. of Gauss Points	Run 1	Run 2	Run 3	Average Time
2	0-46.24	0-50.25	1-11.29	0-55.92
3	1-17.84	1-21.55	1-45.75	1-28.38
4	1-57.64	1-53.80	1-48.95	1-53.46

routine ELCOL did not apply for this application ($\{t\}^e = 0$) and was omitted.

The above procedure was easily implemented since most of the routines consisted of a few simple expressions. Some expressions were combined. For example, noting that the outer DO loop in routine ELCORD has the same index limit as the single loop in LNODES, the element indices expression

$$\text{LNODE (I)} = \text{NODES (IE, I)} \tag{63}$$

was placed in the outer loop of ELCORD. A similar procedure was employed for routines STORSQ and STORCL. Here, the insertion of

$$\text{CC(I)} = \text{C(L)} + \text{CC(I)} \tag{64}$$

in the outer loop of STORSQ eliminated calculation of I = INDEX(L) twice for each system degree of freedom.

An additional improvement was made for routine INDEXL. Note that the values NG*(K-1) and NG*(LNODE(K)-1) are determined within the inner loop for each nodal parameter number IG. These expressions were placed in the outside loop indexed by K, the nodal number. However no savings were expected for the subject element since NG = 1.

Although the element routine ELSQ invoked other routines, this did not present a problem with in-line coding. For instance, SUBROUTINE ZERO which initializes the element matrix, S, was simply listed. The remaining matrix operation routines were easily eliminated due to the relatively small orders involved. Thus the first call for the multiplication routine MMULT to evaluate the Jacobian (AJ = DELTA * COORD) at each integration point was replaced by

```
      DO 30  I = 1, NSPACE
      DO 25  J = 1, NSPACE
      SUM = 0.0
      DO 20  K = 1, N
      SUM = SUM + DEL(I,K,IP)*COORD(K,J)                    (65)
20    CONTINUE
      AJ(I,J) = 0.0
25    CONTINUE
30    CONTINUE
```

Note that both arithmetric IF statements of MMULT were removed as they were rarely utilized for this application. Also note the continued usage of the storage register SUM in order to avoid unnecessary subscript computations for AJ. Observe that the array DEL may be used directly, thereby eliminating expression (62).

By using an EQUIVALENCE statement to overlap AJJ and AJ, and the "element successor" rule [7], the determinant of the Jacobian (DET) was expressed as

```
DET = ((AJJ(1)*AJJ(4)) - (AJJ(3)*AJJ(2)))                   (66)
DET I = 1.0/DET
```

and the Jacobian inverse AJ (array AJ overwritten by J^{-1}) by

```
XII = AJJ(1)
AJJ(1) = AJJ(4)*DET I
AJJ(3) = - AJJ(3)*DET I                                     (67)
AJJ(2) = - AJJ(2)*DET I
AJJ(4) = XII*DET I
```

Note that the reciprocal of DET was evaluated to use the faster multiplication operation in (67).

The listing used to determine the global derivatives (DE = AJ*DELTA), was similar to that used to determine the Jacobian (65). Thus the second call to routine MMULT has been eliminated. Once again DEL was used in lieu of DELTA.

The final revision occurred in the loops required to evaluate the element matrix. Arrays SXX and SYY represented the first and second terms of the element matrix, S. Once the first term was evaluated, it was added to the element matrix by referring to routine MADD, and likewise for the second term, SYY. In both cases, the determinant DET and the weighting function WT(IP) were multiplied within loops for each I and J.

The following simplified expression was used to replace the above procedure:

```
      DETWT = DET*WT(IP)
      DO 70  I = 1, N
      DO 65  J = 1, N                                          (68)
      S(I,J) = DETWT*(XXK*DE(1,I)*DE(1,J)
     1  + YYK*DE(2,I)*DE(2,J)) + S(I,J)
   65 CONTINUE
   70 CONTINUE
```

Not only have the calls to the matrix addition routine been eliminated but also the storage previously required for SXX and SYY.

The effects of these changes are illustrated in Table VII and show that substantial time savings resulted. A core storage of 126,328 bytes was required; an increase of only 3,568 from the original requirement, and 136,000 bytes of storage were allotted.

Thus improved efficiency has been achieved. Here the savings are 14.11, 16.77 and 45.95 seconds for two, three and four Gauss points respectively.

Table VIII summarizes the observations for the four program versions and gives the corresponding efficiency factors.

TABLE VII

FINITE ELEMENT PROGRAM (200 ELEMENTS)--RUN TIMES,
FINAL (OPTIMIZED) MODIFICATION

No. of Gauss Points	Run 1	Run 2	Run 3	Average Time
2	0-35.55	0-36.35	0-32.82	0-34.91
3	0-48.49	0-53.50	0-49.2	0-50.39
4	1-12.62	1-2.80	1-11.44	1-8.93

TABLE VIII

FINITE ELEMENT PROGRAM (200 ELEMENTS)--CORE STORAGE, AVERAGE RUN TIMES AND EFFICIENCY FACTORS (E)

Program Version	Core Storage		No. of Gauss Points					
	Allotment (bytes x 10^3)	Required (bytes)	2		3		4	
			Time	E	Time	E	Time	E
Original	132	122,760	0-49.02	1.0	1-17.16	1.0	1-54.88	1.0
First Modification	134	122,984	0-47.11	1.031	1-16.12	1.002	1-56.86	0.973
Second Modification	134	124,768	0-55.92	0.862	1-28.38	0.859	1-53.46	0.995
Final (Optimized)	136	126,328	0.3491	1.362	0-50.39	1.485	1-8.93	1.619

III. INTEGRATION EFFICIENCY

The first program written exclusively to test integration performance was designed to establish data for the original integration procedures, and consisted of the identification of necessary parameters (including the arbitrary element coordinates) and the call to SUBROUTINE ELSQ. The resulting element matrix was printed for the "first" and "last" element (1 and 1001) as a program check.

The assigned storage was 32,000 bytes and of that amount, 28,536 bytes were used.

Table IX gives the results for this program.

The second program developed was similar to the first except that the call to ELSQ was replaced by the integration sequence as used in the final version of the assembly routine. This required only 27,176 bytes of storage, a reduction of 1360 bytes. The core allotment was unchanged.

The corresponding times are given in Table X. As was expected, substantial savings were achieved.

A summary of the integration efficiences and times are shown in Table XI. Note that for all Gauss points, efficiencies have been greatly increased.

IV. CONCLUDING REMARKS

Due to time variations observed for multiple runs, better results would have been obtained from additional runs. This is especially

TABLE IX

ELEMENT MATRIX INTEGRATION (1001 ELEMENTS)--RUN TIMES, ORIGINAL VERSION

No. of Gauss Points	Run 1	Run 2	Run 3	Average Time
2	1-45.9	1-47.82	2-12.42	1-55.38
3	4-8.05	3-56.27	4-00.37	3-61.56
4	7-16.72	8-04.07	6-59.50	7-26.76

TABLE X

ELEMENT MATRIX INTEGRATION (1001 ELEMENTS)--RUN TIMES, FINAL (OPTIMIZED VERSION)

No. of Gauss Points	Run 1	Run 2	Run 3	Average Time
2	0-40.70	0-34.45	0-46.65	0-40.6
3	1-32.64	1-35.99	1-30.40	1-33.01
4	3-00.04	3-54.17	3-08.24	3-20.82

TABLE XI

ELEMENT MATRIX INTEGRATION (1001 ELEMENTS)--CORE STORAGE, AVERAGE RUN TIMES AND EFFICIENCY FACTORS (E)

Program Version	Core Storage		No. of Gauss Points					
	Allotment (bytes x 10^3)	Required (bytes)	2		3		4	
			Time	E	Time	E	Time	E
Original	32	28,536	1-55.38	1.0	3-61.56	1.0	7-26.76	1.0
Final (Optimized)	32	27,176	0-40.6	2.99	1-33.01	2.72	3-20.82	2.34

apparent from the times obtained from the program developed to test integration efficiency. In this case a variation of 1 min.-4.57 sec. (between runs two and three) was evidenced for the original version using four point quadrature (see Table IX). Time did not permit additional runs to be made for this study.

We have seen that for the subject element, three point quadrature is required for exact integration. However, investigators have proved that for practical work, the minimum number of points is the value that will exactly integrate det[J], [2]. Since this is first order in ξ and η we have $Q_{min} = 1$ so that $Q_{min} \neq 2$ ($Q \neq 1$) being undefined), and thus four integration points gives identical results with three point quadrature. Results of this investigation verify this and show that accuracy was identical for six significant digits. Since the solution time is proportional to $(Q)^s$ [2], the least number of integration points are desirable for practical applications.

CHAPTER VI

CONCLUSIONS

As indicated in Chapter V, integration efficiency was generally not achieved by the proposed modifications. In the first modification, slight increases in efficiency were noted for two and three point quadrature. With the exception of four point quadrature, these values declined for the second modification.

Since the primary objective was to test changes to program logic, no initial attempt was made towards optimization. This no doubt accounted for the trivial results. However when optimizing techniques were later applied, significant improvements were noted. For all Gauss points the efficiency ratings increased. The lowest of these (1.362 for two Gauss points) was considerably higher than those previously obtained. The same was true for the program developed to test only integration performance. Here, the efficiencies were more than doubled.

Thus it can be concluded that the objective of this investigation has been accomplished. The integration efficiency of an isoparametric element has been improved with no sacrifice of accuracy.

The following comments and observations can be made concerning the resulting program.

1. Program versatility has not been damaged. In all cases, the relations unique to a specific element (interpolation functions

and local derivatives) are inserted via the same routine (DE4RQ). Only the point of insertion has been changed.

2. Due to the number of references, the elimination of most routines had perhaps the greatest effect on the improvements. Although this greatly increased the size of routine ASYMBL, the resulting performance more than offsets whatever inconvenience that might result.

3. The efficiency factor established was based on the actual core storage used, and therefore gave actual efficiencies. From a practical standpoint, the efficiency should be determined from the allocated storage space.

Examination of the procedures used suggest a number of areas of further exploration.

1. It would be of interest to use SINGLE PRECISION arithmetic, increase the element size and/or the number of nodal points, and compare with the results of this investigation. Increasing the element size would be of particular interest as the amount of input data is usually a prime economic factor in finite element analysis.

2. All possible optimizing techniques were obviously not investigated. One possibility would be to write the applicable integration SUBROUTINE in machine language, thus offering the programmer an even greater variety of possibilities.

3. No attempt was made to obtain the actual number of calculations (operations) in order to compare with estimates (i.e., C_L, page 38, etc.). This could be achieved by physically counting the

corresponding machine assembly language instructions.

Each of these questions appears to provide additional topics for further work on improvements in the integration efficiency of the isoparametric finite element.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Abel, J. F. and Desai, C. S. Introduction to the Finite Element Method--A Numerical Method for Engineering Analysis. New York: Van Nostrand Reinhold Co., 1971, c 1972.

2. Akin, J. E., Engineering Mechanics 5860 notes, Spring 1973.

3. IBM System/360 and System/370 Fortran IV Language, Tenth edition. New York: IBM Corporation, Programming Publications (October 1972); amended March 1973.

4. IBM System/360 Operation System Fortran IV (G and H) Programmer's Guide, Fourth edition. New York: IBM Corporation, Programming Publications, 1972.

5. Irons, B. M. and Zienkiewicz, O. C. "The Isoparametric Finite Element System--A New Concept in Finite Element Analysis," Conference on Recent Advances in Stress Analysis, Joint Brit. Comm. Stress Analysis, London, March 1968.

6. Larson, Chris. "The Efficient Use of Fortran," Datamation. Los Angeles: Technical Publishing Co., August 1, 1971.

7. McCracken, Daniel D. A Guide to Fortran IV Programming. New York: John Wiley and Sons, Inc., 1965.

8. Zienkiewicz, O. C. and Cheung, Y. K. "Finite Elements in the Solution of Field Problems." The Engineer. London: September 24, 1965.

9. Zienkiewicz, O. C. The Finite Element Method in Engineering Science. London: McGraw Hill Co., 1971.

APPENDICES

APPENDIX A

The Gaussian quadrature formula, which can integrate a polynomial f(x) of degree 2n-1 exactly as a weighted mean of its n particular values at specified points, are used exclusively in finite element analysis because of their high efficiency.

Some principles of Gaussian integration will be summarized here together with tables of convenient numerical coefficients for three, three, and four point quadrature.

One-Dimensional Integration

Consider the n point Gaussian quadrature of a function. We illustrate this in Figure 7 with n = 4. We evaluate

$$I = \int_{-1}^{1} f(\xi)d\xi = \sum_{i=1}^{n} H_i f(\xi_i)$$

where H_i are the weighing coefficients, $f(\xi_i)$ the value of the function at specified Gaussian points ξ_i, and n the number of Gauss points used.

We note that the above example illustrates the exact integration of a seventh order polynomial.

Two-Dimensional Integration--Rectangular or Right Prism Regions

The most obvious way of obtaining the integral

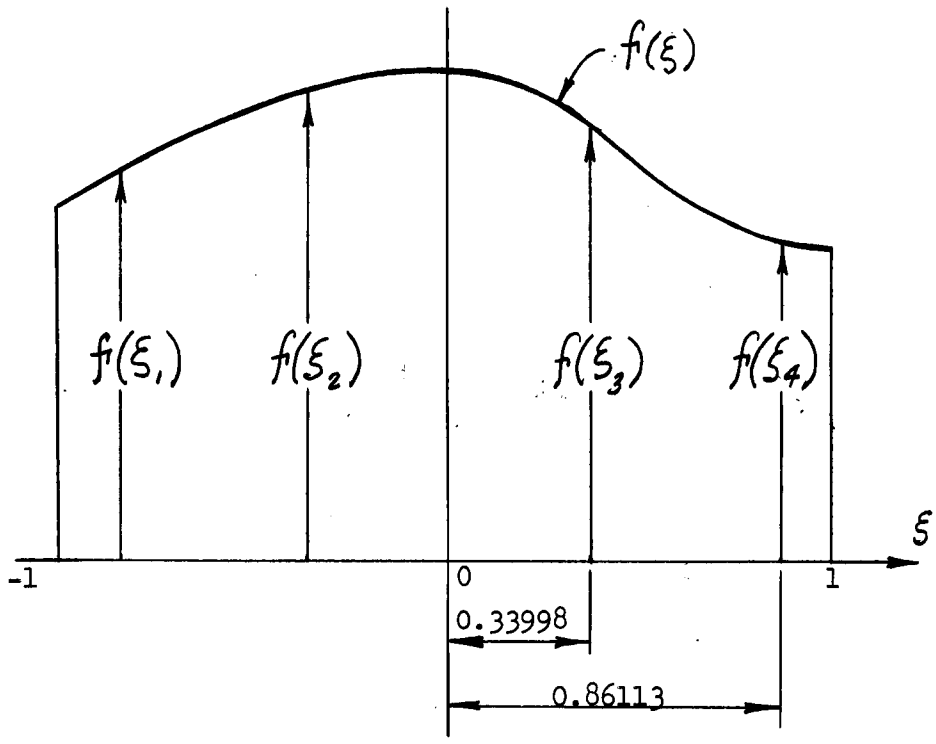


Figure 7. One-dimensional Gaussian integration.

$$I = \int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta)\, d\xi\, d\eta$$

is to first evaluate the inner integral keeping η constant, i.e.,

$$\int_{-1}^{1} f(\xi, \eta)\, d\xi \cong \sum_{j=1}^{n} H_i\, f(\xi_j, \eta) = \psi(\eta)$$

Evaluating the outer integral in a similar manner, we have

$$I = \int_{-1}^{1} \psi(\eta)\, d\eta = \sum_{i=1}^{n} H_i\, \psi(\eta_i)$$

$$= \sum_{i=1}^{n} H_i \sum_{j=1}^{n} H_j\, f(\xi_j, \eta_i)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} H_i H_j\, f(\xi_j, \eta_i)$$

For a right prism we have similarly

$$I \cong \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta, \zeta)\, d\xi\, d\eta\, d\zeta$$

$$= \sum_{m=1}^{n} \sum_{j=1}^{n} \sum_{i=1}^{n} H_i H_j H_m\, f(\xi_i, \eta_j, \zeta_m)$$

In the above the number of integrating points in each direction was assumed to be the same. Clearly this is not necessary and on occasion it may be of advantage to use different numbers in each direction of integration.

It is of interest to note that in fact the double summation can be readily interpreted as a single one over $(n \times n)$ points for a rectangle (or n^3 points for a cube). Thus in Figure 8 we show the nine sampling points which result in exact integrals of order five in each direction.

However, we could approach the problem directly and require an exact integration of fifth order polynomial in two directions. At any sampling point two coordinates and a value of f have to be determined in a weighting formula of type

$$I = \int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta)\, d\xi\, d\eta \cong \sum_{1}^{m} W_i\, f(\xi_i, \eta_i)$$

There it would appear that only seven points would suffice to obtain the same order of accuracy. Some such formulae for three-dimensional cubic elements have been derived [9] and used successfully.

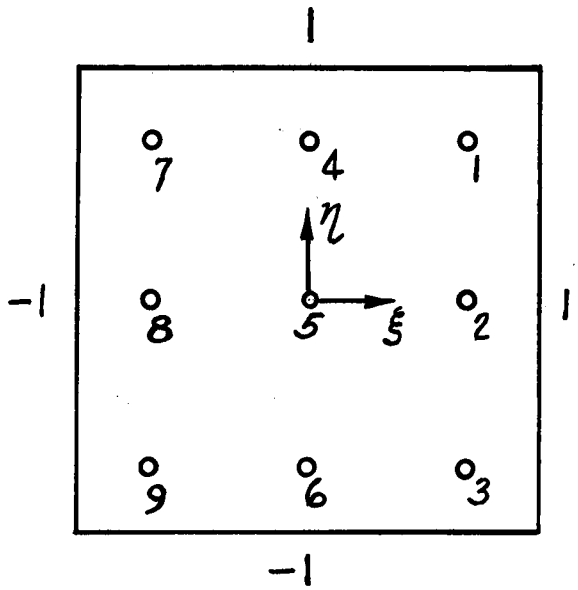


Figure 8. Integration points for n = 3 in a square region (exact for polynomial of fifth order in each direction).

TABLE A-1

ABSCISSAE AND WEIGHT COEFFICIENTS OF THE GAUSSIAN QUADRATURE FORMULA

$$\int_{-1}^{1} f(x)dx \cong \sum_{i=1}^{n} H_i \, f(a_i)$$

± a	H
n=2	
0.57735 02691 90626	1.00000 00000 00000
n=3	
0.77459 66692 41483	0.55555 55555 55556
0.00000 00000 00000	0.88888 88888 88889
n=4	
0.86113 63115 94053	0.34785 48451 37454
0.33998 10435 84856	0.65214 51548 62546

APPENDIX B

PROGRAM LISTING

This appendix contains listings for:

1. The assembly routine ASYMBL and the element routine ELSQ for the original, first modified, and second modified versions, and the final (optimized) version of ASYMBL for the given program using 200 elements.

2. The original and final versions of the element matrix integration testing programs utilizing 1001 elements.

3. SUBROUTINE ELCOM called by the second and final modified versions of ASYMBL to evaluate Gaussian integration data and local derivatives.

4. All routines originally called by ASYMBL and ELSQ and later replaced by in-line source code expansion.

```
      SUBROUTINE ASYMBL (M,N,NE,NG,NSPACE,NELFRE,MAXBAN,
     1          NDFREE,NITER,LPTEST,LHOMO,NNPFLO,IP1,IP3,IP4,
     2          IP5,JP2,KP1,KP2,NTAPE1,NODES,LNODE,INDEX,LPFIX
     3          ,LPROP,X,FLTNP,FLTEL,FLTMIS,ELPROP,COORD,S,C,
     4          SS,CC,D,DDOLD,PRTLPT,KP3)
C
C-*****************************************************************
C
C         ASSEMBLE ( IN CORE ) THE ELEMENT EQUATIONS
C
C-*****************************************************************
C
C                 GENERATE ELEMENT EQUATIONS
C                 AND POST SOLUTION MATRICES
C
C-*****************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(M,NSPACE),FLTNP(IP1,KP1),FLTEL(IP3,KP2),
     1          FLTMIS(KP3),ELPROP(KP2),COORD(N,NSPACE),S(NEL
     2          FRE,NELFRE),C(NELFRE),SS(NDFREE,NDFREE),CC(ND
     3          FREE),D(NELFRE),DDOLD(NDFREE),PRTLPT(IP5,KP1)
     4          ,NODES(NE,N),LNODE(N),INDEX(NELFRE),LPFIX(IP4
     5          ,JP2),LPROP(JP2)
C     M = NO. OF SYSTEM NODES, NG = NO. OF PARAMETERS PER
C         NODE
C     N = NUMBER OF NODES PER ELEMENT
C     NE = NUMBER OF ELEMENTS IN SYSTEM
C     NSPACE = DIMENSION OF SPACE
C     NELFRE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT
C     MAXBAN = MAX. HALF BANDWIDTH OF SYSTEM EQUATIONS
C     NDFREE = TOTAL NUMBER OF SYSTEM DEGREES OF FREEDOM
C     NITER = NO. OF ITERATIONS TO BE RUN (USUALLY NITER=1 )
C     IF LHOMO=1 ELEMENT PROPERTIES ARE HOMOGENEOUS
C     NNPFLO = NO. FLOATING POINT NODAL PROPERTIES
C     NTAPE1 = UNIT ON WHIC POST SOLUTION MATRICES ARE TO
C              BE STORED
C     NODES = ELEMENT INCIDENCES OF ALL ELEMENTS IN THE
C             SYSTEM
C     LNODE = THE N ELEMENT INCIDENCES OF THE ELEMENT
C     INDEX = SYSTEM D.O.F. NUMBERS ASSOCIATED WITH THE
C             ELEMENT
C     LPFIX = SYS. ARRAY OF FIXED PT. ELEM. PROP.
C     LPROP = ELEMENT ARRAY
C     X = COORDINATES OF SYSTEM NODES
C     FLTNP, FLTEL, FLTMIS = FLOATING POINT PROP. OF SYSTEM
C                            NODES,ELEMENTS,AND MISC.
C     COORD = COORDINATES OF ELEMENT NODES
```

```
C      SS = SYS. EQ. SQ. MATRIX, CC = SYS. EQ. COL. MATRIX
C      D = ARRAY OF ELEMENT DEGREES OF FREEDOM
C      PRTLPT= FLOATING POINT PROP. ARRAY OF ELEMENT NODES
C
       DO 10  IE = 1,NE
       CALL  LNODES (IE,NE,N,NODES,LNODE)
       CALL  ELCORD (IE,M,N,NSPACE,X,COORD,LNODE)
       IF ( NNPFLO.GT.0 )  CALL  LPTPRT (N,IP1,IP5,KP1,FLTNP,
      1PRTLPT,LNODE)
       CALL  INDXEL (N,NELFRE,NG,LNODE,INDEX)
       IF ( NITER.GT.1 )  CALL  ELFRE (NDFREE,NELFRE,D,DDOLD,
      1 INDEX)
       IF ( LPTEST.GT.0 )  CALL ELPRTY (IE,LHOMO,IP3,IP4,JP2,
      1KP2,LPFIX,FLTEL,LPROP,ELPROP)
C
       CALL ELSQ(N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
      1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,S)
C
       CALL  ELCOL  (N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
      1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,C)
       CALL  STORSQ (NDFREE,MAXBAN,NELFRE,INDEX,S,SS)
       CALL  STORCL (NDFREE,NELFRE,INDEX,C,CC)
       IF ( NTAPE1.GT.0 )  CALL  ELPOST (NTAPE1)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ELSQ (N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
     1                 COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,S)
C
C-*****************************************************************
C
C                    SQUARE ELEMENT MATRIX
C
C                A PROBLEM DEPENDENT ROUTINE
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION COORD(N,NSPACE),D(NELFRE),ELPROP(KP2),LPROP
     1          (JP2),S(NELFRE,NELFRE),PRTLPT(IP5,KP1),FLTMI
     2          S(KP3)
C
C-*****************************************************************
C
C                ISOPARAMETRIC QUADRILATERAL
C                     WITH FOUR NODES
C
C-APPLICATION: SOLUTION OF POISSON'S (OR LAPLACE'S) EQUATION
C              IN 2-D, NELFRE = 4, N =4, NSPACE = 2, NG = 1
C
C-*****************************************************************
C
C-VERSION 1
C
      DIMENSION AJ(2,2),DELTA(2,4),GLOBAL(2,4),DUM1(2),DUM2
     1          (2),SXX(4,4),SYY(4,4),GWT(4),GPT(4)
      DIMENSION  IDUM1(2), IDUM2(2)
      DIMENSION WT(16),ZETAPT(16),ETAPT(16)
C
C-DEFINE CONSTANT MATERIAL PROPERTIES
C
      XXK = ELPROP(1)
      YYK = ELPROP(2)
C
C-SET UP INTEGRATION DATA FOR NITG POINTS IN EACH DIMENSION
C
      NITG=3
C
      CALL  GAUSCO (NITG,GPT,GWT)
      NUMPTS = NITG*NITG
      K = 0
      DO 2  I = 1,NITG
      DO 1  J = 1,NITG
      K = K + 1
      WT(K) = GWT(I)*GWT(J)
      ZETAPT(K) = GPT(I)
      ETAPT(K) = GPT(J)
```

```
    1 CONTINUE
    2 CONTINUE
C
C-ZERO WORKING SPACE MATRICES
C
      LFRE2 = NELFRE*NELFRE
      CALL  ZERO ( LFRE2,1,S,IDUM)
C
C-NUMERICAL INTEGRATION LOOP
C
      DO 10 IP = 1,NUMPTS
C
C-FORM JACOBIAN MATRIX,  AJ
C
      CALL  DER4Q(ZETAPT(IP),ETAPT(IP),DELTA)
      CALL  MMULT(DELTA,COORD,AJ,NSPACE,N,NSPACE)
C
C-CALCULATE THE DETERMINANT OF AJ
C
      DET=AJ(1,1)*AJ(2,2)-AJ(1,2)*AJ(2,1)
C
C-CALCULATE INVERSE ( AJ REPLACED BY INVERSE OF AJ )
C
      X11=AJ(1,1)
      AJ(1,1)=AJ(2,2)/DET
      AJ(1,2)=-AJ(1,2)/DET
      AJ(2,1)=-AJ(2,1)/DET
      AJ(2,2)=X11/DET
C
C-ESTABLISH GLOBAL DERIVATIVES RELATIONS
C
      CALL  MMULT    (AJ,DELTA,GLOBAL,NSPACE,NSPACE,N)
C
C-SXX = XXK * DXT * DX * WT
C
      DO 35  I = 1,N
      DO 30  J = 1,N
   30 SXX(I,J) = DET*XXK*WT(IP)*GLOBAL(1,I)*GLOBAL(1,J)
   35 CONTINUE
C
C-S = S + SXX
C
      CALL  MADD(SXX,S,S,NELFRE,NELFRE)
C
C-SYY = YYK * DYT * DY * WT
C
      DO 45  I = 1,N
      DO 40 J = 1,N
   40 SYY(I,J) = DET*YYK*WT(IP)*GLOBAL(2,I)*GLOBAL(2,J)
```

```
   45 CONTINUE
C
C-S = S + SYY
C
      CALL  MADD(SYY,S,S,NELFRE,NELFRE)
   10 CONTINUE
C
C-INTEGRATION COMPLETED
C
      RETURN
      END
```

```
**********************************************************************

      SUBROUTINE ASYMBL (M,N,NE,NG,NSPACE,NELFRE,MAXBAN,
     1          NDFREE,NITER,LPTEST,LHOMO,NNPFLO,IP1,IP3,IP4,
     2          IP5,JP2,KP1,KP2,NTAPE1,NODES,LNODE,INDEX,LPFIX
     3          ,LPROP,X,FLTNP,FLTEL,FLTMIS,ELPROP,COORD,S,C,
     4          SS,CC,D,DDOLD,PRTLPT,KP3)
C
C-***************************************************************
C
C         ASSEMBLE ( IN CORE ) THE ELEMENT EQUATIONS
C
C-***************************************************************
C
C                GENERATE ELEMENT EQUATIONS
C                AND POST SOLUTION MATRICES
C
C-***************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(M,NSPACE),FLTNP(IP1,KP1),FLTEL(IP3,KP2),
     1          FLTMIS(KP3),ELPROP(KP2),COORD(N,NSPACE),S(NEL
     2          FRE,NELFRE),C(NELFRE),SS(NDFREE,NDFREE),CC(ND
     3          FREE),D(NELFRE),DDOLD(NDFREE),PRTLPT(IP5,KP1)
     4          ,NODES(NE,N),LNODE(N),INDEX(NELFRE),LPFIX(IP4
     5          ,JP2),LPROP(JP2)
C     M = NO. OF SYSTEM NODES, NG = NO. OF PARAMETERS PER
C         NODE
C     N = NUMBER OF NODES PER ELEMENT
C     NE = NUMBER OF ELEMENTS IN SYSTEM
C     NSPACE = DIMENSION OF SPACE
C     NELFRE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT
C     MAXBAN = MAX. HALF BANDWIDTH OF SYSTEM EQUATIONS
C     NDFREE = TOTAL NUMBER OF SYSTEM DEGREES OF FREEDOM
C     NITER = NO. OF ITERATIONS TO BE RUN (USUALLY NITER=1 )
C     IF LHOMO=1 ELEMENT PROPERTIES ARE HOMOGENEOUS
C     NNPFLO = NO. FLOATING POINT NODAL PROPERTIES
C     NTAPE1 = UNIT ON WHIC POST SOLUTION MATRICES ARE TO
C              BE STORED
C     NODES = ELEMENT INCIDENCES OF ALL ELEMENTS IN THE
C             SYSTEM
C     LNODE = THE N ELEMENT INCIDENCES OF THE ELEMENT
C     INDEX = SYSTEM D.O.F. NUMBERS ASSOCIATED WITH THE
C             ELEMENT
C     LPFIX = SYS. ARRAY OF FIXED PT. ELEM. PROP.
C     LPROP = ELEMENT ARRAY
C     X = COORDINATES OF SYSTEM NODES
C     FLTNP, FLTEL, FLTMIS = FLOATING POINT PROP. OF SYSTEM
C                            NODES,ELEMENTS,AND MISC.
```

```
C      COORD = COORDINATES OF ELEMENT NODES
C      SS = SYS. EQ. SQ. MATRIX, CC = SYS. EQ. COL. MATRIX
C      D = ARRAY OF ELEMENT DEGREES OF FREEDOM
C      PRTLPT= FLOATING POINT PROP. ARRAY OF ELEMENT NODES
       DIMENSION A(4),W(4),WT(16),XI(16),ETA(16)
C
C-GAUSSIAN QUADRATURE ABSCISSAE
C-AND WEIGHT COEFFICIENTS.
C
C-NUMBER OF GAUSS POINTS IN ONE DIMENSION
       NGP=3
       CALL GAUSCO(NGP,A,W)
       IP=0
       DO 1 I=1,NGP
       DO 1 J=1,NGP
       IP=IP+1
       WT(IP)=W(I)*W(J)
       XI(IP)=A(I)
       ETA(IP)=A(J)
     1 CONTINUE
C
       DO 10  IE = 1,NE
       CALL  LNODES (IE,NE,N,NODES,LNODE)
       CALL  ELCORD (IE,M,N,NSPACE,X,COORD,LNODE)
       IF ( NNPFLO.GT.0 )  CALL  LPTPRT (N,IP1,IP5,KP1,FLTNP,
      1PRTLPT,LNODE)
       CALL  INDXEL (N,NELFRE,NG,LNODE,INDEX)
       IF ( NITER.GT.1 )  CALL  ELFRE (NDFREE,NELFRE,D,DDOLD,
      1 INDEX)
       IF ( LPTEST.GT.0 )  CALL ELPRTY (IE,LHOMO,IP3,IP4,JP2,
      1KP2,LPFIX,FLTEL,LPROP,ELPROP)
C
       CALL ELSQ(N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
      1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,S,NGP,WT,
      2XI,ETA)
C
       CALL  ELCOL  (N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
      1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,C)
       CALL  STORSQ (NDFREE,MAXBAN,NELFRE,INDEX,S,SS)
       CALL  STORCL (NDFREE,NELFRE,INDEX,C,CC)
       IF ( NTAPE1.GT.0 )  CALL  ELPOST (NTAPE1)
    10 CONTINUE
       RETURN
       END
```

```
      SUBROUTINE ELSQ(N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
     1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,S,NITG,
     2WT,ZETAPT,ETAPT)
C
C-*************************************************************
C
C                    SQUARE ELEMENT MATRIX
C
C                 A PROBLEM DEPENDENT ROUTINE
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION COORD(N,NSPACE),D(NELFRE),ELPROP(KP2),LPROP
     1          (JP2),S(NELFRE,NELFRE),PRTLPT(IP5,KP1),FLTMI
     2          S(KP3)
C
C-*************************************************************
C
C                 ISOPARAMETRIC QUADRILATERAL
C                      WITH FOUR NODES
C
C-APPLICATION: SOLUTION OF POISSON'S (OR LAPLACE'S) EQUATION
C              IN 2-D, NELFRE = 4, N =4, NSPACE = 2, NG = 1
C
C-*************************************************************
C
C-VERSION 1
C
      DIMENSION AJ(2,2),DELTA(2,4),GLOBAL(2,4),DUM1(2),DUM2
     1          (2),SXX(4,4),SYY(4,4)
      DIMENSION  IDUM1(2), IDUM2(2)
      DIMENSION WT(16),ZETAPT(16),ETAPT(16)
C
C-DEFINE CONSTANT MATERIAL PROPERTIES
C
      XXK = ELPROP(1)
      YYK = ELPROP(2)
C
C-ZERO WORKING SPACE MATRICES
C
      LFRE2 = NELFRE*NELFRE
      CALL  ZERO ( LFRE2,1,S,IDUM)
C
C-NUMERICAL INTEGRATION LOOP
C
C
C-SET UP INTEGRATION DATA FOR NITG POINTS IN EACH DIMENSION
C
```

```
      NUMPTS = NITG*NITG
      DO 10 IP = 1,NUMPTS
C
C-FORM JACOBIAN MATRIX,  AJ
C
      CALL  DER4Q(ZETAPT(IP),ETAPT(IP),DELTA)
      CALL  MMULT(DELTA,COORD,AJ,NSPACE,N,NSPACE)
C
C-CALCULATE THE DETERMINANT OF AJ
C
      DET=AJ(1,1)*AJ(2,2)-AJ(1,2)*AJ(2,1)
C
C-CALCULATE INVERSE ( AJ REPLACED BY INVERSE OF AJ )
C
      X11=AJ(1,1)
      AJ(1,1)=AJ(2,2)/DET
      AJ(1,2)=-AJ(1,2)/DET
      AJ(2,1)=-AJ(2,1)/DET
      AJ(2,2)=X11/DET
C
C-ESTABLISH GLOBAL DERIVATIVES RELATIONS
C
      CALL  MMULT    (AJ,DELTA,GLOBAL,NSPACE,NSPACE,N)
C
C-SXX = XXK * DXT * DX * WT
C
      DO 35  I = 1,N
      DO 30  J = 1,N
   30 SXX(I,J) = DET*XXK*WT(IP)*GLOBAL(1,I)*GLOBAL(1,J)
   35 CONTINUE
C
C-S = S + SXX
C
      CALL  MADD(SXX,S,S,NELFRE,NELFRE)
C
C-SYY = YYK * DYT * DY * WT
C
      DO 45  I = 1,N
      DO 40 J = 1,N
   40 SYY(I,J) = DET*YYK*WT(IP)*GLOBAL(2,I)*GLOBAL(2,J)
   45 CONTINUE
C
C-S = S + SYY
C
      CALL  MADD(SYY,S,S,NELFRE,NELFRE)
   10 CONTINUE
C
C-INTEGRATION COMPLETED
C
```

```
RETURN
END
```

```
      SUBROUTINE ASYMBL (M,N,NE,NG,NSPACE,NELFRE,MAXBAN,
     1          NDFREE,NITER,LPTEST,LHOMO,NNPFLO,IP1,IP3,IP4,
     2          IP5,JP2,KP1,KP2,NTAPE1,NODES,LNODE,INDEX,LPFIX
     3          ,LPROP,X,FLTNP,FLTEL,FLTMIS,ELPROP,COORD,S,C,
     4          SS,CC,D,DDOLD,PRTLPT,KP3)
C
C-*****************************************************************
C
C         ASSEMBLE ( IN CORE ) THE ELEMENT EQUATIONS
C
C-*****************************************************************
C
C                 GENERATE ELEMENT EQUATIONS
C                 AND POST SOLUTION MATRICES
C
C-*****************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(M,NSPACE),FLTNP(IP1,KP1),FLTEL(IP3,KP2),
     1          FLTMIS(KP3),ELPROP(KP2),COORD(N,NSPACE),S(NEL
     2          FRE,NELFRE),C(NELFRE),SS(NDFREE,NDFREE),CC(ND
     3          FREE),D(NELFRE),DDOLD(NDFREE),PRTLPT(IP5,KP1)
     4          ,NODES(NE,N),LNODE(N),INDEX(NELFRE),LPFIX(IP4
     5          ,JP2),LPROP(JP2)
C     M = NO. OF SYSTEM NODES, NG = NO. OF PARAMETERS PER
C         NODE
C     N = NUMBER OF NODES PER ELEMENT
C     NE = NUMBER OF ELEMENTS IN SYSTEM
C     NSPACE = DIMENSION OF SPACE
C     NELFRE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT
C     MAXBAN = MAX. HALF BANDWIDTH OF SYSTEM EQUATIONS
C     NDFREE = TOTAL NUMBER OF SYSTEM DEGREES OF FREEDOM
C     NITER = NO. OF ITERATIONS TO BE RUN (USUALLY NITER=1 )
C     IF LHOMO=1 ELEMENT PROPERTIES ARE HOMOGENEOUS
C     NNPFLO = NO. FLOATING POINT NODAL PROPERTIES
C     NTAPE1 = UNIT ON WHIC POST SOLUTION MATRICES ARE TO
C              BE STORED
C     NODES = ELEMENT INCIDENCES OF ALL ELEMENTS IN THE
C             SYSTEM
C     LNODE = THE N ELEMENT INCIDENCES OF THE ELEMENT
C     INDEX = SYSTEM D.O.F. NUMBERS ASSOCIATED WITH THE
C             ELEMENT
C     LPFIX = SYS. ARRAY OF FIXED PT. ELEM. PROP.
C     LPROP = ELEMENT ARRAY
C     X = COORDINATES OF SYSTEM NODES
C     FLTNP, FLTEL, FLTMIS = FLOATING POINT PROP. OF SYSTEM
C                            NODES,ELEMENTS,AND MISC.
C     COORD = COORDINATES OF ELEMENT NODES
C     SS = SYS. EQ. SQ. MATRIX, CC = SYS. EQ. COL. MATRIX
```

```
C      D = ARRAY OF ELEMENT DEGREES OF FREEDOM
C      PRTLPT= FLOATING POINT PROP. ARRAY OF ELEMENT NODES
       DIMENSION WT(16)
       DIMENSION DELTA(2,4),DEL(2,4,16)
C
C-GAUSSIAN QUADRATURE ABSCISSAE
C-AND WEIGHT COEFFICIENTS.
C
C-NUMBER OF GAUSS POINTS IN ONE DIMENSION
       NGP=3
C
C-ELEMENT COMMON PROPERTIES
C
       CALL ELCOM(NGP,WT,DEL)
C
C-ELEMENT INDIVIDUAL PROPERTIES
C
C
       DO 10  IE = 1,NE
       CALL  LNODES (IE,NE,N,NODES,LNODE)
       CALL  ELCORD (IE,M,N,NSPACE,X,COORD,LNODE)
       IF ( NNPFLO.GT.0 )  CALL  LPTPRT (N,IP1,IP5,KP1,FLTNP,
      1PRTLPT,LNODE)
       CALL  INDXEL (N,NELFRE,NG,LNODE,INDEX)
       IF ( NITER.GT.1 )  CALL  ELFRE (NDFREE,NELFRE,D,DDOLD,
      1 INDEX)
       IF ( LPTEST.GT.0 )  CALL ELPRTY (IE,LHOMO,IP3,IP4,JP2,
      1KP2,LPFIX,FLTEL,LPROP,ELPROP)
C
       CALL ELSQ(N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
      1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,S,NGP,WT,
      2DEL)
C
       CALL  ELCOL  (N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
      1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,C)
       CALL  STORSQ (NDFREE,MAXBAN,NELFRE,INDEX,S,SS)
       CALL  STORCL (NDFREE,NELFRE,INDEX,C,CC)
       IF ( NTAPE1.GT.0 )  CALL  ELPOST (NTAPE1)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ELSQ(N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,
     1COORD,D,ELPROP,LPROP,PRTLPT,FLTMIS,S,NITG,
     2WT,DEL)
C
C-*****************************************************************
C
C                    SQUARE ELEMENT MATRIX
C
C                 A PROBLEM DEPENDENT ROUTINE
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION COORD(N,NSPACE),D(NELFRE),ELPROP(KP2),LPROP
     1          (JP2),S(NELFRE,NELFRE),PRTLPT(IP5,KP1),FLTMI
     2          S(KP3)
C
C-*****************************************************************
C
C                 ISOPARAMETRIC QUADRILATERAL
C                      WITH FOUR NODES
C
C-APPLICATION: SOLUTION OF POISSON'S (OR LAPLACE'S) EQUATION
C              IN 2-D, NELFRE = 4, N =4, NSPACE = 2, NG = 1
C
C-*****************************************************************
C
C-VERSION 1
C
      DIMENSION AJ(2,2),DELTA(2,4),GLOBAL(2,4),DUM1(2),DUM2
     1          (2),SXX(4,4),SYY(4,4)
      DIMENSION  IDUM1(2), IDUM2(2)
      DIMENSION WT(16),DEL(2,4,16)
C
C-DEFINE CONSTANT MATERIAL PROPERTIES
C
      XXK = ELPROP(1)
      YYK = ELPROP(2)
C
C-ZERO WORKING SPACE MATRICES
C
      LFRE2 = NELFRE*NELFRE
      CALL  ZERO ( LFRE2,1,S,IDUM)
C
C-NUMERICAL INTEGRATION LOOP
C
C
C-SET UP INTEGRATION DATA FOR NITG POINTS IN EACH DIMENSION
C
      NUMPTS = NITG*NITG
      DO 10 IP = 1,NUMPTS
```

```
C
C-FORM JACOBIAN MATRIX,  AJ
C
      DO 5 I=1,NSPACE
      DO 5 J=1,N
      DELTA(I,J)=DEL(I,J,IP)
    5 CONTINUE
      CALL  MMULT(DELTA,COORD,AJ,NSPACE,N,NSPACE)
C
C-CALCULATE THE DETERMINANT OF AJ
C
      DET=AJ(1,1)*AJ(2,2)-AJ(1,2)*AJ(2,1)
C
C-CALCULATE INVERSE ( AJ REPLACED BY INVERSE OF AJ )
C
      X11=AJ(1,1)
      AJ(1,1)=AJ(2,2)/DET
      AJ(1,2)=-AJ(1,2)/DET
      AJ(2,1)=-AJ(2,1)/DET
      AJ(2,2)=X11/DET
C
C-ESTABLISH GLOBAL DERIVATIVES RELATIONS
C
      CALL  MMULT    (AJ,DELTA,GLOBAL,NSPACE,NSPACE,N)
C
C-SXX = XXK * DXT * DX * WT
C
      DO 35  I = 1,N
      DO 30  J = 1,N
   30 SXX(I,J) = DET*XXK*WT(IP)*GLOBAL(1,I)*GLOBAL(1,J)
   35 CONTINUE
C
C-S = S + SXX
C
      CALL  MADD(SXX,S,S,NELFRE,NELFRE)
C
C-SYY = YYK * DYT * DY * WT
C
      DO 45  I = 1,N
      DO 40 J = 1,N
   40 SYY(I,J) = DET*YYK*WT(IP)*GLOBAL(2,I)*GLOBAL(2,J)
   45 CONTINUE
C
C-S = S + SYY
C
      CALL  MADD(SYY,S,S,NELFRE,NELFRE)
   10 CONTINUE
C
C-INTEGRATION COMPLETED
```

```
C
      RETURN
      END
```

```
      SUBROUTINE ASYMBL (M,N,NE,NG,NSPACE,NELFRE,MAXBAN,
     1          NDFREE,NITER,LPTEST,LHOMO,NNPFLC,IP1,IP3,IP4,
     2          IP5,JP2,KP1,KP2,NTAPE1,NODES,LNODE,INDEX,LPFIX
     3          ,LPROP,X,FLTNP,FLTEL,FLTMIS,ELPROP,COORD,S,C,
     4          SS,CC,D,DDOLD,PRTLPT,KP3)
C
C-*****************************************************************
C
C         ASSEMBLE ( IN CORE ) THE ELEMENT EQUATIONS
C
C-*****************************************************************
C
C                 GENERATE ELEMENT EQUATIONS
C                 AND POST SOLUTION MATRICES
C
C-*****************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(M,NSPACE),FLTNP(IP1,KP1),FLTEL(IP3,KP2),
     1          FLTMIS(KP3),ELPROP(KP2),COORD(N,NSPACE),S(NEL
     2          FRE,NELFRE),C(NELFRE),SS(NDFREE,NDFREE),CC(ND
     3          FREE),D(NELFRE),DDOLD(NDFREE),PRTLPT(IP5,KP1)
     4          ,NODES(NE,N),LNODE(N),INDEX(NELFRE),LPFIX(IP4
     5          ,JP2),LPROP(JP2)
C     M = NO. OF SYSTEM NODES, NG = NO. OF PARAMETERS PER
C         NODE
C     N = NUMBER OF NODES PER ELEMENT
C     NE = NUMBER OF ELEMENTS IN SYSTEM
C     NSPACE = DIMENSION OF SPACE
C     NELFRE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT
C     MAXBAN = MAX. HALF BANDWIDTH OF SYSTEM EQUATIONS
C     NDFREE = TOTAL NUMBER OF SYSTEM DEGREES OF FREEDOM
C     NITER = NO. OF ITERATIONS TO BE RUN (USUALLY NITER=1 )
C     IF LHOMO=1 ELEMENT PROPERTIES ARE HOMOGENEOUS
C     NNPFLC = NO. FLOATING POINT NODAL PROPERTIES
C     NTAPE1 = UNIT ON WHIC POST SOLUTION MATRICES ARE TO
C              BE STORED
C     NODES = ELEMENT INCIDENCES OF ALL ELEMENTS IN THE
C             SYSTEM
C     LNODE = THE N ELEMENT INCIDENCES OF THE ELEMENT
C     INDEX = SYSTEM D.O.F. NUMBERS ASSOCIATED WITH THE
C             ELEMENT
C     LPFIX = SYS. ARRAY OF FIXED PT. ELEM. PROP.
C     LPROP = ELEMENT ARRAY
C     X = COORDINATES OF SYSTEM NODES
C     FLTNP, FLTEL, FLTMIS = FLOATING POINT PROP. OF SYSTEM
C                            NODES,ELEMENTS,AND MISC.
C     COORD = COORDINATES OF ELEMENT NODES
C     SS = SYS. EQ. SQ. MATRIX, CC = SYS. EQ. COL. MATRIX
```

```
C      D = ARRAY OF ELEMENT DEGREES OF FREEDOM
C      PRTLPT= FLOATING POINT PROP. ARRAY OF ELEMENT NODES
C
C-ELEMENT COMMON PROPERTIES
C
       DIMENSION WT(16),DEL(2,4,16)
       DIMENSION AJ(2,2),DE(2,4)
       DIMENSION AJJ(4)
       EQUIVALENCE(AJ(1,1),AJJ(1))
C      NGP-NUMBER OF GAUSSIAN INTEGRATION POINTS IN ONE DIM.
C      WT-MULTIPLE OF GAUSSIAN WEIGHTS. (W(I)*W(J))
C      DEL-ARRAY OF LOCAL DERIVATIVES OF INTERPOLATION
C          FUNCTION FOR EACH INTEGRATION POINT.
C
C-NUMBER OF GAUSSIAN INTEGRATION POINTS IN ONE DIMENSION
C
       NGP=3
       CALL ELCOM(NGP,WT,DEL)
C
C-ELEMENT INDIVIDUAL PROPERTIES
C
       DO 100 IE=1,NE
C
C-NODES ASSOCIATED WITH ELEMENT NUMBER IE AND
C-CO-ORDINATES OF NODAL POINTS ON ELEMENT NUMBER IE.
C
       DO 5 I=1,N
       LNODE(I)=NODES(IE,I)
       DO 4 K=1,NSPACE
       COORD(I,K)=X(LNODE(I),K)
     4 CONTINUE
     5 CONTINUE
       IF(NNPFLG.GT.0)CALL LPTPRT(N,IP1,IP5,KP1,FLTNP,PRTLPT,
      1LNODE)
C
C-DETERMINE DEGREES OF FREEDOM ASSOCIATED WITH ELEMENT.
C
       DO 7 K=1,N
       NGKM1=NG*(K-1)
       NGLKM1=NG*(LNODE(K)-1)
       DO 6 IG=1,NG
       INDEX(IG+NGKM1)=NGLKM1+IG
     6 CONTINUE
     7 CONTINUE
       IF(NITER.LE.1)GO TO 15
C
C-EXTRACT ELEMENT DEGREES OF FREEDOM FROM SYSTEM DEGREES OF
C-FREEDOM
C
```

```
      NLESS=N-1
      DO 12 I=1,NELFRE
      D(I)=DDOLD(INDEX(I))
   12 CONTINUE
   15 CONTINUE
      IF(LPTEST.LE.0)GO TO 19
C
C-EXTRACT PROPERTIES OF A SINGLE ELEMENT,IE,FROM TOTAL
C-PROPERTIES ARRAYS
C
      I=IE
      IF(LHOMO.EQ.1)I=1
      DO 17 J=1,KP2
      ELPROP(J)=FLTEL(I,J)
   17 CONTINUE
      DO 18 J=1,JP2
      LPROP(J)=LPFIX(I,J)
   18 CONTINUE
   19 CONTINUE
C
C-DEFINE CONSTANT MATERIAL PROPERTIES
C
      XXK=ELPROP(1)
      YYK=ELPROP(2)
C
C-INITIALIZE (ZERO) ELEMENT MATRIX (S)
C
      DO 34 I=1,NELFRE
      DO 33 J=1,NELFRE
      S(I,J)=0.0
   33 CONTINUE
   34 CONTINUE
C
C-NUMERICAL(GAUSSIAN QUADRATURE)INTEGRATION LOOP
C
C-NUMBER OF INTEGRATION POINTS PER ELEMENT
C
      NIP=NGP*NGP
      DO 75 IP=1,NIP
C
C-JACOBIAN(AJ)
C
      DO 30 I=1,NSPACE
      DO 25 J=1,NSPACE
      SUM=0.0
      DO 20 K=1,N
      SUM=DEL(I,K,IP)*CCORD(K,J)+SUM
```

```
   20 CONTINUE
      AJ(I,J)=SUM
   25 CONTINUE
   30 CONTINUE
C
C-DETERMINANT OF JACOBIAN MATRIX(DET)
C
      DET=((AJJ(1)*AJJ(4))-(AJJ(3)*AJJ(2)))
      DETI=1.0/DET
C
C-INVERSE OF JACOBIAN(AJ REPLACED BY INVERSE OF AJ)
C
      X11=AJJ(1)
      AJJ(1)=AJJ(4)*DETI
      AJJ(3)=-AJJ(3)*DETI
      AJJ(2)=-AJJ(2)*DETI
      AJJ(4)=X11*DETI
C
C-GLOBAL DERIVATIVES OF INTERPOLATION FUNCTIONS(DE)
C
      DO 60 I=1,NSPACE
      DO 50 J=1,N
      SUM=0.0
      DO 40 K=1,NSPACE
      SUM=AJ(I,K)*DEL(K,J,IP)+SUM
   40 CONTINUE
      DE(I,J)=SUM
   50 CONTINUE
   60 CONTINUE
C
C-ELEMENT SQUARE MATRIX (S)
C
      DETWT=DET*WT(IP)
      DO 70 I=1,N
      DO 65 J=1,N
      S(I,J)=DETWT*(XXK*DE(1,I)*DE(1,J)+YYK*DE(2,I)*DE(2,J))
     1 +S(I,J)
   65 CONTINUE
   70 CONTINUE
   75 CONTINUE
C
C-INTEGRATION COMPLETED
C
C-UPPER HALF BAND OF SYSTEM MATRIX FOR A SYMMETRIC MATRIX
C-AND ELEMENT COLUMN MATRIX STORAGE IN SYSTEM COLUMN MATRIX.
C
      DO 95 L=1,NELFRE
      I=INDEX(L)
      CC(I)=C(L)+CC(I)
```

```
      DO 90 K=1,NELFRE
      J=INDEX(K)
      IF(I.GT.J) GO TO 90
      JJ=J-I+1
      SS(I,JJ)=SS(I,JJ)+S(L,K)
   90 CONTINUE
   95 CONTINUE
      IF(NTAPE1.GT.0)CALL ELPOST(NTAPE1)
  100 CONTINUE
      RETURN
      END
```

```
C
C-*****************************************************************
C
C                        ORIGINAL VERSION
C
C                INTEGRATION OF ELEMENT MATRICES
C
C                          1001  ELEMENTS
C
C                     3 PT. GAUSSIAN QUADRATURE
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION COORD(4,2),D(4),ELPROP(2),
     1LPROP(1),S(4,4),PRTLPT(1,1),FLTMIS(1)
C
C-CALLING ARGUMENTS
C
      N=4
      NELFRE=4
      NSPACE=2
      IP5=1
      JP2=1
      KP1=1
      KP2=2
      KP3=1
      COORD(1,1)=1.
      COORD(2,1)=3.
      COORD(3,1)=2.
      COORD(4,1)=0.0
      COORD(1,2)=0.0
      COORD(2,2)=0.0
      COORD(3,2)=1.
      COORD(4,2)=1.
      ELPROP(1)=1.
      ELPROP(2)=1.
C
C -NUMBER OF ELEMENTS (IDENTICAL CO-ORDINATES)
C
      NE=1001
C
      DO 950 IE=1,NE
C
      CALL ELSQ(N,NELFRE,NSPACE,IP5,JP2,KP1,KP2,KP3,COORD,
     1          D,ELPROP,LPROP,PRTLPT,FLTMIS,S)
```

```
C
C-PRINTS ELEMENT SYSTEM ARRAY
C
      IF(IE.GT.1.AND.IE.LT.1001) GO TO 950
      WRITE(6,100)
  100 FORMAT('1',10X,'ELEMENT SYSTEM ARRAY')
      DO 250 I=1,4
      WRITE(6,200)(S(I,J),J=1,4)
  200 FORMAT('0',5X,4(5X,E15.8))
  250 CONTINUE
  950 CONTINUE
      WRITE(6,975) IE
  975 FORMAT('0','INTEGRATION COMPLETED       IE=',I6)
      CALL EXIT
      END
```

```
C
C-*************************************************************
C
C                        FINAL VERSION
C
C              INTEGRATION OF ELEMENT MATRICES
C
C                        1001  ELEMENTS
C
C                   3 PT. GAUSSIAN QUADRATURE
C
C-*************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION S(4,4),ELPROP(2),COORD(4,2)
C - S = ELEMENT SQUARE MATRIX
C - COORD = MATRIX OF COORDINATES OF ELEMENT NODES
C - ELPROP = MATRIX OF MATERIAL PROPERTIES
      N=4
      NSPACE=2
      COORD(1,1)=1.
      COORD(2,1)=3.
      COORD(3,1)=2.
      COORD(4,1)=0.0
      COORD(1,2)=0.0
      COORD(2,2)=0.0
      COORD(3,2)=1.
      COORD(4,2)=1.
      ELPROP(1)=1.
      ELPROP(2)=1.
C
C - NUMBER OF ELEMENTS(IDENTICAL CO-ORDINATES)
C
      NE=1001
C
C-ELEMENT COMMON PROPERTIES
C
      DIMENSION WT(16),DEL(2,4,16)
      DIMENSION AJ(2,2),DE(2,4)
      DIMENSION AJJ(4)
      EQUIVALENCE(AJJ(1),AJ(1,1))
C     NGP-NUMBER OF GAUSSIAN INTEGRATION POINTS IN ONE DIM.
C     WT-MULTIPLE OF GAUSSIAN WEIGHTS. (W(I)*W(J))
C     DEL-ARRAY OF LOCAL DERIVATIVES OF INTERPOLATION
C         FUNCTION FOR EACH INTEGRATION POINT.
C
C-NUMBER OF GAUSSIAN INTEGRATION POINTS IN ONE DIMENSION
C
      NGP=3
```

```
      CALL ELCON(NGP,WT,DEL)
C
C-ELEMENT INDIVIDUAL PROPERTIES
C
      DO 950 IE=1,NE
C
C-DEFINE CONSTANT MATERIAL PROPERTIES
C
      XXK=ELPROP(1)
      YYK=ELPROP(2)
C
C-INITIALIZE (ZERO) ELEMENT MATRIX (S)
C
      DO 34 I=1,N
      DO 33 J=1,N
      S(I,J)=0.0
   33 CONTINUE
   34 CONTINUE
C
C-NUMERICAL(GAUSSIAN QUADRATURE)INTEGRATION LOOP
C
C-NUMBER OF INTEGRATION POINTS PER ELEMENT
C
      NIP=NGP*NGP
      DO 75 IP=1,NIP
C
C-JACOBIAN(AJ)
      DO 35 I=1,NSPACE
      DO 30 J=1,NSPACE
      SUM=0.0
      DO 25 K=1,N
      SUM=SUM+DEL(I,K,IP)*COORD(K,J)
   25 CONTINUE
      AJ(I,J)=SUM
   30 CONTINUE
   35 CONTINUE
C
C-DETERMINANT OF JACOBIAN MATRIX(DET)
C
      DET=((AJJ(1)*AJJ(4))-(AJJ(3)*AJJ(2)))
      DETI=1.0/DET
C
C-INVERSE OF JACOBIAN(AJ REPLACED BY INVERSE OF AJ)
C
      X11=AJJ(1)
      AJJ(1)=AJJ(4)*DETI
      AJJ(3)=-AJJ(3)*DETI
      AJJ(2)=-AJJ(2)*DETI
      AJJ(4)=X11*DETI
```

```
C
C-GLOBAL DERIVATIVES OF INTERPOLATION FUNCTIONS(DE)
C
      DO 60 I=1,NSPACE
      DO 50 J=1,N
      SUM=0.0
      DO 40 K=1,NSPACE
      SUM=SUM+AJ(I,K)*DEL(K,J,IP)
   40 CONTINUE
      DE(I,J)=SUM
   50 CONTINUE
   60 CONTINUE
C
C-ELEMENT SQUARE MATRIX (S)
C
      DETWT=DET*WT(IP)
      DO 70 I=1,N
      DO 65 J=1,N
      S(I,J)=S(I,J)+DETWT*(XXK*DE(1,I)*DE(1,J)+YYK*DE(2,I)
     1 *DE(2,J))
   65 CONTINUE
   70 CONTINUE
   75 CONTINUE
C
C-INTEGRATION COMPLETED
C
C-PRINTS ELEMENT SQUARE MATRIX
C
      IF(IE.GT.1.AND.IE.LT.1001) GO TO 950
      WRITE(6,100)
  100 FORMAT('1',10X,'ELEMENT SQUARE MATRIX')
      DO 250 I=1,4
      WRITE(6,200)(S(I,J),J=1,4)
  200 FORMAT('0',5X,4(5X,E15.8))
  250 CONTINUE
  950 CONTINUE
      WRITE(6,975)IE
  975 FORMAT('0','INTEGRATION COMPLETED     IE=',I6)
      CALL EXIT
      END
```

```
      SUBROUTINE ELCOM(NGP,WT,DEL)
C
C-***************************************************************
C-PURPOSE
C     ESTABLISHES GAUSSIAN WEIGHT PRODUCTS AND
C     ARRAY OF LOCAL DERIVATIVES OF GAUSSIAN
C     INTEGRATION POINTS COMMON TO ALL
C     2-D ISOPARAMETIC QUADRILATERIAL ELEMENTS
C     WITH FOUR NODES AND HAVING THE FOLLOWING
C     INTERPOLATION (SHAPE) FUNCTION.
C
C     N(XI,ETA,I)=0.25(1.+XI*XI(I))(1.+ETA*ETA(I))
C     WHERE XI AND ETA ARE LOCAL COORDINATES FOR ANY
C     ELEMENT POINT AND XI(I) AND ETA(I) ARE
C     ELEMENT NODAL LOCAL COORDINATES.
C
C-DESCRIPTION OF PARAMETERS
C     NGP-NO. OF GAUSSIAN INTEGRATION POINTS IN ONE DIM.
C     WT- ARRAY OF GAUSSIAN INTEGRATION POINT WEIGHT
C         PRODUCTS ( W(I) * W(J) )
C     A - ABSCISSAE OF GAUSSIAN INTEGRATION POINTS.
C     DEL-ARRAY OF LOCAL DERIVATIVES OF INTERPOLATION
C         FUNCTION FOR EACH INTEGRATION POINT.
C     W - WEIGHTS OF GAUSSIAN INTEGRATION POINTS IN ONE DIM.
C     XI-ARRAY OF LOCAL 'X' COORDINATES.
C     ETA-ARRAY OF LOCAL 'Y' COORDINATES.
C
C-***************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(4),W(4),WT(16),XI(16),ETA(16),
     1          DELTA(2,4),DEL(2,4,16)
C
C-GAUSSIAN QUADRATURE ABSCISSAE AND WEIGHT COEFFICIENTS
C
      CALL GAUSCO(NGP,A,W)
      IP=0
      DO 1 I=1,NGP
      DO 1 J=1,NGP
      IP=IP+1
      WT(IP)=W(I)*W(J)
      XI(IP)=A(I)
      ETA(IP)=A(J)
    1 CONTINUE
C
C-NUMBER OF ELEMENT INTEGRATION POINTS
C
      NIP=IP
C
```

```
C-LOCAL DERIVATIVES OF INTERPOLATION
C-FUNCTION AT EACH INTEGRATION POINT.(DEL)
C
      DO 2 IP=1,NIP
      CALL DER4Q(XI(IP),ETA(IP),DELTA)
      DO 2 J=1,2
      DO 2 M=1,4
      DEL(J,M,IP)=DELTA(J,M)
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE  DER4Q (ZETA,ETA,DELTA)
C
C-*****************************************************************
C
C     LOCAL DERIVATIVES OF THE INTERPOLATION (SHAPE)
C     FUNCTIONS FOR AN ISOPARAMETRIC QUADRILATERAL
C     WITH FOUR NODES
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION  DELTA(2,4)
C     DH(I)/DZETA = 0.25*ZETA(I)*(1.+ETA*ETA(I))
C          ZETA AND ETA ARE THE LOCAL COORDINATES
      EP = 1. + ETA
      EM = 1. - ETA
      ZP = 1. + ZETA
      ZM = 1. - ZETA
      DELTA(1,1) = -0.25*EM
      DELTA(1,2) =  0.25*EM
      DELTA(1,3) =  0.25*EP
      DELTA(1,4) = -0.25*EP
      DELTA(2,1) = -0.25*ZM
      DELTA(2,2) = -0.25*ZP
      DELTA(2,3) =  0.25*ZP
      DELTA(2,4) =  0.25*ZM
      RETURN
      END
```

```
      SUBROUTINE GAUSCO (N,A,W)
C
C-*****************************************************************
C
C           TABLE OF GAUSSIAN QUADRATURE ABSCISSAE
C
C                AND WEIGHT COEFFICIENTS
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(N), W(N)
C     N = NO. OF GAUSS POINTS IN ONE DIMENSION
C     A = ABSCISSAE, W = WEIGHTS      OF GAUSS POINTS
C     NMAX = MAX. NO. OF POINTS TABULATED HEREIN
      NMAX = 4
      NLES1 = N - 1
      GO TO (10,20,30,40), NLES1
 10   CONTINUE
C
C N=2
C
      A(1) = 0.577350269189626D+00
      A(2) = -A(1)
      W(1) = 1.D+00
      W(2) = 1.D+00
      RETURN
 20   CONTINUE
C
C N=3
C
      A(1) = 0.774596669241483D+00
      A(2) = 0.D+00
      A(3) = -A(1)
      W(1) = 0.555555555555556D+00
      W(2) = 0.888888888888889D+00
      W(3) = W(1)
      RETURN
 30   CONTINUE
C
C N=4
C
      A(1) = 0.861136311594053D+00
      A(2) = 0.339981043584856D+00
      A(3) = -A(2)
      A(4) = -A(1)
      W(1) = 0.347854845137454D+00
      W(2) = 0.652145154862546D+00
      W(3) = W(2)
```

```
      W(4) = W(1)
      RETURN
40    CONTINUE
      WRITE (6,1000) NMAX
1000 FORMAT ('ON EXCEEDS ',I3,'. COEFFICIENTS NOT
     1TABULATED.')
      RETURN
      END
```

```
      SUBROUTINE MADD (A,B,C,M,N)
C
C-*****************************************************************
C
C                        MATRIX ADDITION
C
C                          A + B = C
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION  A(M,N), B(M,N), C(M,N)
      DO 2  I = 1,M
      DO 1  J = 1,N
    1 C(I,J) = A(I,J) + B(I,J)
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE  MMULT (A,B,C,L,M,N)
C
C-*****************************************************************
C
C                   MATRIX MULTIPLICATION
C
C                A(L,M) X B(M,N) = C(L,N)
C
C-*****************************************************************
C
C                      FAST VERSION
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(L,M), B(M,N), C(L,N)
      DO 50  I = 1,L
      DO 40  J = 1,N
      SUM = 0.0
      DO 30  K = 1,M
      BKJ = B(K,J)
      IF ( BKJ)  10,30,10
 10   AIK = A(I,K)
      IF ( AIK )  20,30,20
 20   SUM = SUM + AIK*BKJ
 30   CONTINUE
      C(I,J) = SUM
 40   CONTINUE
 50   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE  ZERO (N1,N2,A,I)
C
C-*****************************************************************
C
                INITIALIZE (ZERO) STORAGE VECTORS
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(N1), I(N2)
      DO 10  J = 1,N1
 10   A(J) = 0.0
      DO 20  J = 1,N2
 20   I(J) = 0
      RETURN
      END
```

```
      SUBROUTINE ELCORD (LID,M,N,NSPACE,X,COORD,LNODE)
C
C-****************************************************************
C
C
C              DETERMINE COORDINATES OF NODAL
C
C               POINTS ON ELEMENT NUMBER LID
C
C
C-****************************************************************
C
C-PARAMETERS
C
C     LID = ELEMENT NUMBER
C     M = NO. OF NODES IN SYSTEM
C     N = NUMBER OF NODES PER ELEMENT
C     NSPACE = DIMENSION OF SPACE
C     X = COORDINATES OF SYSTEM NODES
C     COORD = COORDINATES OF ELEMENT NODES
C     LNODE = ELEMENT INCIDENCES ARRAY OF THE ELEMENT
C
C-****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(M,NSPACE),COORD(N,NSPACE),LNODE(N)
      DO 10 I = 1,N
      DO 10 K = 1,NSPACE
      COORD(I,K) = X(LNODE(I),K)
      RETURN
      END
```

```
      SUBROUTINE ELFRE (NDFREE,NELFRE,D,DD,INDEX)
C
C-*****************************************************************
C
C
C             EXTRACT ELEMENT DEGREES OF FREEDOM
C
C               FROM SYSTEM DEGREES OF FREEDOM
C
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION D(NELFRE), DD(NDFREE), INDEX(NELFRE)
      NLESS = N - 1
      DO 10 I = 1,NELFRE
   10 D(I) = DD(INDEX(I))
      RETURN
      END
```

```
      SUBROUTINE ELPRTY (LID,LHOMO,IP3,IP4,JP2,KP2,LPFIX,
     1                   FLTEL,LPROP,ELPROP)
C
C-****************************************************************
C-PURPOSE
C
C           EXTRACT PROPERTIES OF A SINGLE ELEMENT,
C
C              LID, FROM TOTAL PROPERTIES ARRAYS
C
C-****************************************************************
C
C-DESCRIPTION OF PARAMETERS
C
C     LPFIX = SYS. ARRAY OF FIXED PT. ELEMENT PROPERTIES
C     LPROP = ELEMENT ARRAY
C     FLTEL = SYS. ARRAY OF FLOATING PT. NODAL PROPERTIES
C     ELPROP = ELEMENT ARRAY
C
C-REMARKS
C
C  (1) IF LHOMO = 1 THEN ELEMENT PROPERTIES ARE HOMOGENEOUS
C      THROUGHOUT THE SYSTEM
C
C-****************************************************************
C
      DIMENSION FLTEL(IP3,KP2),ELPROP(KP2),LPFIX(IP4,JP2),
     1          LPROP(JP2)
      I=LID
      IF ( LHOMO.EQ.1 ) I=1
      DO 1 J = 1,KP2
    1 ELPROP(J) = FLTEL(I,J)
      DO 2 J=  1,JP2
    2 LPROP(J) = LPFIX(I,J)
      RETURN
      END
```

```
      SUBROUTINE INDXEL (N,LEMFRE,NG,LNODE,INDEX)
C
C-*****************************************************************
C
C                 DETERMINE DEGREES OF FREEDOM
C
C                   ASSOCIATED WITH ELEMENT
C
C-*****************************************************************
C
C-PARAMETERS
C
C     N = NO. OF NODES PER ELEMENT
C     NG = NO. OF PARAMETERS PER NODE
C     LEMFRE = N*NG
C     LNODE = THE N ELEMENT INCIDENCES OF THE ELEMENT
C     INDEX = SYS. DEGREES OF FREEDOM CORRESPONDING TO
C             (N*NG) ELEMENT PARAMETERS
C
C-*****************************************************************
C
      DIMENSION INDEX(LEMFRE),LNODE(N)
      DO 10 K = 1,N
      DO 10 IG = 1,NG
      IELM = NG*(K-1) + IG
   10 INDEX(IELM) = NG*(LNODE(K)-1) + IG
      RETURN
      END
```

```
      SUBROUTINE LNODES (LID,NE,N,NODES,LNODE)
C
C-*****************************************************************
C
C
C                  DETERMINE NODES ASSOCIATED
C
C                   WITH ELEMENT NUMBER LID
C
C-*****************************************************************
C
C-PARAMETERS
C
C      NE = NUMBER OF ELEMENTS IN SYSTEM
C      N = NUMBER OF NODES PER ELEM
C      LID = ELEMENT NUMBER
C      NODES = ELEMENT INCIDENCES OF ALL ELEMENTS IN THE SYS.
C      LNODE = ELEMENT INCIDENCES OF ELEMENT LID
C
C-*****************************************************************
C
      DIMENSION NODES(NE,N), LNODE(N)
      DO 10 I = 1,N
   10 LNODE(I) = NODES(LID,I)
      RETURN
      END
```

```
      SUBROUTINE STORCL (NDFREE,NDF,INDEX,C,CC)
C
C-*****************************************************************
C
C                     STORE ELEMENT COLUMN
C
C                 MATRIX IN SYSTEM COLUMN MATRIX
C
C-*****************************************************************
C
C-PARAMETERS
C
C     INDEX = ARRAY CONTAINING SYSTEM INDICES OF NODAL
C             DEGREES OF FREEDOM OF THE ELEMENT TO BE
C             STORED
C     C = ELEMENT COLUMN MATRIX
C     CC = SYSTEM COLUMN MATRIX
C     NDFREE = DEGREES OF FREEDOM OF THE SYSTEM
C     NDF = DEGREES OF FREEDOM IN ONE ELEMENT
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION C(NDF),CC(NDFREE),INDEX(NDF)
      DO 10 I = 1,NDF
      J = INDEX(I)
   10 CC(J) = C(I) + CC(J)
      RETURN
      END
```

```
      SUBROUTINE STORSQ (NDFREE,IBW,NDF,INDEX,S,SS)
C
C-*****************************************************************
C
C              THIS SUBROUTINE BUILDS THE UPPER
C
C               HALF BAND OF THE SYSTEM MATRIX
C
C-*****************************************************************
C
C-PARAMETERS
C
C      INDEX - SYSTEM INDICES OF NODAL DEGREES OF FREEDOM
C              OF THE ELEMENT
C      S - SQUARE ELEMENT MATRIX
C      SS - SQUARE SYSTEM MATRIX
C      NDFREE - DEGREES OF FREEDOM IN SYSTEM
C      IBW - HALF BAND WIDTH INCLUDING THE DIAGONAL
C      NDF - NUMBER OF DEGREES OF FREEDOM IN ELEMENT
C      I,J - ROW AND COLUMN POSITIONS IN THE UNPACKED SYSTEM
C            MATRIX
C      JJ - COLUMN POSITION IN THE PACKED SYSTEM MATRIX
C
C-*****************************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION S(NDF,NDF), SS(NDFREE,IBW), INDEX(NGF)
C
      DO 10 L = 1,NDF
      I = INDEX(L)
      DO 10 K = 1,NDF
      J = INDEX(K)
      IF(I.GT.J) GO TO 10
      JJ = J-I+1
      SS(I,JJ) = SS(I,JJ) + S(L,K)
   10 CONTINUE
      RETURN
      END
```

VITA

William Wear Murphy was born in Knoxville, Tennessee, on December 29, 1936. He attended elementary schools in Clinton, Tennessee and was graduated from Clinton High School in 1954. The following September he entered The University of Tennessee, and in June 1959 he received a Bachelor of Science degree in Mechanical Engineering. Following graduation he accepted employment with Thiokol Chemical Corporation, Huntsville, Alabama.

He returned to Knoxville in 1961 to accept employment with the Tennessee Valley Authority where he is presently employed.

He enrolled as an unclassified graduate student at The University of Tennessee in March 1961, and later applied and was accepted to the Graduate School.

Hobbies include painting, hiking, camping, hunting, and skiing.