# COMPARISON OF ENERGY MINIMIZATION WITH DIRECT STIFFNESS FOR LINEAR STRUCTURAL ANALYSIS

by

David Thomas Griffith

Thesis submitted to the Graduate Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Civil Engineering

APPROVED:

A. E. Somers, Jr., Chairman

S. M. Holzer

R. H. Plaut

May, 1979

Blacksburg, Virginia

## ACKNOWLEDGMENTS

The author wishes to express his deepest appreciation for the continued encouragement and suggestions of Dr. A. E. Somers during the writing of this thesis. Thanks are given to Dr. S. M. Holzer and Dr. R. H. Plaut for reviewing this thesis.

Special thanks is given to for her efficient typing of this thesis.

TABLE OF CONTENTS


| | Page |
|---|---|
| Acknowledgements | ii |
| List of Figures | v |
| List of Tables | vii |
| Notation | viii |
| 1. Introduction | 1 |
| 1.1 Purpose and Scope | 1 |
| 1.2 Literature Review | 2 |
| a. Variable Metric Algorithms | 2 |
| b. Conjugate Gradient Algorithms | 8 |
| c. Scaling Applications | 10 |
| 1.3 Algorithm Selection | 13 |
| 2. Problem Formulation | 18 |
| 2.1 Energy Method | 18 |
| a. Total Potential Energy | 18 |
| b. Principle of Minimum Potential Energy | 22 |
| c. Total Potential of a Structural System and its Gradient Vector | 24 |
| 2.2 Stiffness Method | 29 |
| 3. Solution Process | 33 |
| 3.1 Structural Interpretations of Mathematical Relations used by the Minimization Algorithms | 33 |
| 3.2 Variable Metric Algorithm | 41 |
| 3.3 Conjugate Gradient Algorithm | 47 |
| 3.4 Gaussian Elimination | 50 |
| 4. Results | 53 |
| 4.1 Example Problems | 53 |
| 4.2 Results from a Single Load Condition | 61 |
| a. Results from the Energy Method | 61 |
| b. Results from the Stiffness Method | 65 |
| 4.3 Results of Multiple Load Conditions | 67 |

Page

5. Summary ..... 77

5.1 Conclusions ..... 77
5.2 Recommendations ..... 79

Appendix I - References ..... 81

Appendix II - User's Guide for ENERGY ..... 86

Appendix III - User's Guide for STIFF ..... 90

Appendix IV - ENERGY Computer Code Listing ..... 94

Appendix V - STIFF Computer Code Listing .....113

Vita .....126

Abstract

LIST OF FIGURES


| Figure | | Page |
|---|---|---|
| 1.1 | Optimally Scaled 2 Variable Quadratic Function ..... | 11 |
| 2.1 | Strain Energy Density ..... | 21 |
| 2.2 | Equilibrium Points ..... | 23 |
| 2.3 | Finite Element ..... | 25 |
| 2.4 | Stiffness Method ..... | 30 |
| 3.1 | Energy Surface ..... | 36 |
| 3.2 | Search Directions ..... | 39 |
| 3.3 | DFP Algorithm ..... | 42 |
| 3.4 | FR Algorithm ..... | 48 |
| 4.1 | Frame 6A ..... | 54 |
| 4.2 | Frame 18A ..... | 55 |
| 4.3 | Frame 18B ..... | 56 |
| 4.4 | Frame 36A ..... | 57 |
| 4.5 | Frame 36B ..... | 58 |
| 4.6 | Frame 60A ..... | 59 |
| 4.7 | Frame 63B ..... | 60 |
| 4.8 | Execution Time vs. D.O.F. for ENERGY and STIFF ..... | 62 |
| 4.9 | Iterations per d.o.f. Required by FR and DFP ..... | 64 |
| 4.10 | Increase in Computer Storage for $[\tilde{K}]^{-1}$ Based on d.o.f. ..... | 66 |
| 4.11 | Effect of Problem Size on STIFF Execution Time ..... | 69 |
| 4.12 | Effect of Load Increments on ENERGY (DFP) and STIFF ..... | 71 |

| Figure | | Page |
|---|---|---|
| 4.13 | Effect of Additional Load Vectors on Number of Iterations Required by ENERGY (DFP) ..... | 72 |
| 4.14 | Cumulative Number of Iterations Required by ENERGY (DFP) when Load Increments are Applied ..... | 75 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 4.1 | Major Computer Storage Requirements of ENERGY (DFP) and STIFF ..... | 68 |

## NOTATION

| | |
|---|---|
| $A$ | cross-sectional area |
| $E$ | Young's modulus |
| $I$ | moment of inertia |
| $L$ | length |
| $U$ | total potential energy |
| $V$ | volume |
| $[\ ]$ | square matrix |
| $\{\ \}$ | column vector |
| $[A]$ | hessian matrix |
| $[\tilde{A}]$ | approximation of hessian matrix |
| $[K]$ | constrained stiffness matrix |
| $[\tilde{K}]^{-1}$ | approximate inverse hessian of the total potential |
| $\{d^k\}$ | $k^{th}$ search direction used in minimization algorithms |
| $\{g^k\}$ | gradient vector of total potential evaluated at $\{q^k\}$ |
| $\{q\}$ | generalized coordinates |
| $\{q^k\}$ | $k^{th}$ estimate of the deformation vector |
| $\{Q\}$ | load vector |
| $\nabla f(x)$ | gradient vector of $f(x)$ evaluated at $x$ |
| $\nabla^2 f(x)$ | hessian matrix of $f(x)$ evaluated at $x$ |
| $\alpha_k$ | scalar steplength used in $k^{th}$ 1-D minimization |
| $\varepsilon$ | strain per unit length |
| $\varepsilon_1, \varepsilon_2$ | limits on convergence criteria |
| $\Omega$ | potential energy of external forces |

| | |
|---|---|
| $\Pi$ | strain energy |
| $\Pi^*$ | strain energy density |
| $\sigma$ | normal stress per unit area |

# Chapter 1

# INTRODUCTION

This chapter contains the purpose and scope of this study. Included is a survey of the current literature on the variable metric and conjugate gradient methods of nonlinear function minimization. Emphasis has been placed on the minimization of quadratic functions. The effects of scaling on these algorithms has been included. Criteria for algorithm selection and the selection of the algorithms used in this study are also included.

## 1.1 Purpose and Scope

The purpose of this study is to compare the efficiency of two different approaches for finding the generalized displacement vector of a plane frame subject to static joint loads. The methods of analysis used are: the principle of minimum potential energy (the energy method) and the direct stiffness method of matrix structural analysis (stiffness method). The frames considered in this study are limited to small displacements and behave linearly. Evaluation is based on the amount of computational time required by each method to solve the same problem.

The energy method used in this study locates the generalized displacements of a structure by minimizing the total potential energy of that structure. The minimization algorithms used are iterative. The variable metric algorithms take information from a present iteration to improve the next iteration. Some of these algorithms exhibit

a special property: if there are no roundoff errors and the number of iterations required to find the solution is the same as the number of unknowns, the flexibility matrix of the structure is produced as a result of the minimization process. Because of roundoff errors only an approximation of the flexibility matrix can be expected. However, this approximation may be retained and used to reduce computational time when more than one load condition is applied to the same structure.

The stiffness method used in this study assembles the constrained stiffness matrix for a given structure and directly solves the resulting set of equilibrium equations. These equations are solved by Gaussian elimination. The algorithm used for the solution of the equilibrium equations requires a triangular decomposition of the stiffness matrix. When several load conditions are applied to the same structure the previously decomposed stiffness matrix may still be used.

Several frames have been considered to determine the efficiency of the energy method compared to the stiffness method. The effect of the number of degrees of freedom and configuration of a frame on the solution time required by both methods has been examined. A comparison of the rate of increase in computational time when additional load vectors are applied to the same structure is also given.

## 1.2 Literature Review

### a. Variable Metric Algorithms

Variable metric is the name given to a large group of algorithms

which have been developed that will minimize a nonlinear function of n variables. These algorithms are also referred to as quasi-Newton algorithms because an attempt is made to simulate Newton's method [35]. Newton's method requires second derivative information about the function being minimized. This information is contained in an (n x n) matrix known as the hessian. In addition to calculating the hessian matrix, Newton's method also requires the hessian be inverted. Working under the assumption that evaluation and inversion of the hessian is impractical or costly, the idea underlying quasi-Newton algorithms is to use an approximation to the inverse hessian in place of the true inverse that is required in Newton's method [27]. The variable metric (quasi-Newton) algorithms are iterative and require only first derivative information about the function being minimized. The approximation of the inverse hessian is improved during each iteration by using the information obtained during that iteration and the previous iteration. This process will continue until some convergence criteria, which sufficiently defines a minimum value of the function, are satisfied.

The variables of a function may be considered as coordinates of a point in an n dimensional linear space. The set of variables for which the function has a constant value forms an n-1 dimensional surface in this space [8]. In Newton's method, the inverse hessian is a measure of distance in this space, a metric. A continual update of the approximate inverse hessian suggests referring to the iterative process as a variable metric algorithm.

The first variable metric algorithm was developed, and so named, in 1959 by Davidon [8]. The original method devised by Davidon was simplified by Fletcher and Powell [14] who also proved convergence and stability criteria for their simplified version of Davidon's original work. The modified method is known as the Davidon-Fletcher-Powell method, DFP. The convergence criteria proved by Fletcher and Powell for the DFP method was that of quadratic convergence. For a quadratic function, the method will locate the minimum of the function in at most n iterations. The stability criteria proved was that for each iteration the value of the function being minimized is reduced.

Since the first introduction of the variable metric method there have been numerous new algorithms developed to update the inverse hessian. There has also been the development of families of algorithms. These families have constants or parameters that may be changed to lead to new algorithms or demonstrate that most previously developed algorithms belong to a particular family. The first of these families was developed by Broyden [5] and another very important family was developed by Huang [21]. Dixon [10,11] has indicated the sequence of points generated in minimizing a general function by different members of Broyden's family are identical when the line search is exact and that Broyden's family is a subset of Huang's family. A survey on some of the methods used for construction of variable metric algorithms is given by Oren [35].

The construction of Huang's family illustrates the theory basic to many of the variable metric algorithms and uses four properties that

are essential to the theory:

a) the algorithms use one dimensional search only

b) for a quadratic function, the algorithms are capable of converging quadratically to the minimal point

c) the algorithms employ the function and its gradient only

d) the algorithms employ only information at the present iteration and the iteration immediately previous to the present

The requirement of one dimensional search by property (a) reduces the n dimensional minimization problem to a series of linear minimization problems, each of which is solved during each iteration. The search is carried out in the space defined by the variables of the function; for each iteration a new search direction is chosen. This is continued until a minimum of the function is located. Property (b) is of importance because even a nonquadratic function behaves approximately quadratically in the neighborhood of a minimal point [21]. Property (c) identifies the algorithms as only needing first derivative information and property (d) requires the only information that must be retained is that from the previous iteration. This takes advantage of previous information about the function being minimized to improve the next iteration.

The DFP method is probably the most widely known variable metric algorithm and one of the most successful which has the following

property. If for a positive definite quadratic function the convergence requires the full n iterations, then the $n^{th}$ approximation of the inverse hessian is identically equal to the inverse hessian [31]. To guarantee the minimum will be located in no more than n interations, quadratic convergence must be insured, which it is if the linear minimization problem or line search is done exactly [25]. Another variable metric method that has been introduced more recently is the Broyden-Fletcher-Goldfarb-Shanno, BFGS, algorithm. This algorithm is a member of Broyden's family and is considered to be the most successful member from Broyden's family in practice [4]. For general function minimization Brodlie [4] shows that for test functions in his study the BFGS algorithm was at least equivalent to or superior to the DFP algorithm.

There have been new algorithms proposed that eliminate the need for an accurate solution to the linear minimization problem. A symmetric rank one algorithm that avoids accurate line search was proposed by Fletcher [15]. In this algorithm quadratic convergence has been replaced by a property in which the approximate inverse hessian tends to the inverse hessian. A proof is given by Vial and Zang [39] that for a particular rank one algorithm without exact line search convergence will occur in at most n+1 iterations for a nonsingular quadratic function and if n+1 iterations are used the approximate inverse hessian will be the inverse hessian. Norris and Gerken [29] have presented a rank one method with a limited line search but are unable to prove convergence for quadratic functions

with eigenvalues both larger and smaller than one. However the numerical results given by Norris and Gerken indicate their method superior to the DFP algorithm for the nonlinear functions in their study.

The most recent innovation in variable metric algorithms has been the development of self-scaling variable metric, SSVM, algorithms. A family introduced by Oren [31], in which the particular algorithm is dependent upon two parameters, produces the effect of scaling the function being minimized by scaling the update formula used to approximate the inverse hessian. The numerical results in [31] indicate promise for SSVM algorithms. Further studies by Oren and Luenberger [32], Oren [33], Brodlie [4], and Spedicato [38] indicate a SSVM algorithm superior to a conventional variable metric algorithm for a general function with a large number of variables. However, for some functions the SSVM algorithms have been shown to deteriorate with an increase in variables, suggesting there are particular functions to which they are best suited to minimize [4]. It was shown by Spedicato [38] that for a Hilbert quadratic, with the number of variables ranging from 5 to 80, the performance of a conventional variable metric algorithm and a SSVM algorithm was practically identical.

Theoretically the SSVM and DFP algorithms perform the same for an exact line search when minimizing a quadratic function [32]. However, if the scale factor is varied at each iteration there is no guarantee that the approximate inverse hessian will be the inverse

hessian for a positive definite quadratic function [33]. The performance of a SSVM algorithm critically depends upon the selection of parameters that scale the update formula. The selection of parameters reported by Oren and Spedicato [34] are heuristic in nature but appear to give good results.

b. Conjugate Gradient Algorithms

The conjugate gradient algorithms for minimization of nonlinear functions have developed from the introduction of a method for solving a system of linear equations developed by Hestenes and Stiefel [18] and later work done by Beckman [3]. Conjugate gradient algorithms applied to the minimization of nonlinear functions locate a minimum value of a function by solving a series of linear minimization problems. This is similar to the way the variable metric algorithms perform. In fact, Myers [28] proves that DFP algorithms and conjugate gradient algorithms, such as the Fletcher-Reeves algorithm, theoretically generate search directions that are positive scalar multiples of each other for a quadratic function if the initial search direction is the steepest descent. The difference in the algorithms is the way in which they generate the search directions.

The linear minimizations take place along a series of search directions which have been constructed such that they are A-conjugate for a positive definite quadratic function. A-conjugate directions require [19]:

$$\{d^i\}^T[A]\{d^j\} = 0 \qquad (1.1)$$

where

$1 \leq i \neq j \leq n$

$\{d^i\}$ = search direction

$[A]$ = hessian matrix

The importance of a quadratic function is the same as mentioned for the variable metric algorithms.

The conjugate gradient method is iterative and the process is terminated when some convergence criteria are satisfied. The gradient at the previous iteration and the gradient at the present iteration are required to construct the A-conjugate directions. There is no need to compute the hessian or its inverse. The gradient is a vector whose negative identifies the direction of greatest decrease, locally, in the value of a function, the direction of steepest descent [17]. It has been shown by Allwright [1] that if the first search direction is the steepest descent, the conjugate gradient algorithms are superior to continuing to search in directions of steepest descent in minimizing a quadratic function.

The convergence properties of three conjugate gradient algorithms have been studied by Cohen [6]. The algorithms, Polak and Ribière, Daniel and Fletcher-Reeves all behave identically for a quadratic function. For a large class of nonlinear functions reinitializing the search direction periodically (when the number of iterations for each period is greater than or equal to the number of variables) they converge n-step quadratically [6]. Lenard [26] has also shown that theoretically the Fletcher-Reeves and Polak-Ribière algorithms exhibit

n-step quadratic convergence when restarted under certain conditions. The difference in the above algorithms is in the way they compute the conjugate directions. The Fletcher-Reeves method [13] is the simplest of the three to implement and appears quite frequently in the literature and texts [13,19,27] which demonstrate the properties of conjugate gradient methods.

To guarantee quadratic convergence the initial search direction of the conjugate gradient algorithms must be the steepest descent [37]. However for a poorly conditioned quadratic function it can take more than n iterations to reach the minimum, due fundamentally to the finite digit arithmetic in which all calculations must be performed [17]. This has been shown in the numerical experiences of Fox and Stanton [16].

c. Scaling Applications

The variable metric and conjugate gradient algorithms both search for a minimum in the space defined by the variables. The search directions make use of the curvature of the surface defined by the variables. The value of scaling can easily be illustrated by Fig. 1.1. Figure 1.1a is a series of ellipsoids which can be considered as contours formed by constant values of a quadratic function. Figure 1.1b is a representation of the same quadratic function that has undergone a scaling transformation to produce a series of contours which are circles.

Consider a search made in the direction of steepest descent initially in both Figs. 1.1a and 1.1b. In Fig. 1.1a the first search

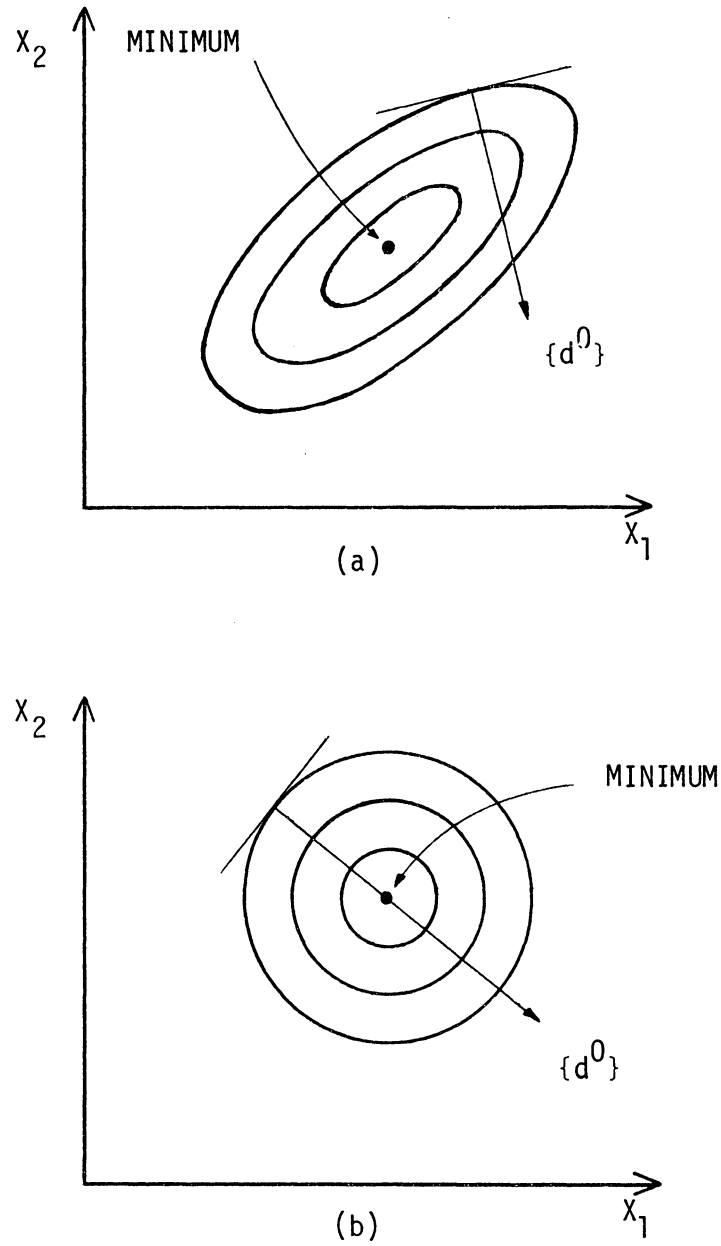



FIGURE 1.1 OPTIMALLY SCALED 2 VARIABLE QUADRATIC FUNCTION

direction will not lead to the minimum unless it should happen to be along a principle axis. While in Fig. 1.1b the first search direction will find the minimum regardless of where it originates from. Figure 1.1b is the result of optimally scaling the function represented by Fig. 1.1a by minimizing its eccentricities (the contours defined by ellipses were transformed to circles). However, in practice proper scaling of a function with many variables can be a difficult task since there is no set of rules to follow for all functions.

A method of reducing the condition number of the hessian of a positive definite quadratic function to being near optimally scaled is presented by Fox and Stanton [16]. A low condition number will reduce roundoff error and improve stability [34]. This method is easy to implement but requires the hessian of the function be known. The numerical results of [16] showed a poorly scaled function that could not be minimized by the Fletcher-Reeves conjugate gradient algorithm was minimized in less than n iterations by the same algorithm after scaling. The DFP algorithm used did however minimize the poorly scaled function, but did it at an improved rate after scaling. It should be noted scaling of this type requires a coordinate transformation and the function has to be known in matrix notation.

The effect of scaling the function can be created in variable metric algorithms by initial scaling of the approximate inverse hessian only or by scaling the update formula during each iteration [2,4,31,32,33,34,38]. For general nonlinear functions, scaling during each iteration appears superior to methods which do not employ

scaling [4,33,38]. In numerical experiments Spedicato [38] scaled the initial approximate inverse hessian by letting it be a diagonal matrix whose elements were those of the true inverse hessian at a minimum. For general nonlinear functions it was competitive with methods that scaled the update at each iteration. But it was not as efficient in minimizing a Hilbert quadratic with a large number of variables as when the initial matrix used in estimating the inverse hessian was the identity matrix. Indeed, it has been proven by Bard [2] improper scaling may worsen the problem and cause a variable metric algorithm to become unstable.

## 1.3 Algorithm Selection

The selection of the algorithm used in this study is based upon which of the algorithms surveyed best satisfies the requirements given in this section. The problem will be to minimize the total potential energy function of an assemblage of linear beam-column elements. The function may be expressed by the following quadratic form:

$$f(x) = \frac{1}{2} \{x\}^T [A] \{x\} - \{b\}^T \{x\} \qquad (1.2)$$

where

$\{x\}$ = variables

$[A]$ = hessian matrix

$\{b\}$ = vector of constants

Equation 1.2 will be minimized several times where the hessian matrix remains constant while the vector of constants is changed (in structural analysis this corresponds to applying different load

vectors to the same structure).

The variable metric algorithms solve the following one dimensional minimization problem during each iteration:

$$\{x^{k+1}\} = \{x^k\} - \alpha_k[\tilde{A}_k]^{-1}\{g^k\} \tag{1.3}$$

where

$\alpha_k$ = scalar step length that minimizes f(x) along the direction of search

$[\tilde{A}_k]^{-1}$ = estimate of inverse hessian at $\{x^k\}$ (updated during each iteration)

$\{g^k\}$ = gradient vector evaluated at $\{x^k\}$

Because of the form of Eq. 1.2 the step length and gradient vector can be computed explicitly. Knowing the step length explicitly allows for an "exact" line search (dependent upon the precision of the math used). Explicit gradient evaluation also improves the numerical qualities of the algorithms [23].

Because it is desired to minimize Eq. 1.2 with several different values of the vector of constants while the hessian remains the same, the update of the inverse hessian in Eq. 1.3 becomes of special interest. If the inverse hessian is accurately approximated during the first minimization of Eq. 1.2 then it may be used as the initial estimate of the inverse hessian during the next minimization of Eq. 1.2. When Eq. 1.3 is applied it will now be reduced to Newton's method where the step length will be unity and the minimum will be located in a single iteration, a full Newton step. Therefore, the only algorithms that

will be considered are those that guarantee in n iterations the approximate inverse hessian will be the true inverse hessian; quadratic convergence is a necessary but not sufficient requirement for this.

The choice of algorithms was the Davidon-Fletcher-Powell variable metric algorithm which is supported by the literature surveyed as a good selection. It satisfies quadratic convergence and the approximate inverse hessian is the inverse hessian after n iterations. The method is also easily implemented into a computer program. Another possible choice could have been the Broyden-Fletcher-Goldfarb-Shanno, BFGS, algorithm. It appears to be preferred over the DFP algorithm in the minimization of some nonlinear functions. No indication is given, however, that the BFGS algorithm is superior to the DFP algorithm in minimizing quadratic functions, which fully satisfies the requirements of this section.

The algorithms that relax the line search were immediately rejected since they do not guarantee quadratic convergence and a relaxed line search is of no value in this study since the step length can be computed exactly. The self-scaling variable metric algorithms relinquish the property that the approximate inverse hessian will be the inverse hessian in n iterations if scaling is applied during each iteration. This eliminates SSVM algorithms even if quadratic convergence is satisfied.

Although the conjugate gradient algorithms do not compute the inverse hessian as a result of minimizing a function, they have two properties which make them attractive:

a) easily implemented into a computer code requiring modest storage

b) in the process of minimizing a positive definite quadratic function A-conjugate directions are generated

Property (a) is important from a programming standpoint but more important because of savings in cost in terms of computer time. Conjugate gradient algorithms only require matrix operations involving (n x 1) column vectors while variable metric algorithms require matrix operations involving (n x 1) column vectors as well as manipulation of a full (n x n) matrix. Although the (n x n) matrix is symmetric and this may be taken advantage of in storage, the variable metric algorithms still require much more computational effort than the conjugate gradient algorithms.

Property (b) is of importance because of the following relation:

$$[A]^{-1} = \sum_{i=1}^{n} \frac{\{d^i\}\{d^i\}^T}{\{d^i\}^T[A]\{d^i\}} \tag{1.4}$$

where

$\{d^i\}$ = search direction at $i^{th}$ iteration

If the search directions are A-conjugate and n iterations are used in minimizing the function, Eq. 1.4 can be used to compute the inverse hessian as a by-product of the information required by the conjugate gradient algorithm (although Eq. 1.4 requires the storage of an (n x n) symmetric matrix, considerably less computational effort is required

to compute the inverse hessian by this method than by an update formula required by a variable metric algorithm). The result of Eq. 1.4 can now be used in Eq. 1.3 as the approximate inverse hessian. For a new vector of constants the minimum should be located in one iteration as previously stated.

The only requirement for a conjugate gradient algorithm is that A-conjugate search directions are generated; this implies quadratic convergence [19]. The methods which employ relaxed line search will not be used because quadratic convergence is not guaranteed. The algorithms of Fletcher-Reeves, Polak-Ribière and Daniel perform identically when applied to minimizing a quadratic function. They all are guaranteed quadratic convergence. The Fletcher-Reeves algorithm has the simplest method of generating conjugate directions. Because of its simplicity and quadratic convergence the FR algorithm was chosen. Although scaling has not been used in the algorithms selected and will not be used in the first minimization of Eq. 1.2 it is inferred in the second and subsequent minimization of Eq. 1.2. The use of the inverse hessian obtained during the first minimization as the approximate inverse hessian for the next minimization is actually the best scaled choice possible, the inverse hessian of the function being minimized.

# Chapter 2

# PROBLEM FORMULATION

This chapter contains the energy principles used in this study and a brief outline of the direct stiffness method of structural analysis.

## 2.1 Energy Method

### a. Total Potential Energy

The following assumptions are used in this study: a) the system is conservative; b) the system behaves linearly; c) the body weight of the system is neglected; d) external forces consist of only joint loads; e) all members have prismatic cross sections; f) deformations caused by external forces occur infinitesimally slowly; g) all processes are adiabatic.

The total potential of a structural system is formulated in terms of the generalized coordinates of that structure and may be expressed by the following relation:

$$U = \Pi + \Omega \tag{2.1}$$

where

$U$ = total potential

$\Pi$ = total potential energy of internal forces

$\Omega$ = total potential energy of external forces

The total potential energy of the external forces is the negative of the work performed by these forces. The forces applied to structures in this study are concentrated forces and moments. The

values of the forces are constant and therefore independent of displacements or rotations. These external forces will undergo displacements expressed in units of length and rotations expressed in the nondimensional units of radians respectively. The external forces are applied in the direction of the generalized coordinates of the structure and are represented by the following load vector:

$$\{Q\}^T = \{Q_1 \quad Q_2 \quad Q_3 \ldots\ldots Q_n\} \tag{2.2}$$

The displacements and rotations corresponding to the load vector are represented by the following vector:

$$\{q\}^T = \{q_1 \quad q_2 \quad q_3 \ldots\ldots q_n\} \tag{2.3}$$

The total potential of the external forces, which is referred to as load potential, may be expressed as the negative of the scalar product of the preceding two vectors (where Q is constant).

$$\Omega = - \{Q\}^T\{q\} \tag{2.4}$$

The strain energy is a result of the internal forces developed in a structure. These internal forces are a result of the stresses caused by the deformation of the structure. The strain energy of a structure in a given configuration is equal to the work done in the absence of external forces to lead the structure slowly and adiabatically from the zero configuration to the given configuration [24]. The strain energy is the total potential of the internal forces. Neglecting the effects of shear, the strain energy for a linear beam-

column element is composed of two parts, the strain energy due to axial deformation and the strain energy due to bending. The strain energy of such an element may be computed by the following equation:

$$\Pi = \iiint_V \Pi^* dV \tag{2.5}$$

where

$$\Pi^* = \int_0^{\varepsilon} \sigma d\varepsilon \tag{2.6}$$

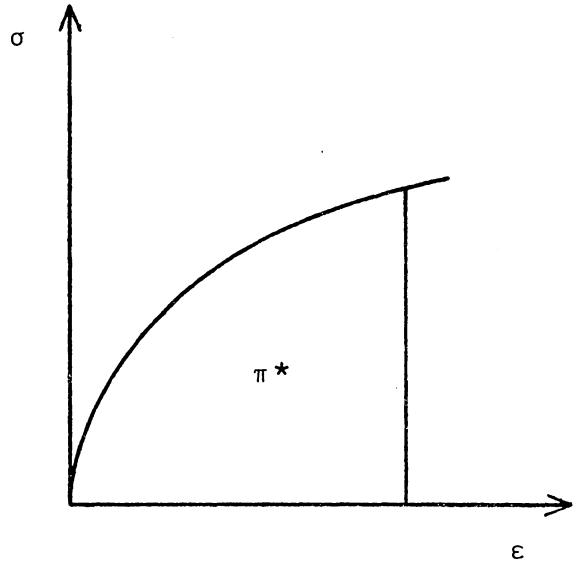
Equation 2.6 is the expression for strain energy density, the value of the strain energy at a point. The strain energy density can be illustrated graphically. Figure 2.1a represents the strain energy density for a general material and Fig. 2.1b represents the strain energy density for a material that obeys Hooke's law. From Fig. 2.1b and Hooke's law Eq. 2.6 can be rewritten as:

$$\Pi^* = \frac{\sigma^2}{2E} \tag{2.7}$$

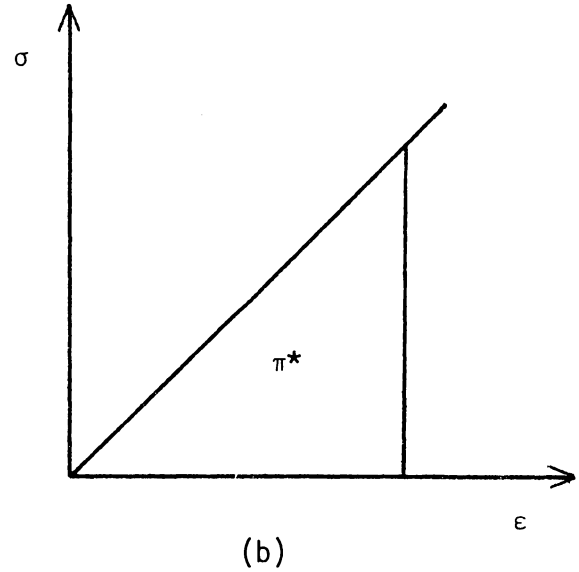
The strain energy density of a linear beam-column element is computed by substituting the relation for normal stress due to bending and axial displacement into Eq. 2.7. This value of the strain energy density is then substituted into Eq. 2.5. After simplification the strain energy in the linear beam-column element can be expressed as:

$$\Pi = \Pi_b + \Pi_a \tag{2.8}$$

where




(a)




(b)

FIGURE 2.1 STRAIN ENERGY DENSITY

$$\Pi_b = \frac{EI}{2} \int_0^L \left(\frac{\partial^2 y}{\partial x^2}\right)^2 dx \tag{2.9}$$

$$\Pi_a = \frac{AE}{2L} \Delta x^2 \tag{2.10}$$

$\Delta x$ = change in axial length

b. Principle of Minimum Potential Energy

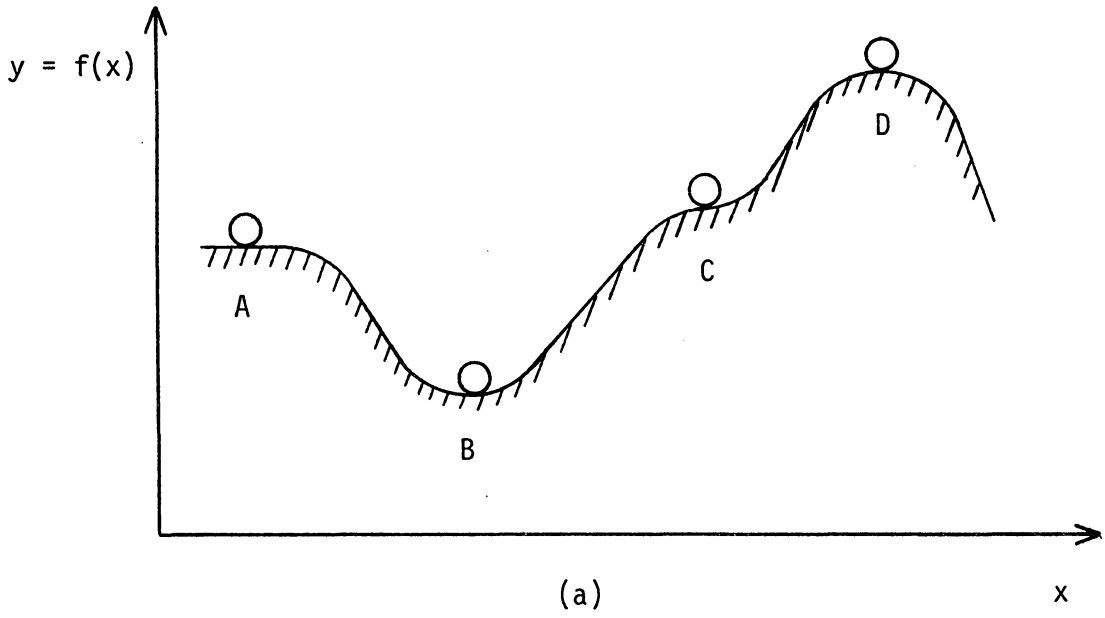
The basis for the energy method used in this study is the principle of minimum potential energy. This principle may be stated as:

> Of all the displacements which satisfy the boundary conditions of a structural system, those corresponding to stable equilibrium configurations make the total potential energy a relative minimum [30].

This principle can be illustrated by Figs. 2.2a and 2.2b. Figure 2.2a is a conservative system which represents different locations of a rigid ball of weight W on a fixed contour. Figure 2.2b is a plot of the total potential of the ball for locations corresponding to those in Fig. 2.2a. The total potential of the ball at any point may be expressed as:

$$U = Wy \tag{2.11}$$

In Fig. 2.2a, points A, B, C and D represent equilibrium points. The stability of these points is determined by their behavior when they are subject to any possible small disturbance caused by an external force. If the ball at point A is slightly disturbed it will move to a new point and remain there. Since there is no change in total potential, this represents neutral equilibrium. If the ball at point B is



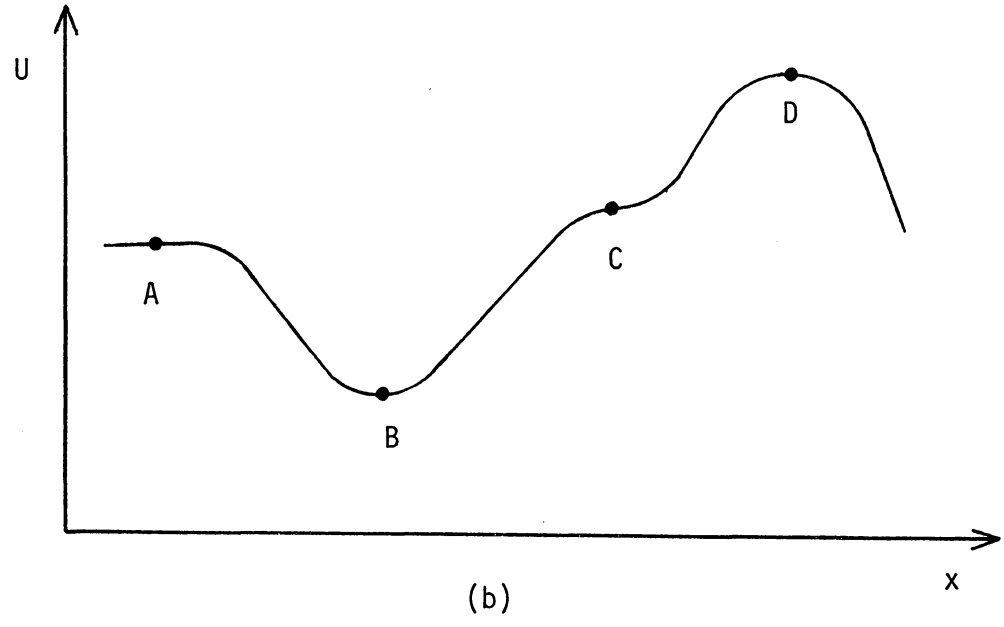




FIGURE 2.2 EQUILIBRIUM POINTS

slightly disturbed, it will oscillate about its original point, any disturbance causes an initial increase in total potential. This represents stable equilibrium. If the ball at point C is slightly disturbed, there is either an increase in total potential initially or a continuous decrease in total potential. This is unstable equilibrium. For any displacement of the ball at point D, there is an initial decrease in the total potential, which is also unstable equilibrium. From these conditions of equilibrium the only stable equilibrium point is that for which all possible small disturbances cause an increase in total potential. This requires the value of the total potential to be a relative minimum as stated in the principle of minimum potential energy.

c. Total Potential of a Structural System and its Gradient Vector

The total potential of a continuous structure is computed by modeling it as an assemblage of finite elements. The total potential of the structure may then be computed as the sum of the strain energy of each element added to the sum of the load potentials at all nodes.

A general finite element is represented by Fig. 2.3. The load potential is computed in terms of the global coordinates and the strain energy is computed in terms of the local coordinates. The load potential of an entire structure determined by Eq. 2.4 may be rewritten as:

$$\Omega = - \sum_{i=1}^{NDF} Q_i q_i \qquad (2.12)$$

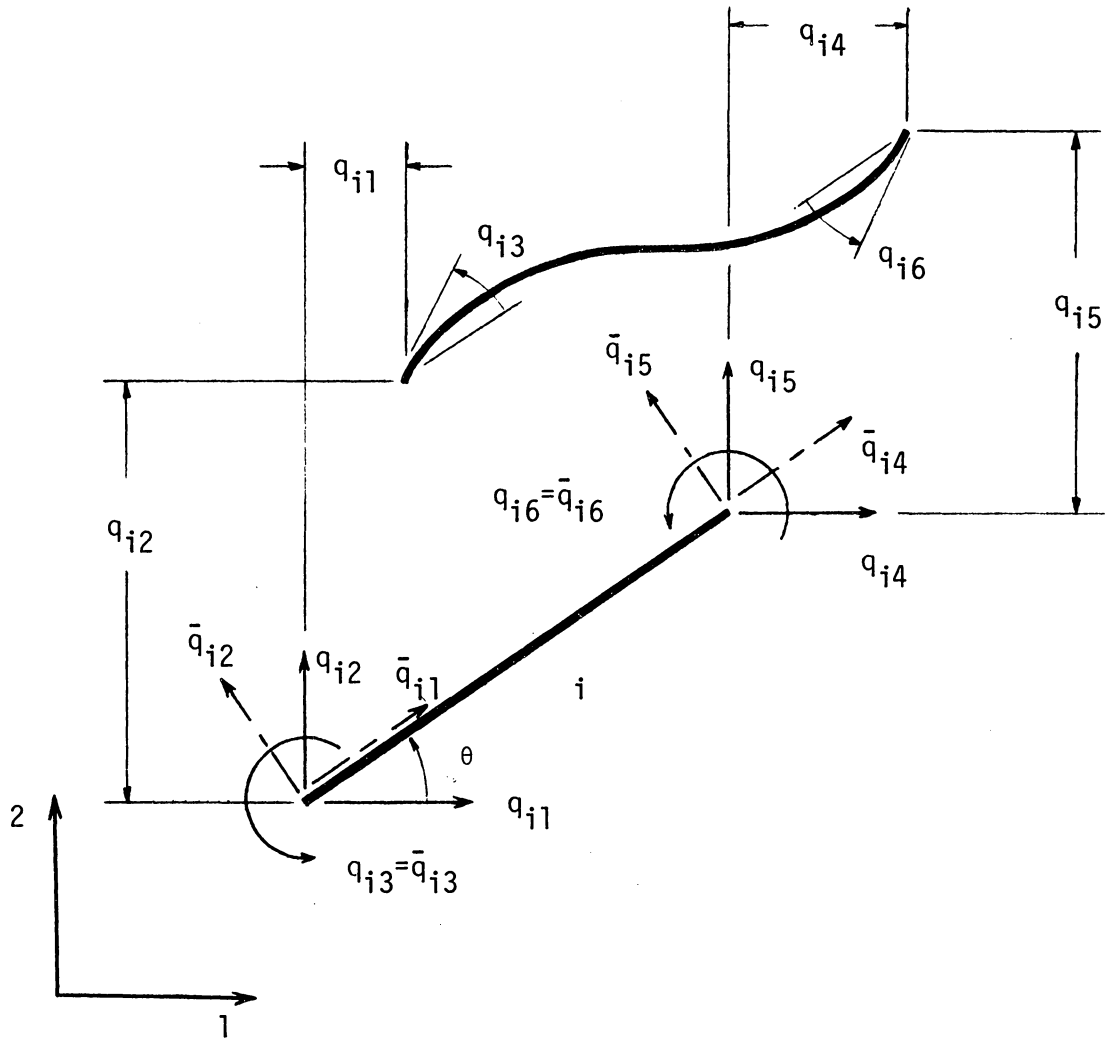



FIGURE 2.3 FINITE ELEMENT

where

NDF = number of degrees of freedom

The strain energy is computed in two parts, the strain energy due to bending and the strain energy due to axial displacement.

In computing the strain energy the local coordinates are defined as components of the global coordinates in accordance with Fig. 2.3

$$\bar{q}_{i1} = q_{i2}s_i + q_{i1}c_i \tag{2.13a}$$

$$\bar{q}_{i2} = q_{i2}c_i - q_{i1}s_i \tag{2.13b}$$

$$\bar{q}_{i3} = q_{i3} \tag{2.13c}$$

$$\bar{q}_{i4} = q_{i5}s_i + q_{i4}c_i \tag{2.13d}$$

$$\bar{q}_{i5} = q_{i5}c_i - q_{i4}s_i \tag{2.13e}$$

$$\bar{q}_{i6} = q_{i6} \tag{2.13f}$$

where

$q_{ij}$ = global coordinate of element i in j direction

$\bar{q}_{ij}$ = local coordinate of element i in j direction

$c_i$ = cos$\theta$ of element i

$s_i$ = sin$\theta$ of element i

The strain energy due to bending is based on the transverse deflection, y(x), at any point along the longitudinal axis of the

element. The function y(x) can be obtained by solving the differential equation of a beam in bending and applying the local coordinates which contribute to bending as the boundary conditions. This expression for y(x) is used in Eq. 2.9 where the local coordinates are replaced by components of the global coordinates yielding the following expression for strain energy due to bending in element i:

$$\Pi_{bi} = \frac{E_i I_i}{2L_i} \{\phi_i^2 + 2\phi_i \xi_i + \frac{4}{3} \xi_i^2\} \tag{2.14}$$

where

$$\phi_i = \frac{6}{L_i} \{q_{i1} s_i - q_{i2} c_i - \frac{2}{3} q_{i3} L_i - q_{i4} s_i + q_{i5} c_i - \frac{L_i}{3} q_{i6}\} \tag{2.15}$$

$$\xi_i = \frac{6}{L_i} \{-q_{i1} s_i + q_{i2} c_i + \frac{1}{2} q_{i3} + q_{i4} s_i - q_{i5} c_i + \frac{L_i}{2} q_{i6}\} \tag{2.16}$$

The strain energy due to axial deformation is computed by using Eq. 2.10. Again, the local coordinates are replaced by components of the global coordinates to obtain for element i:

$$\Pi_{ai} = \frac{A_i E_i}{2L_i} \beta_i^2 \tag{2.17}$$

where

$$\beta_i^2 = -q_{i1} c_i - q_{i2} s_i + q_{i4} c_i + q_{i5} s_i \tag{2.18}$$

The strain energy of any element, i, can now be expressed as the sum of Eqs. 2.14 and 2.17:

$$\Pi_i = \frac{E_i I_i}{2L_i} \{\phi_i^2 + 2\phi_i \xi_i + \frac{4}{3} \xi_i^2\} + \frac{A_i E_i}{2L_i} \beta_i^2 \tag{2.19}$$

The total potential of a system of linear beam column elements is represented by:

$$U = \sum_{i=1}^{NEL} \left[\frac{E_i I_i}{2L_i}\left\{\phi_i^2 + 2\phi_i\xi_i + \frac{4}{3}\xi_i^2\right\} + \frac{A_i E_i}{2L_i}\beta_i^2\right]$$

$$- \sum_{j=1}^{NDF} Q_j q_j \tag{2.20}$$

where

NEL = Number of elements

The gradient vector (see section 3.1 for definition and structural interpretations) required by the minimization algorithms consists of the following contributions from the strain energy in each element:

$$\frac{\partial \Pi_i}{\partial q_{i1}} = -\frac{A_i E_i}{L_i}\beta_i c_i - \frac{2E_i I_i}{L_i^2}\xi_i s_i \tag{2.21}$$

$$\frac{\partial \Pi_i}{\partial q_{i2}} = -\frac{A_i E_i}{L_i}\beta_i s_i + \frac{2E_i I_i}{L_i^2}\xi_i c_i \tag{2.22}$$

$$\frac{\partial \Pi_i}{\partial q_{i3}} = -\frac{E_i I_i}{L_i}\phi_i \tag{2.23}$$

$$\frac{\partial \Pi_i}{\partial q_{i4}} = -\frac{\partial \Pi_i}{\partial q_{i1}} \tag{2.24}$$

$$\frac{\partial \Pi_i}{\partial q_{i5}} = -\frac{\partial \Pi_i}{\partial q_{i2}} \tag{2.25}$$

$$\frac{\partial \Pi_i}{\partial q_{i6}} = \frac{E_i I_i}{L_i}\{\phi_i + 2\xi_i\} \tag{2.26}$$

The contribution to each component, j, of the gradient vector from the load potential is:

$$\frac{\partial \Omega}{\partial q_j} = - Q_j \tag{2.27}$$

Each component of the gradient vector may now be computed by using:

$$\frac{\partial U}{\partial q_j} = \sum_{i=1}^{NEL} \frac{\partial \Pi_i}{\partial q_j} - Q_j \tag{2.28}$$

where

$q_{in}$, n = 1,2,..,6 used in Eqs. 2.21 through 2.26 have been replaced by the corresponding nodal degree of freedom $q_j$

## 2.2 Stiffness Method

The purpose of structural analysis is to determine the state of deformation or stress of a structure, based on known force-displacement characteristics of the component parts, and satisfying all the conditions of compatibility and equilibrium [22]. The analysis technique described in this section is matrix structural analysis via stiffness method [20]. The stiffness method is well known and highly suited to computer programming and analysis of large structures. Because of its notoriety only a brief outline of the stiffness method used in this study is presented. This method is used in comparison with the energy method used in this study.

This section is developed around the flow chart of the stiffness method shown in Fig. 2.4 [20]. The element model is represented in

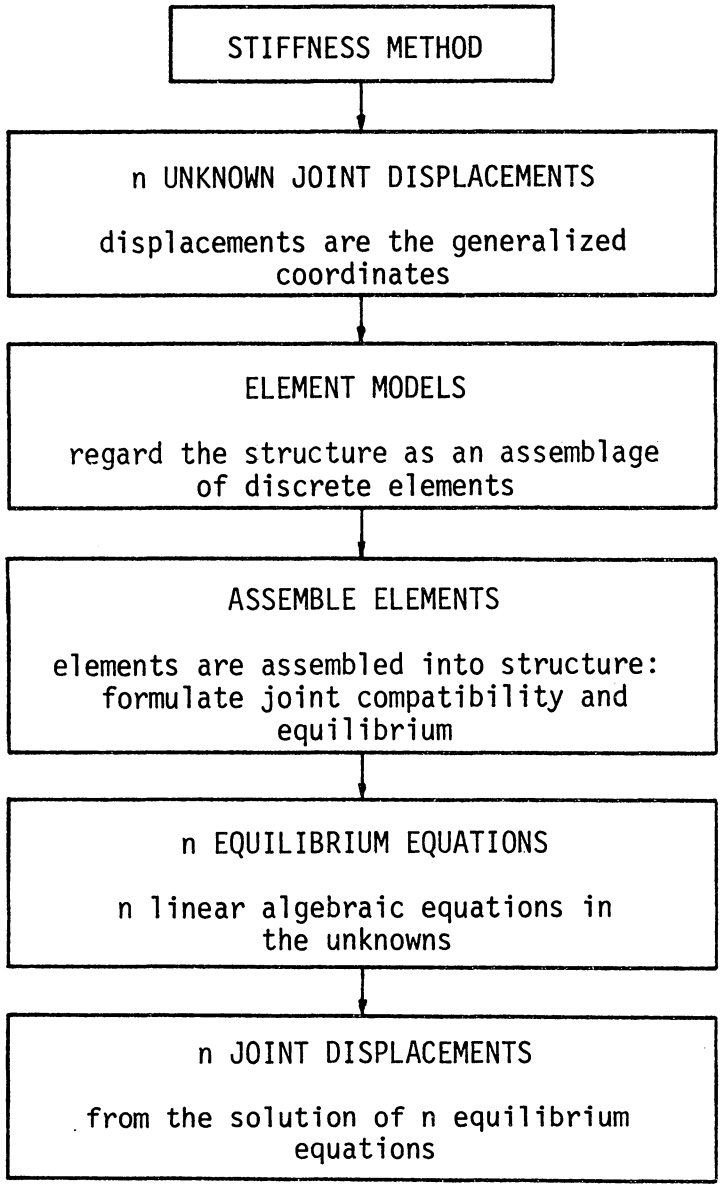



FIGURE 2.4 STIFFNESS METHOD

local coordinates by the force-displacement relation:

$$\{f\} = [k]\{u\} \tag{2.29}$$

where

$\{f\}$ = force vector

$[k]$ = element stiffness matrix

$\{u\}$ = unknown displacements

The elements of $[k]$, $k_{ij}$, are coefficients of the displacements defined as the force at i induced by a unit displacement at j while all other displacements are constrained. Any general element may be represented by the same finite element in Fig. 2.3 with the exception that all $\bar{q}_{ij}$ are replaced by $u_{ij}$.

Before assembling the element models to form the system model each element stiffness matrix must be transformed from the local form to the global form. This is done by the following transformation

$$[K] = [R][k][R]^T \tag{2.30}$$

where

$$[R] = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \tag{2.31}$$

$$\lambda = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.32}$$

$c = \cos\theta$
$s = \sin\theta$

The system stiffness matrix is assembled from the global element models so that compatibility at the nodes is guaranteed. The assemblage is done by the degrees of freedom of the system so that constraints are omitted in the process, thereby generating the constrained stiffness matrix directly. Since the stiffness matrix is symmetric and usually banded, only the elements contained in the upper semiband width are computed.

Equilibrium at the joints requires that the joint forces must be balanced by the corresponding element forces. This allows the equilibrium equations to be expressed as:

$$\{Q\} = [K]\{q\} \tag{2.33}$$

where

$[K]$ = stiffness matrix

$\{Q\}$ = applied joint loads

$\{q\}$ = deformation vector in global coordinates

Equation 2.33 is a system of linear equations whose solution is the unknown joint displacements.

# Chapter 3

# SOLUTION PROCESS

This chapter identifies the structural interpretations of some of the mathematical relations used in the Davidon-Fletcher-Powell variable metric and Fletcher-Reeves conjugate gradient algorithms. The DFP algorithm is developed taking advantage of known characteristics of the total potential. These characteristics are also applied to the FR algorithm. The method of solving the set of equlibrium equations generated by the stiffness method is also included.

## 3.1 Structural Interpretations of Mathematical Relations Used by the Minimization Algorithms

There are several important relations used in the minimization algorithms that have significant interpretations when applied to structural analysis. The relations used in the minimization algorithms and their counterparts in structural analysis are linked together by the hessian matrix and gradient vector of the function being minimized, the total potential. The total potential defined by Eq. 2.20 is a quadratic function and may be represented in matrix notation as:

$$U = \frac{1}{2} \{q\}^T [K] \{q\} - \{Q\}^T \{q\} \tag{3.1}$$

where

$$\Pi = \frac{1}{2} \{q\}^T [K] \{q\} \tag{3.2}$$

$$\Omega = - \{Q\}^T \{q\} \tag{3.3}$$

[K] = stiffness matrix

{Q} = load vector

The hessian is a square matrix of second partial derivatives of a function, f(x), evaluated at x and may be represented as:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & & & \vdots \\ \vdots & & & \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix} \tag{3.4}$$

The hessian of a quadratic function results in a matrix of constant coefficients. The hessian of the strain energy of a structure composed of linear beam-column elements is the stiffness matrix of that structure. In the case of Eq. 3.1 where the load vector is not a function of the displacement, the hessian of the total potential is the same as the hessian of the strain energy:

$$\nabla^2 U = [K] \tag{3.5}$$

$$\nabla^2 \Pi = [K] \tag{3.6}$$

In mathematical terms the hessian can be used to identify the type of surface defined by a function. Equation 3.1 defines a strictly convex surface; this is a surface that has only one minimum value and no

inflection points. This is determined by the fact that the stiffness (or hessian) matrix for a linear beam-column element is positive definite, which is also a stability requirement in structural analysis. This type of surface is represented in Fig. 3.1 for a two degree of freedom system. The energy surface is an elliptical hollow; the contours are ellipses and represent constant energy values.

The importance of the hessian to the DFP algorithm is that an approximation of its inverse is used in generating the search direction to be used during each iteration in minimizing a function. These directions are generated as follows:

$$\{d^k\} = - [\tilde{K}_k]^{-1}\{g^k\} \tag{3.7}$$

$\{d^k\}$ = $k^{th}$ search direction

$\{g^k\}$ = gradient vector evaluated at $\{q^k\}$

$[\tilde{K}_k]^{-1}$ = approximate inverse hessian at the $k^{th}$ iteration

The inverse of the hessian provides a measure of the curvature of the function in the vicinity where it is evaluated [19]. The approximate inverse hessian used in Eq. 3.7 allows the use of an approximation of the curvature of the function being minimized to be used in generating the search directions. The approximate inverse hessian is in turn updated during each iteration, therefore accumulating information about the curvature of the function during each iteration.

The significance of the inverse hessian of Eq. 3.1 is that it is the flexibility matrix of the structure represented by Eq. 3.1. The elements of the flexibility matrix are referred to as flexibility

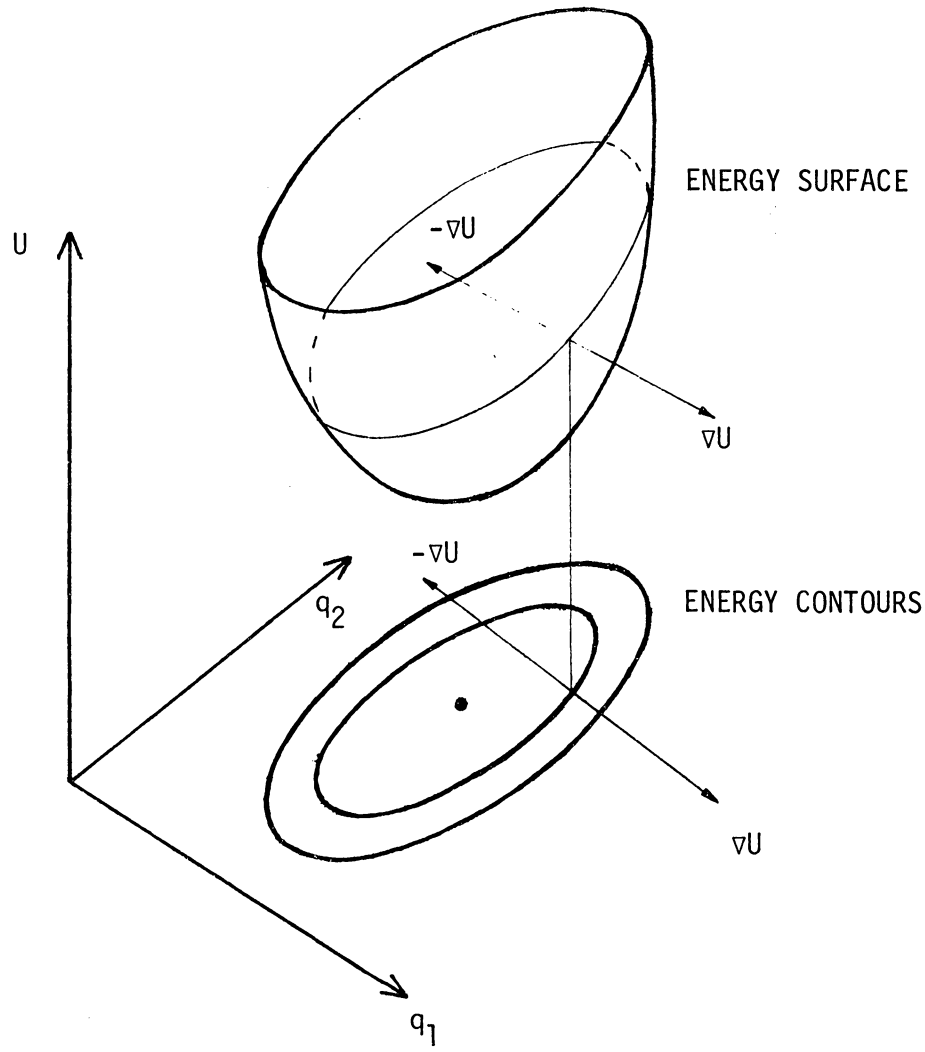



FIGURE 3.1 ENERGY SURFACE

influence coefficients. The coefficients are identified as $f_{ij}$ which is the displacement in the direction of i due to a unit force in the direction of j [22]. Once the flexibility matrix is known the deformation vector may be solved for directly by simple matrix multiplication:

$$\{q\} = [K]^{-1}\{Q\} \tag{3.8}$$

where

$[K]^{-1}$ = flexibility matrix (or inverse hessian)

If the flexibility matrix can be obtained as a by-product of the DFP algorithms solution process it must be considered as a valuable piece of information when several load vectors are considered for the same structure. This is indicated by Eq. 3.8.

The gradient is the column vector of first partial derivatives of a function, f(x), evaluated at x and may be represented as:

$$\nabla f(x) = \{\frac{\partial f(x)}{\partial x_1} \frac{\partial f(x)}{\partial x_2} ..... \frac{\partial f(x)}{\partial x_n}\}^T \tag{3.9}$$

The negative of the gradient points in the direction of greatest decrease, locally, in a function's value, the steepest descent, and is illustrated in Fig. 3.1. The gradient vector is used in the FR conjugate gradient algorithm exclusively in constructing the search directions. The directions are generated by the following equation where the first search direction is the steepest descent (DFP also uses the gradient, refer Eq. 3.7):

$$\{d^{k+1}\} = \{d^k\} \frac{\{g^{k+1}\}^T\{g^{k+1}\}}{\{g^k\}^T\{g^k\}} - \{g^{k+1}\} \tag{3.10}$$

The search directions defined by Eqs. 3.7 and 3.10 are illustrated in Fig. 3.2 for a two degree of freedom system defined by Eq. 3.1. The initial search direction, $\{d^0\}$, is the steepest descent. The next search direction, $\{d^1\}$, is taken such that it is A-conjugate (refer to Eq. 1.1). The minimum is located in two iterations; quadratic convergence for the two degree of freedom system is therefore satisfied. In this case, because the initial search direction was the steepest descent, the DFP and FR algorithms generated identical search directions. Figure 3.2 also illustrates the expected superiority of the DFP and FR algorithms, which use A-conjugate directions, over the method of using the steepest descent during each iteration. When the second search direction was the steepest descent, $-\{g^1\}$, it can be seen that it did not point to the global minimum as did the A-conjugate search direction computed by the DFP and FR algorithms.

The gradient of Eq. 3.1 may be expressed as follows:

$$\nabla U = [K]\{q\} - \{Q\} \tag{3.11}$$

When the initial estimate of the deformation vector is zero and the steepest descent is used for the first search direction, the gradient vector is the negative of the load vector and the initial search direction is defined by the load vector:

$$\{g^0\} = -\{Q\} \tag{3.12}$$

$$\{d^0\} = \{Q\} \tag{3.13}$$

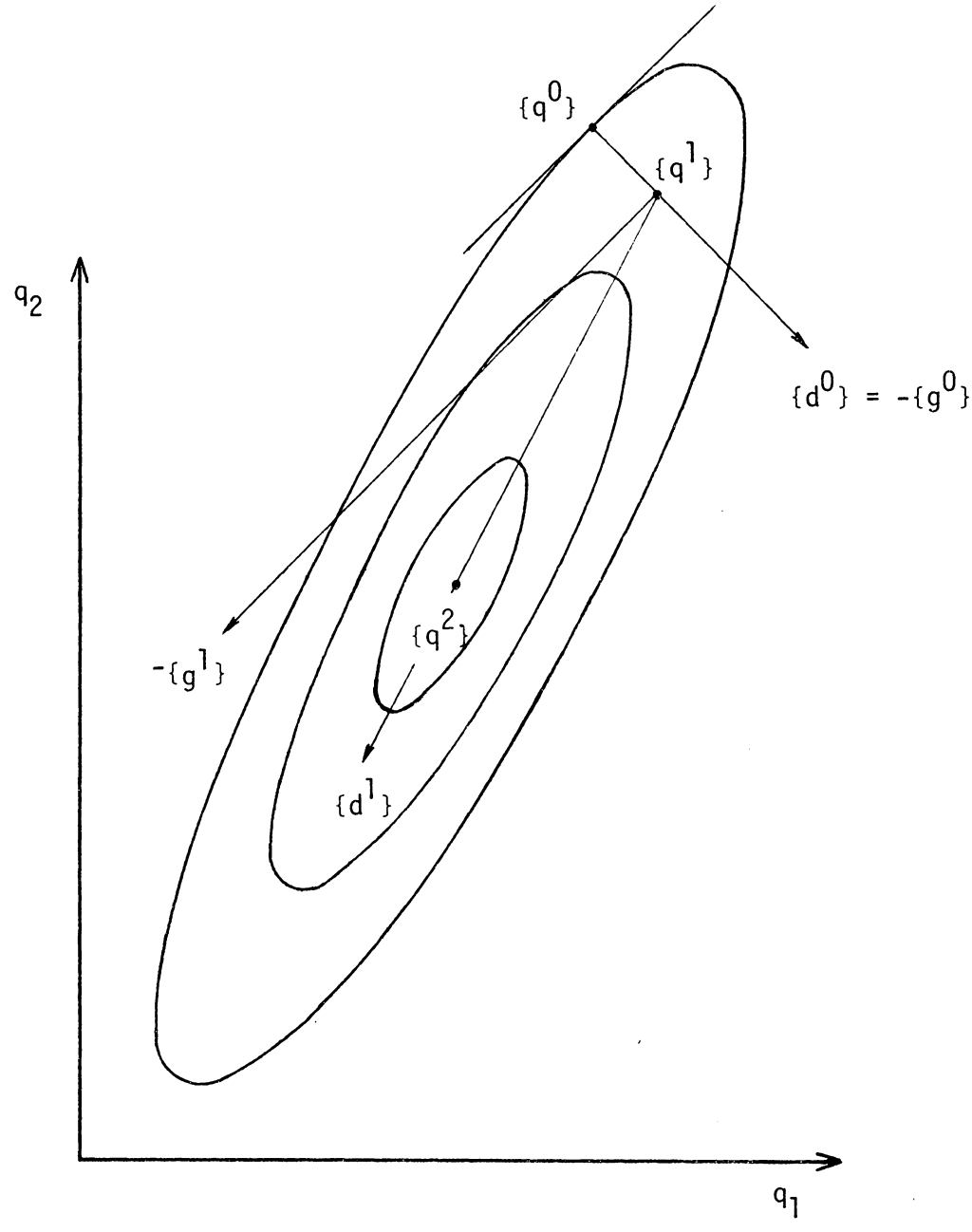



FIGURE 3.2 SEARCH DIRECTIONS

This applies to both the DFP and FR algorithms.

In structural analysis an equilibrium point can be defined as a point where the gradient vector is zero. A zero gradient vector may also be used as a termination criterion for the minimization algorithms. But because a zero gradient value may define an unstable equilibrium point additional convergence criteria should be considered for a general function. However in this study the function being minimized is strictly convex; therefore a zero gradient value guarantees a stable equilibrium point. When this convergence criterion is applied to Eq. 3.11 the equilibrium equations of the structure being considered are obtained:

$$[K]\{q\} = \{Q\} \tag{3.14}$$

This is the identical system of equations solved by the stiffness method which were arrived at by a completely different process (section 2.2).

By applying some of the previous conditions mentioned in this section it can be shown that in special cases where several load vectors are applied to the same structure Eq. 1.3 (which describes the linear minimization problem solved by the DFP algorithm during each iteration) reduces to Eq. 3.8 (a direct solution for the deformation vector by simple matrix multiplication). If the inverse hessian is accurately computed during the minimization of the total potential for the first load vector Eq. 1.3 reduces to Newton's method:

$$\{q^{k+1}\} = \{q^k\} - [K]^{-1}\{g^k\} \tag{3.15}$$

When Eq. 3.12 and the conditions that lead to it are applied to Eq. 3.15 Eq. 3.8 is arrived at:

$$\{q\} = [K]^{-1}\{Q\} \tag{3.16}$$

where

$$\{q\} = \{q^{k+1}\} \tag{3.17}$$

reducing an iterative technique, Eq. 1.3, to a simple matrix multiplication.

## 3.2 Variable Metric Algorithm

The Davidon-Fletcher-Powell variable metric algorithm is developed in this section. Several simplications to the general DFP algorithm are made possible because the total potential is a quadratic function and its form is known in matrix notation (Eq. 3.1). A flow chart of the basic algorithm used in this study is illustrated in Fig. 3.3. The algorithm is iterative and requires initial starting conditions. An initial estimate of the deformation vector, {q}, and the inverse hessian are required for the algorithm to begin the minimization process. The material and geometric properties of the structure to be analyzed and the load vector, {Q}, are required to define the total potential. Theoretically the load vector and initial estimate of the deformation vector do not influence convergence in linear problems [16]. Since a good estimate of the deformation vector would be difficult to obtain and any estimate is theoretically acceptable as equally good, the

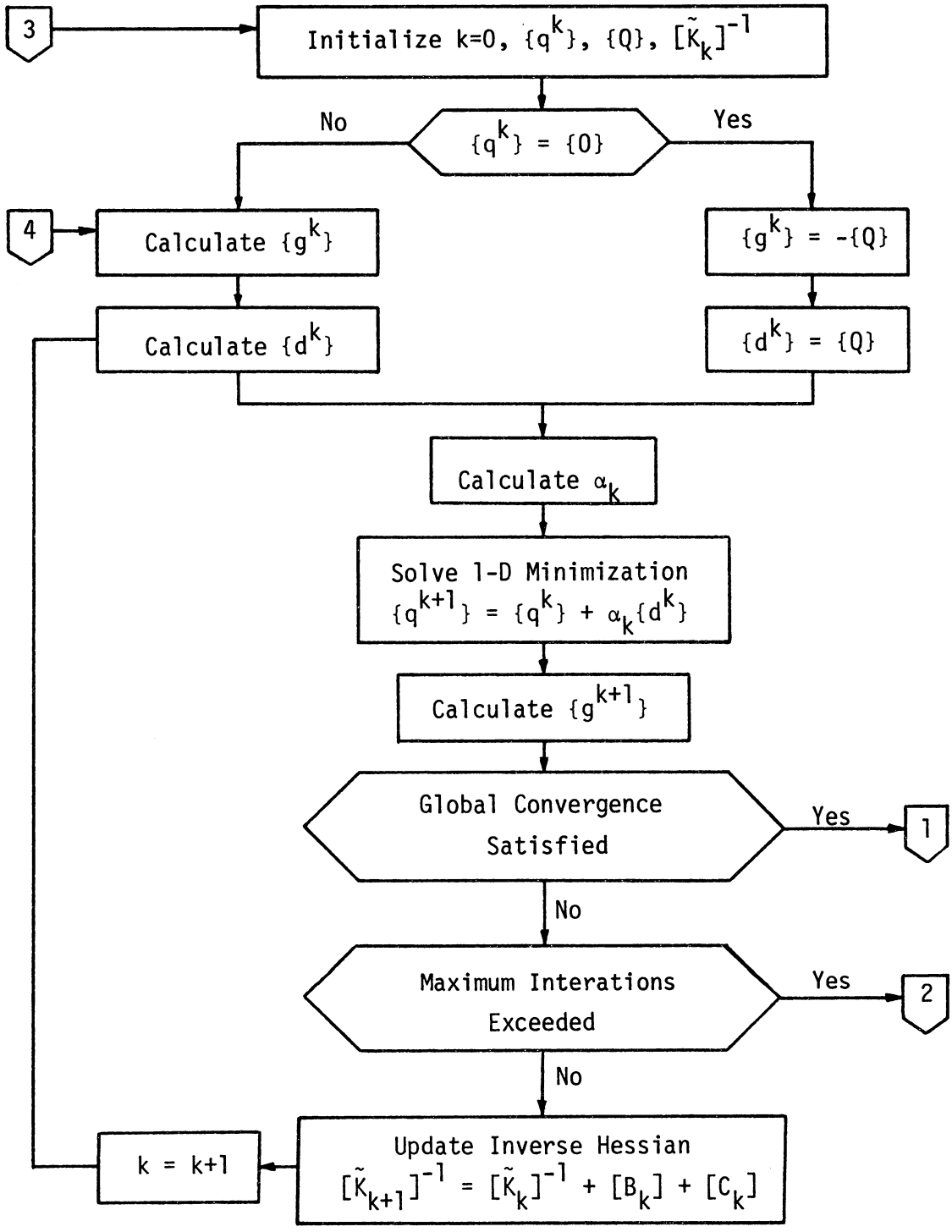



FIGURE 3.3 DFP ALGORITHM

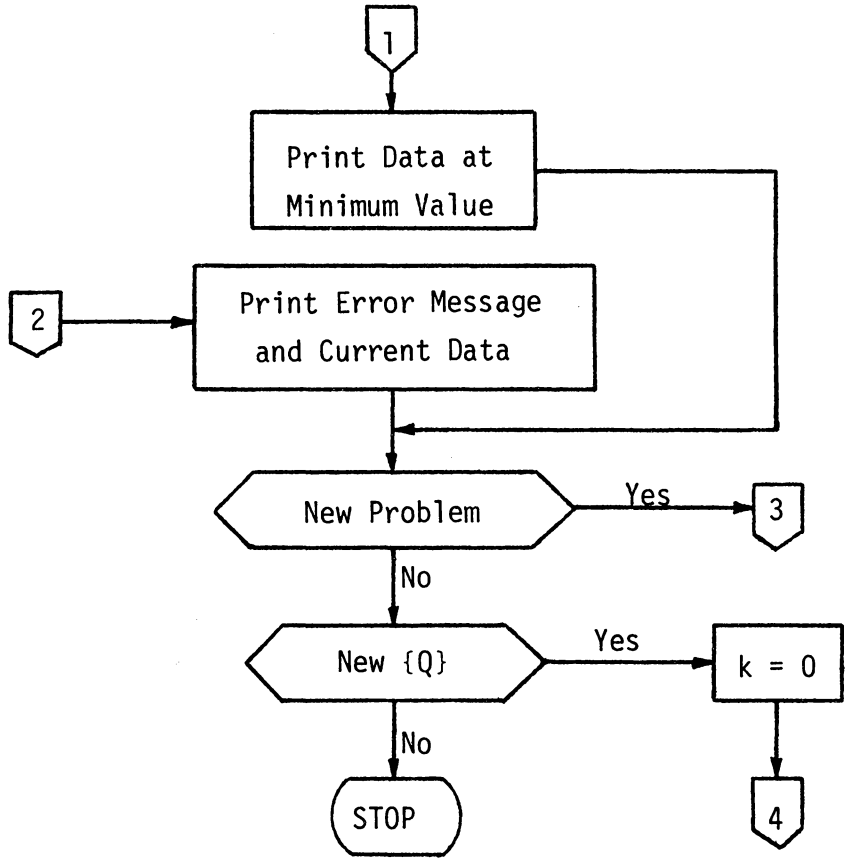



FIGURE 3.3 (continued)

initial estimate will be the zero vector. The estimate of the initial inverse hessian is of extreme importance. It is essential that this estimate be a symmetric positive definite matrix. The requirement that it be positive definite insures the stability of the algorithm by guaranteeing that the initial search direction will decrease the function value. If the initial estimate is positive definite, all updates will be positive definite. Each update is symmetric and made by the following relation:

$$[\tilde{K}_{k+1}]^{-1} = [\tilde{K}_k]^{-1} + [B_k] + [C_k] \tag{3.18}$$

where

$$[B_k] = \frac{\Delta q^k (\Delta q^k)^T}{(\Delta q^k)^T \Delta g^k} \tag{3.19}$$

$$[C_k] = - \frac{[\tilde{K}_k]^{-1} \Delta g^k ([\tilde{K}_k]^{-1} \Delta g_k)^T}{(\Delta g^k)^T [\tilde{K}_k]^{-1} \Delta g_k} \tag{3.20}$$

$$\Delta q^k = \{q^{k+1}\} - \{q^k\} \tag{3.21}$$

$$\Delta g^k = \{g^{k+1}\} - \{g^k\} \tag{3.22}$$

Equation 3.19 is intended to ensure $[\tilde{K}]^{-1}$ approaches $[K]^{-1}$ and Eq. 3.20 is intended to cancel out the initial estimate of the inverse hessian in the limit. For a quadratic function Eq. 3.19 will produce the true inverse hessian if n iterations are used (n is the number of unknowns), assuming no roundoff errors. In section 1.2c it was indicated that

the best initial estimate of the inverse hessian, $[\tilde{K}]^{-1}$, would be the true inverse hessian or some properly scaled matrix. But when no information is available to rationally make an estimate of $[\tilde{K}]^{-1}$ the identity matrix should be used. The identity matrix is symmetric and positive definite and will produce an initial search direction of the steepest descent. An improper estimate of $[\tilde{K}]^{-1}$ could destroy the convergence rate of the algorithm by possibly causing a search in a direction away from the minimum.

Once the initial values have been determined the gradient and search direction vectors discussed in the previous section are required. It is now necessary to solve the one dimensional minimization problem:

$$\{q^{k+1}\} = \{q^{k}\} + \alpha_k\{d^{k}\} \tag{3.23}$$

The scalar $\alpha_k$ for a quadratic function is arrived at by the minimization of $f(\{q^{k}\} + \alpha_k\{d^{k}\})$ with respect to $\alpha_k$:

$$\alpha_k = - \frac{\{g^{k}\}^T\{d^{k}\}}{\{d^{k}\}^T[K]\{d^{k}\}} \tag{3.24}$$

Although the hessian is required in Eq. 3.24, it does not need to be explicitly known due to the form of the strain energy. The strain energy is computed by Eq. 2.19 which does not require the stiffness matrix but is equivalent to the matrix form of Eq. 3.2. By multiplying Eq. 2.19 by a factor of two and using values from the direction vector instead of the deformation vector the scalar value in the denominator of Eq. 3.24 is computed without using matrix operations. The use of

Eq. 3.24 greatly simplifies the operation of the DFP algorithm. The line search which has been repeatedly mentioned in the literature as a major expenditure of time and source of error has now been replaced by a single "exact" step.

After the one dimensional minimization problem has been solved it is necessary to check the solution to see if it sufficiently locates the minimum of the total potential. Although quadratic convergence is guaranteed theoretically, it may not be realized due to roundoff error. It is also possible the minimum of a quadratic function may be reached in less than n iterations. Quadratic convergence can only be used as an estimate of the maximum number of iterations probably required to locate the minimum. The value of the gradient vector will be used as a convergence criterion. When it is equal to zero it defines the minimum of the total potential and therefore satisfies the equilibrium equations of the structure being analyzed. Due to roundoff errors it is unlikely a zero gradient vector will ever be found. Therefore a sufficiently small value, $\varepsilon_1$, is chosen such that if the absolute value of all elements of the gradient vector is less than $\varepsilon_1$ the gradient vector will be considered close enough to zero to define the minimum. This will be known as property one and represented by:

$$|g_i^k| < \varepsilon_1 \qquad i = 1,\ldots,n \tag{3.25}$$

Since property one is an approximation and dependent upon the selection of $\varepsilon_1$ another convergence criterion, property two, must also be satisfied.

Property two requires the distance to the next minimum be less than some predetermined value, $\varepsilon_2$. This is represented by the following equation:

$$|\alpha_{k+1}|(\{d^k\}^T\{d^k\})^{\frac{1}{2}} < \varepsilon_2 \tag{3.26}$$

The left hand side of Eq. 3.26 is the distance to the next minimum that would be located during the one dimensional minimization if the process were continued. The new search direction used in this convergence test is generated prior to updating the inverse hessian, therefore the distance to the next minimum is approximated. Because this value approximates the distance to the next minimum, it represents a maximum change that can be expected to occur in any member of the present deformation vector. This allows the left hand side of Eq. 3.26 to be interpreted as a measure of the movement of the calculated minimum when a new search direction is generated.

## 3.3 Conjugate Gradient Algorithm

The Fletcher-Reeves conjugate gradient algorithm is discussed in this section. The simplications used in developing the DFP algorithm in section 3.2 are applicable to the FR algorithm. A flow chart of the basic FR algorithm used in this study is illustrated in Fig. 3.4. The FR algorithm is iterative and has the identical requirements for the initial starting conditions as the DFP algorithm with the exception that the inverse hessian is not required. Other than the difference in the methods of generating search directions, Eqs. 3.7 and 3.10, the FR algorithm requires the same steps as the DFP algorithm in mini-

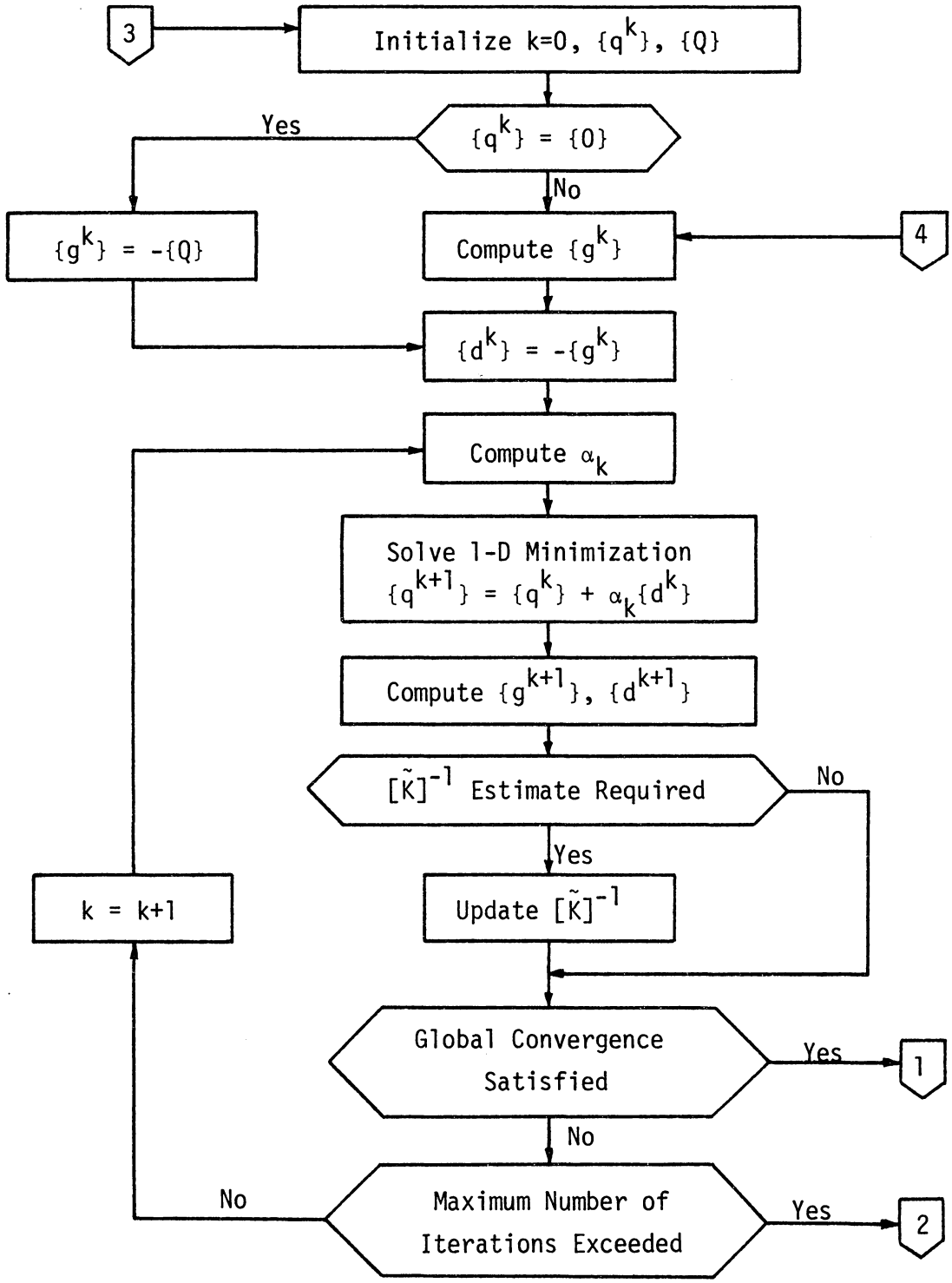



FIGURE 3.4 FR ALGORITHM

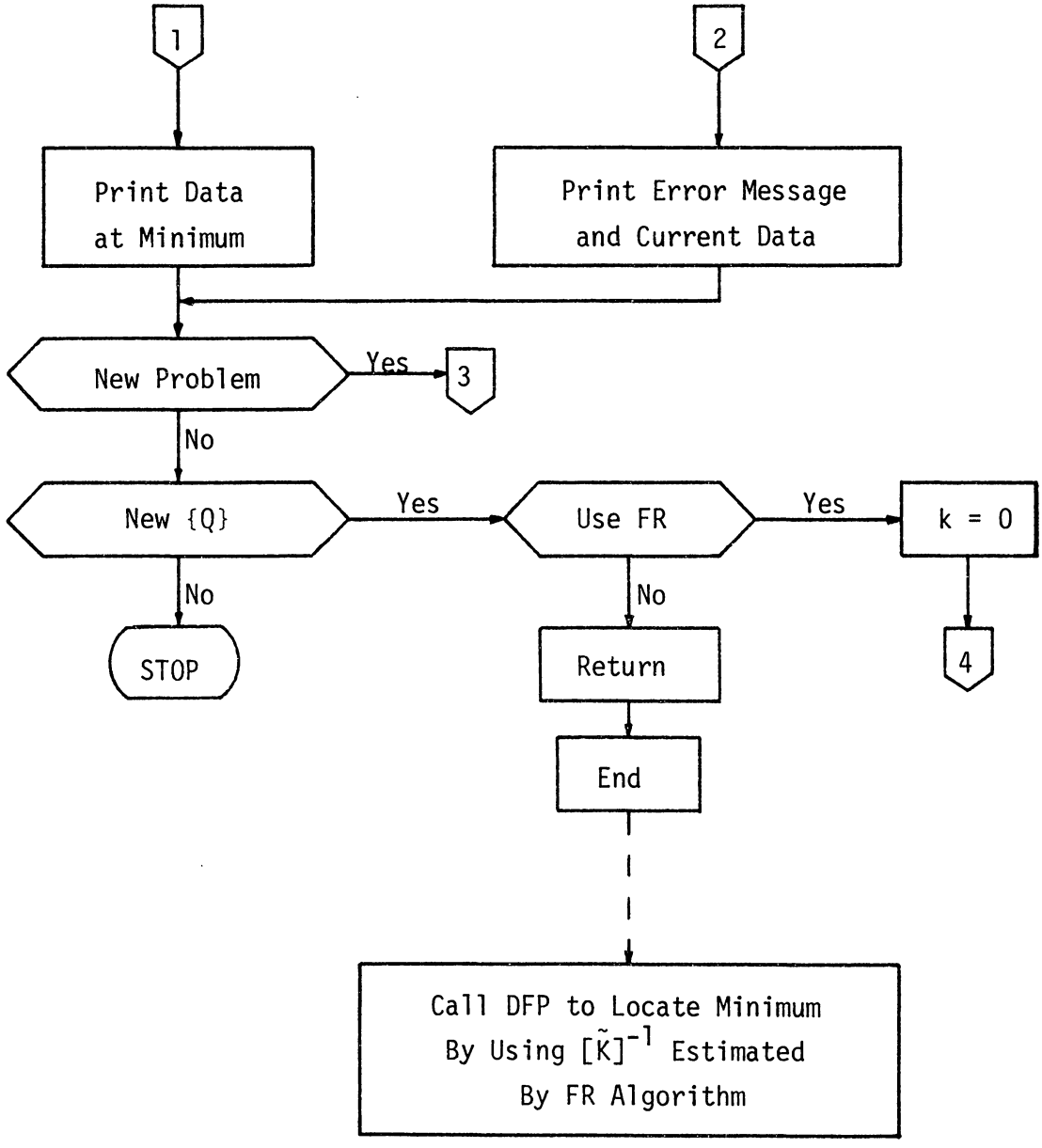



FIGURE 3.4 (continued)

mizing the total potential. The one dimensional minimization problem is solved identically by both methods and the same convergence criteria must be satisfied by both methods.

The significance of this algorithm is its modest storage and ability to estimate the inverse hessian. During the minimization of the total potential, an option is available to use Eq. 3.19 to construct the inverse hessian of the total potential. Although this increases the storage requirements of the FR algorithm, the estimate of the inverse hessian is not required in any operations. The economy of only using column vectors in the minimization process is still retained by the FR algorithm. The use of Eq. 3.19 allows information generated by the FR algorithm to be retained and used to improve the convergence rate of the DFP algorithm when new load vectors are considered for the same structure.

## 3.4 Gaussian Elimination

This section describes the method of solution, Gaussian elimination, of the equilibrium equations generated by the stiffness method. Symmetry and bandedness of the stiffness matrix (matrix of coefficients) is used to reduce storage requirements in computer application. The Gaussian elimination algorithm used in this study is taken from Desai and Able [9].

Gaussian elimination is a direct method of solving a system of linear equations. Unlike the iterative algorithms discussed in sections 3.2 and 3.3 no convergence criteria must be satisfied. The number of operations required to solve a system of equations

and therefore satisfy equilibrium is fixed and determined by the matrix of coefficients (in some cases the accuracy of the elimination may need to be improved by iteration on the residuals [9], but is not necessary in this study). The equilibrium equations to be solved are:

$$[K]\{q\} = \{Q\} \tag{3.27}$$

Gaussian elimination transforms the coefficient matrix, [K], into a unit upper triangular matrix by forward reduction. During this process the load vector, {Q}, is also modified. This allows for the solution of the deformation vector directly by back-substitution. When several load vectors are to be considered for the same structure and therefore same [K] it is desirable to separate the reduction of the stiffness matrix and load vector. Forward reduction of the stiffness matrix is only required once, while forward reduction of the load vector and back-substitution are done as many times as there are load vectors [7].

The algorithm used in this study separates the reduction of the stiffness matrix and the load vector. The following symmetric decomposition of the stiffness matrix is used [12]:

$$[K] = [L][D][L]^T \tag{3.28}$$

where

[L] = unit lower triangular matrix

[D] = diagonal matrix

Only the product $[D][L]^T$ in Eq. 3.28 must be stored. Due to the symmetry and banded form of the stiffness matrix storage requirements

are reduced from an (n x n) matrix to that of an (n x IBAND) matrix, where IBAND is the semiband width of the stiffness matrix. The product $[D][L]^T$ is required for the reduction of the load vector. The matrix $[L]^T$ corresponds to the reduced stiffness matrix and the matrix [D] contains terms used in the reduction of the stiffness matrix that are required to reduce the load vector. The product $[D][L]^T$ is stored by overwriting terms of the stiffness matrix. Once the load vector has been reduced, the deformation vector can be solved for directly by back substitution using $[L]^T$. By inferring the diagonal elements are unity $[D][L]^T$ becomes $[L]^T$. This process is continued for as many load vectors as desired.

# Chapter 4

# RESULTS

This chapter contains the results obtained from the analysis of several plane frames by both the stiffness method and energy method. Both the Fletcher-Reeves conjugate gradient algorithm and the Davidon-Fletcher-Powell variable metric algorithm are used to minimize the total potential. Also included is the analysis of a frame subject to multiple load conditions.

## 4.1 Example Problems

The frames listed in this section were chosen to demonstrate the effect of the number of degrees of freedom, d.o.f., as well as the configuration of a frame on the two methods of analysis. Figures 4.1 through 4.7 illustrate the frames used in this study. They range from 6 to 63 d.o.f. The members in all frames generally decrease in stiffness as the height of the frame is increased. For example, the members in the bottom row of frame 18A, Fig. 4.2, will not be as stiff as the respective members in the bottom row of frame 18B, Fig. 4.3, while the respective members in the top row of both frames will have the same stiffness. Frames 18A and 18B have the same d.o.f. and frames 36A and 36B have the same d.o.f. Frames 60A and 63B differ in d.o.f. by three. These pairs of frames have been selected to demonstrate, in some methods of analysis, that configuration and member properties are more significant than d.o.f. in determining solution time. The frames which have the same number of bays in both the


FIGURE 4.1 FRAME 6A

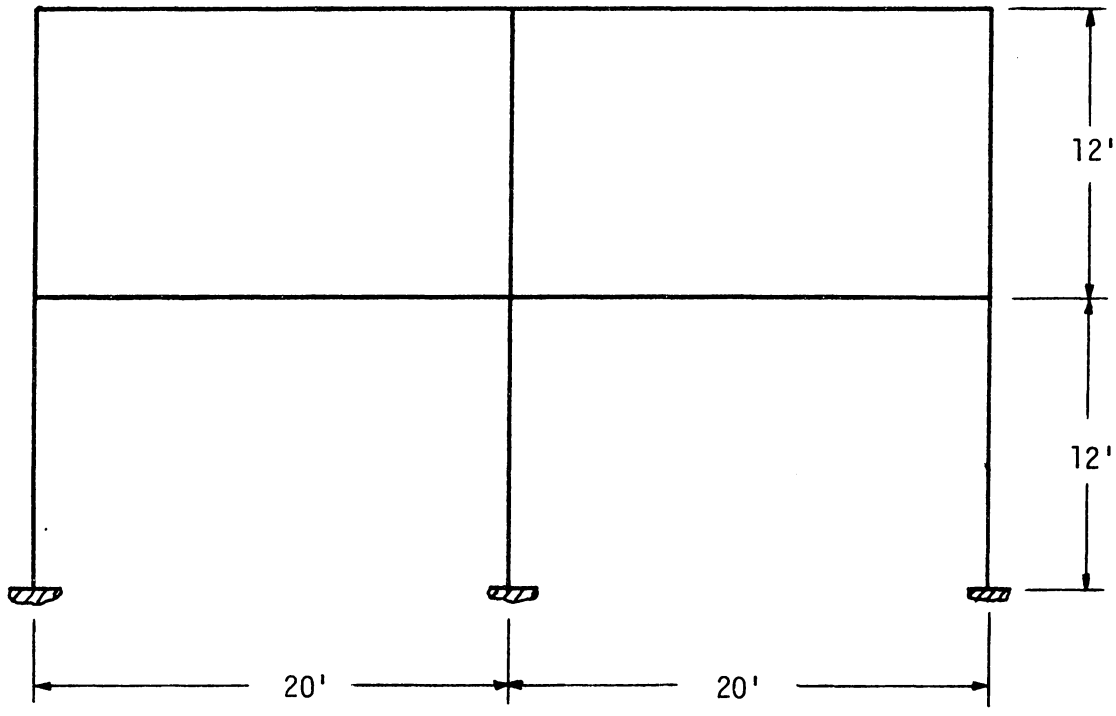



FIGURE 4.2 FRAME 18A

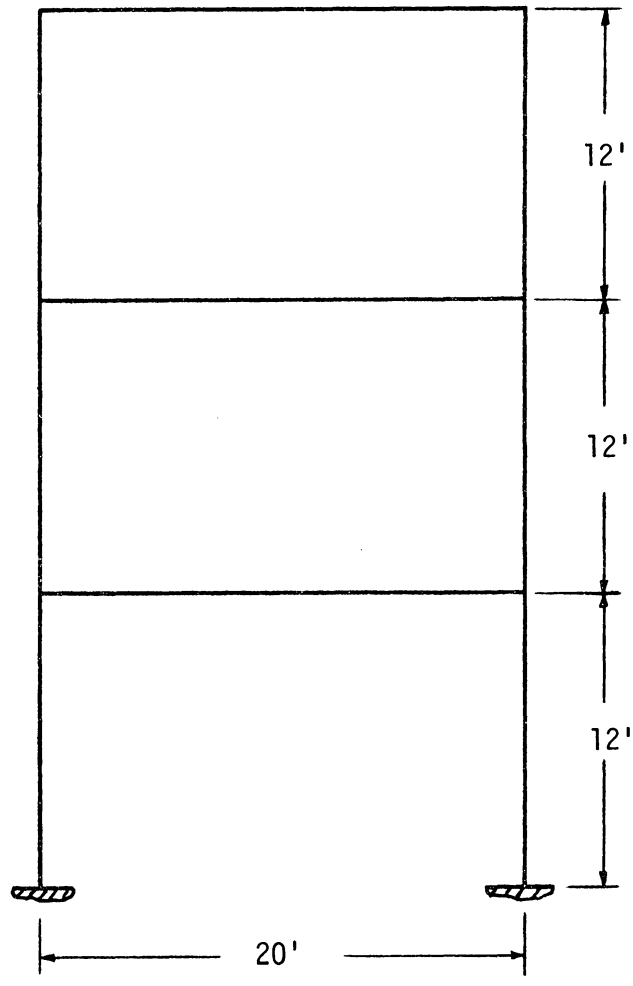



FIGURE 4.3 FRAME 18B

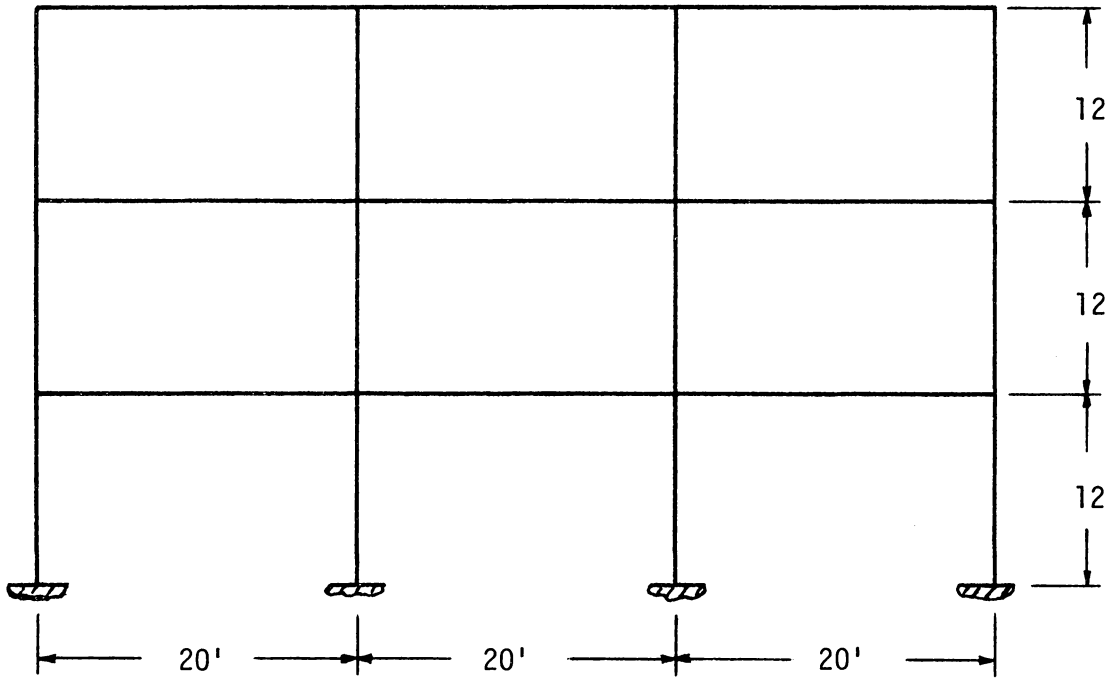



FIGURE 4.4 FRAME 36A

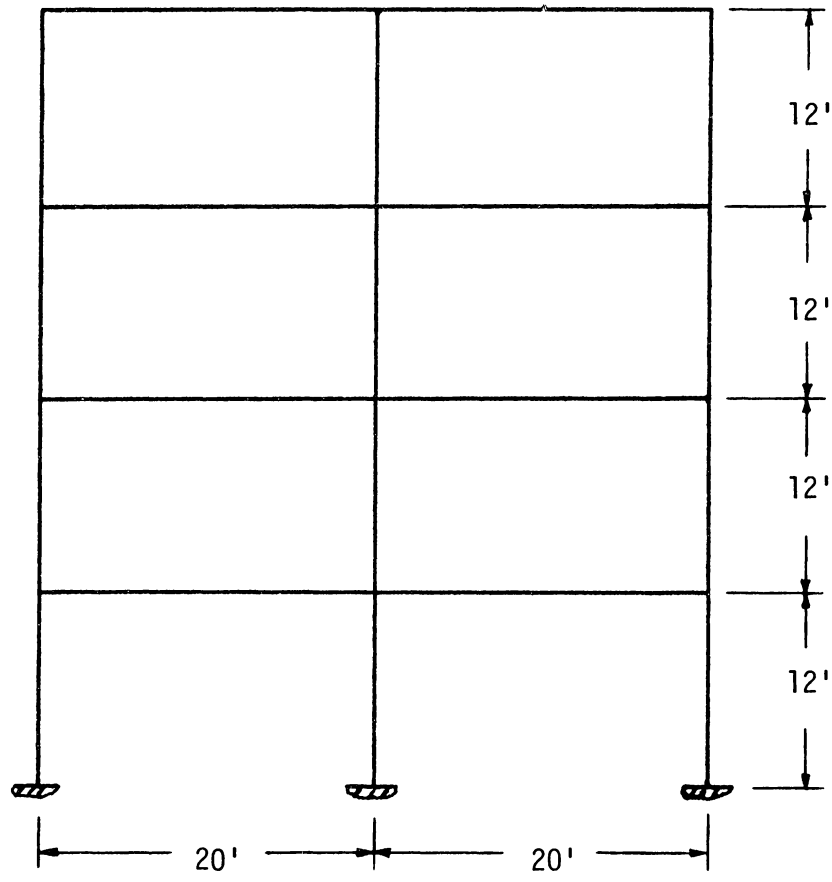



FIGURE 4.5 FRAME 36B

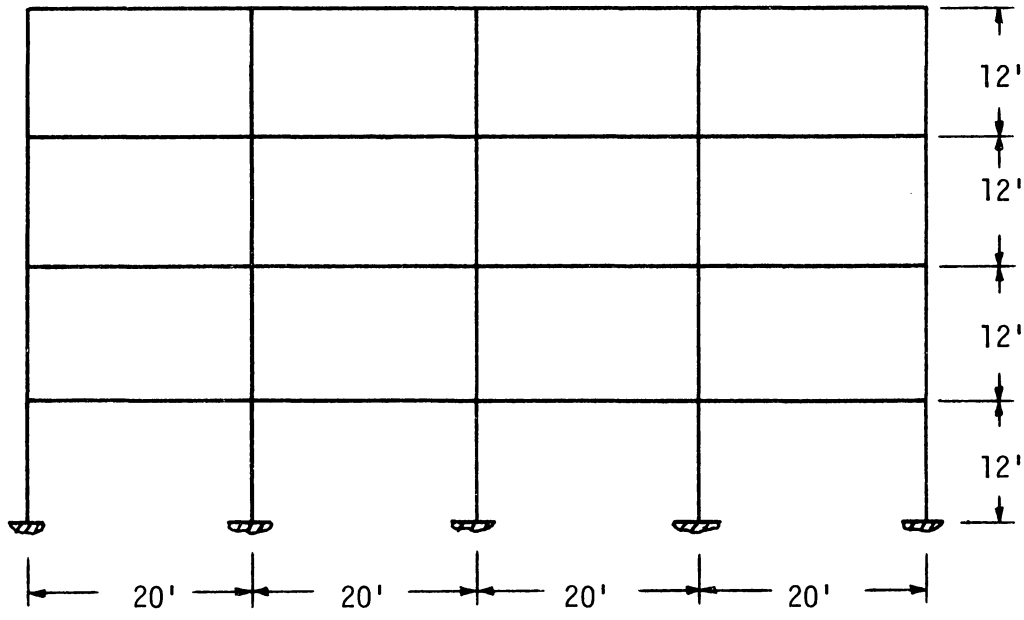



FIGURE 4.6 FRAME 60A

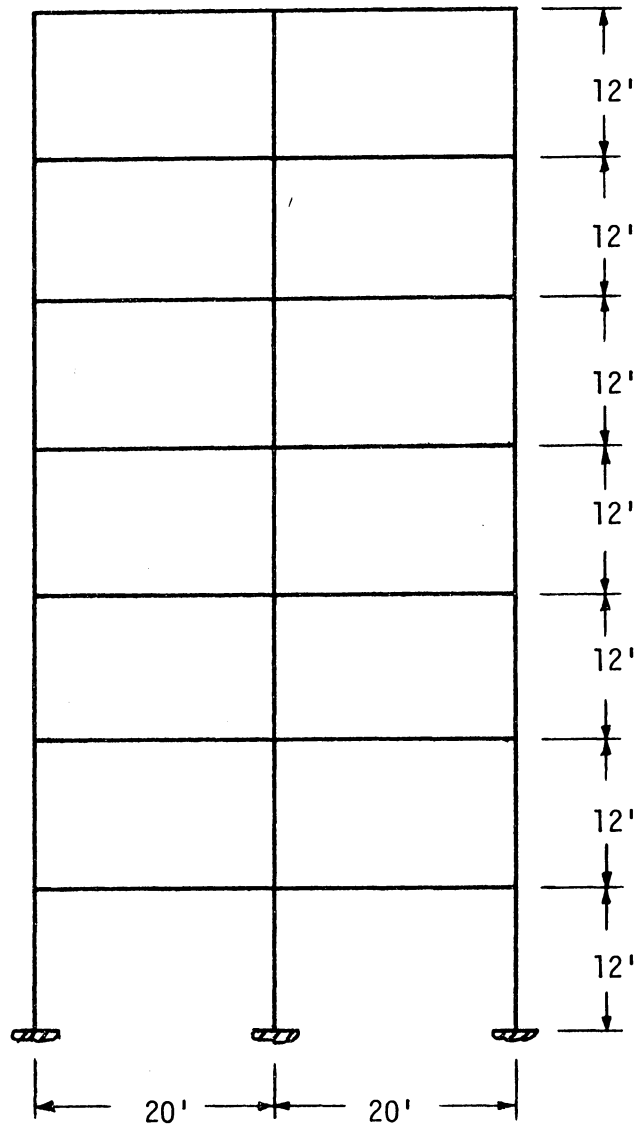



FIGURE 4.7 FRAME 63B

horizontal and vertical directions will be known as class A frames. Those with more bays in the vertical than horizontal direction will be known as class B frames. Frame 6A will be considered the first member in both classes, since both classes are constructed from this basic frame.

Except where noted, all data has been generated by the following type of load. Horizontal joint forces are applied on one side and uniformly decrease in value with story height. Vertical joint loads, all having the same value, are applied along the top row of joints. All other joint forces are zero.

All mathematical operations were done in double precision arithmetic. Numerical results from both methods were identical up to the eight significant digits examined. Execution times reported include all computations required for analysis.

## 4.2 Results From a Single Load Condition

### a. Results From the Energy Method

The frames in Figs. 4.1 through 4.7 have all been analyzed for a single load condition. The energy method, ENERGY, was used and the total potential was minimized by both the Fletcher-Reeves conjugate gradient algorithm, FR, and the Davidon-Fletcher-Powell variable metric algorithm, DFP. The results in terms of execution time for both methods are illustrated in Fig. 4.8.

For a small d.o.f. (10 or less d.o.f.) both the FR and DFP algorithms perform almost identically. As the d.o.f. increases, the amount of execution time increases exponentially, with DFP

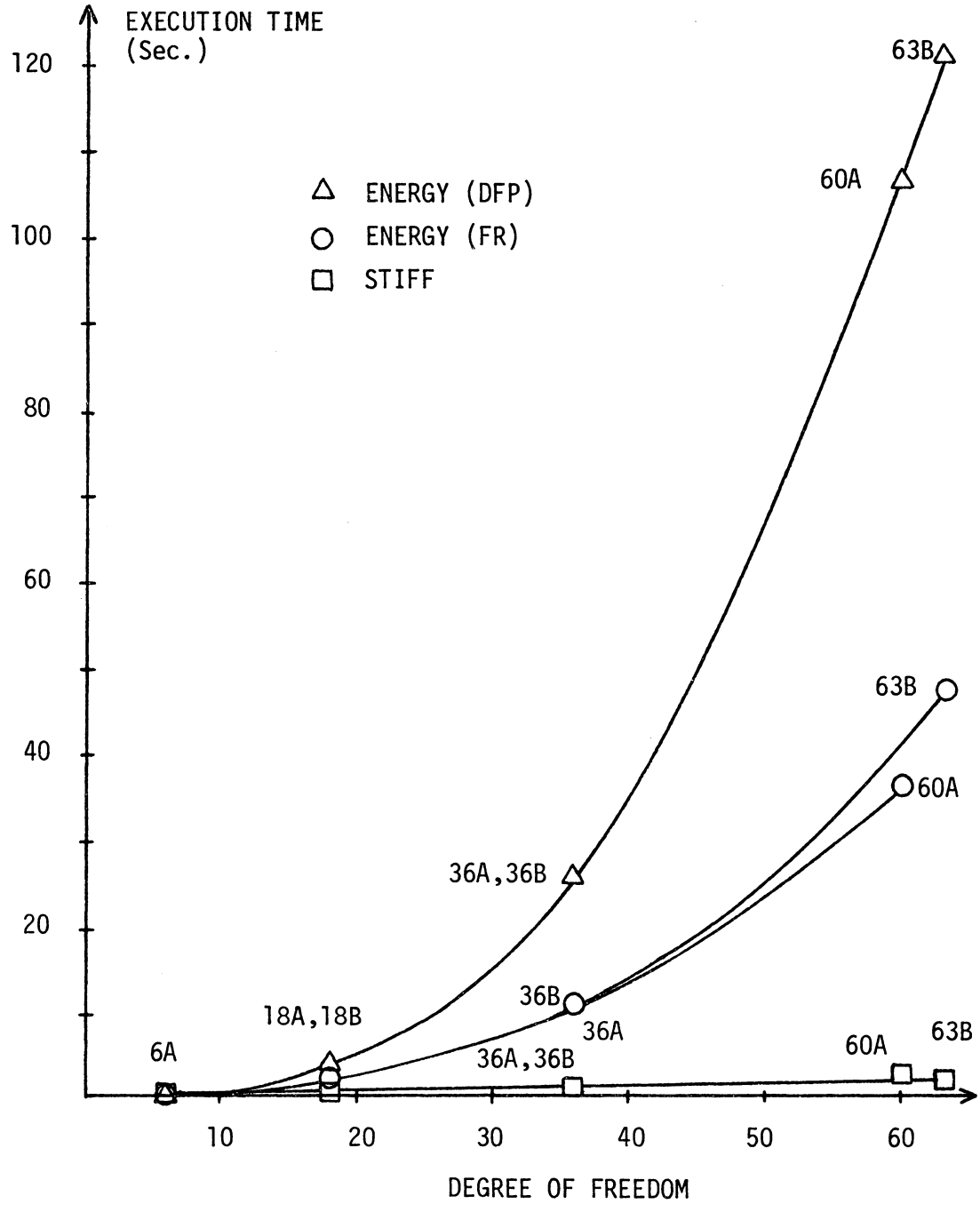



FIGURE 4.8 EXECUTION TIME vs. D.O.F. FOR ENERGY AND STIFF

increasing much more rapidly than FR. The increase when DFP was used produced a smooth curve connecting all points generated by both class A and class B frames. The increase in execution time when FR was used also increased exponentially, but along two different paths, one defined by class A frames and one by class B frames. The reason for the two different paths, shown in Fig. 4.8, is a result of the number of iterations required to minimize the total potential of each frame by FR. Figure 4.9 illustrates that when FR is used to minimize the total potential, the number of iterations required to locate the minimum is always much greater than the quadratic convergence rate required by theory. It also shows more iterations are required to minimize class B frames than class A frames (the number of iterations has a direct effect on execution time). This is influenced by the scale of the total potential function. Due to the wider range in stiffness of the members of class B than of class A the elements of the stiffness matrix, for example along the diagonal, for the same d.o.f. frame will have a larger difference in values for class B frames. This will produce a higher condition number for the stiffness matrix of class B frames than class A frames. This results in class A frames being "scaled better naturally" than class B frames for the same d.o.f. (refer sec. 1.2c). Figure 4.9 indicates that the scale of the total potential has a more dominant effect on the computational effort required to minimize the total potential by FR than the d.o.f. This makes predictions of the behavior of FR for minimizing the total potential uncertain, with Figs. 4.8 and 4.9 being only a rough guide.

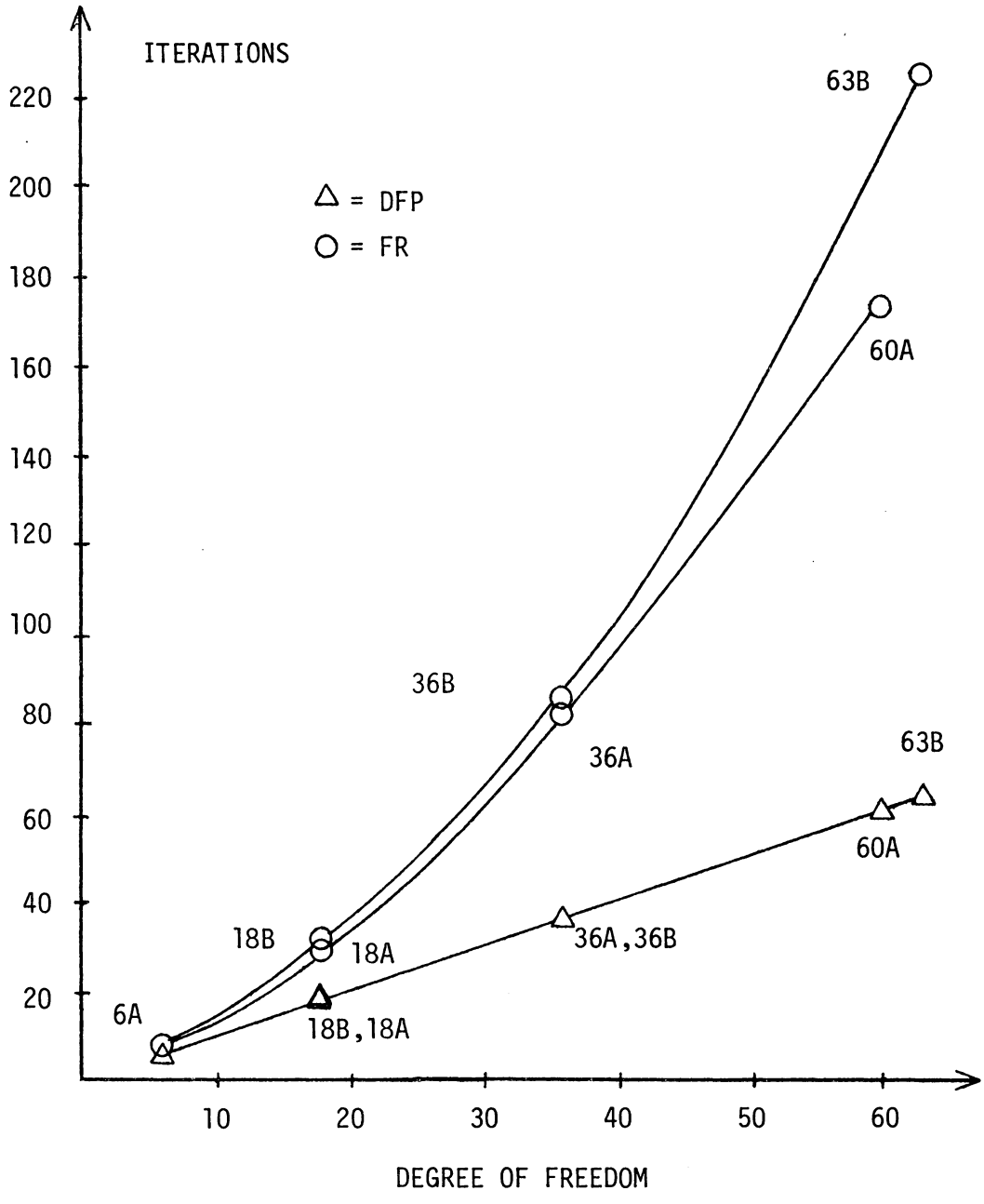



FIGURE 4.9 ITERATIONS PER D.O.F. REQUIRED BY FR AND DFP

The DFP algorithm appears insensitive to the scale problems experienced by FR, based on number of maximum iterations required by theory to minimize the total potential. Figure 4.9 illustrates this by the linear relation between iterations and d.o.f. All problems took NDF+1 iterations except frame 18A, which required only NDF iterations. Theory states NDF is the maximum number of iterations required to minimize a quadratic function of NDF variables, and NDF updates produce the true inverse hessian. In this study, NDF+1 iterations correspond to NDF updates of the inverse hessian. The exponential increase in execution time, Fig. 4.8, with increase in d.o.f. is expected. Figure 4.10 shows the relation between the main user of storage, the approximate inverse hessian, $[\tilde{K}]^{-1}$, and the d.o.f. The increase in execution time can be expected to increase more rapidly than the increase in $[\tilde{K}]^{-1}$ storage. An increase in d.o.f. exponentially increases $[\tilde{K}]^{-1}$ storage requirements and also increases the expected number of times $[\tilde{K}]^{-1}$ must be manipulated. Results from both Figs. 4.9 and 4.10 indicate the behavior of the DFP algorithm, with increase in d.o.f., reliable for the type of problems in this study.

b. Results From the Stiffness Method

The frames, with the same load conditions, used in section 4.2a have also been analyzed by the stiffness method, STIFF. Since STIFF uses a direct solution technique, there is a fixed number of operations required to analyze each structure. The number of operations is directly related to the number of elements of the

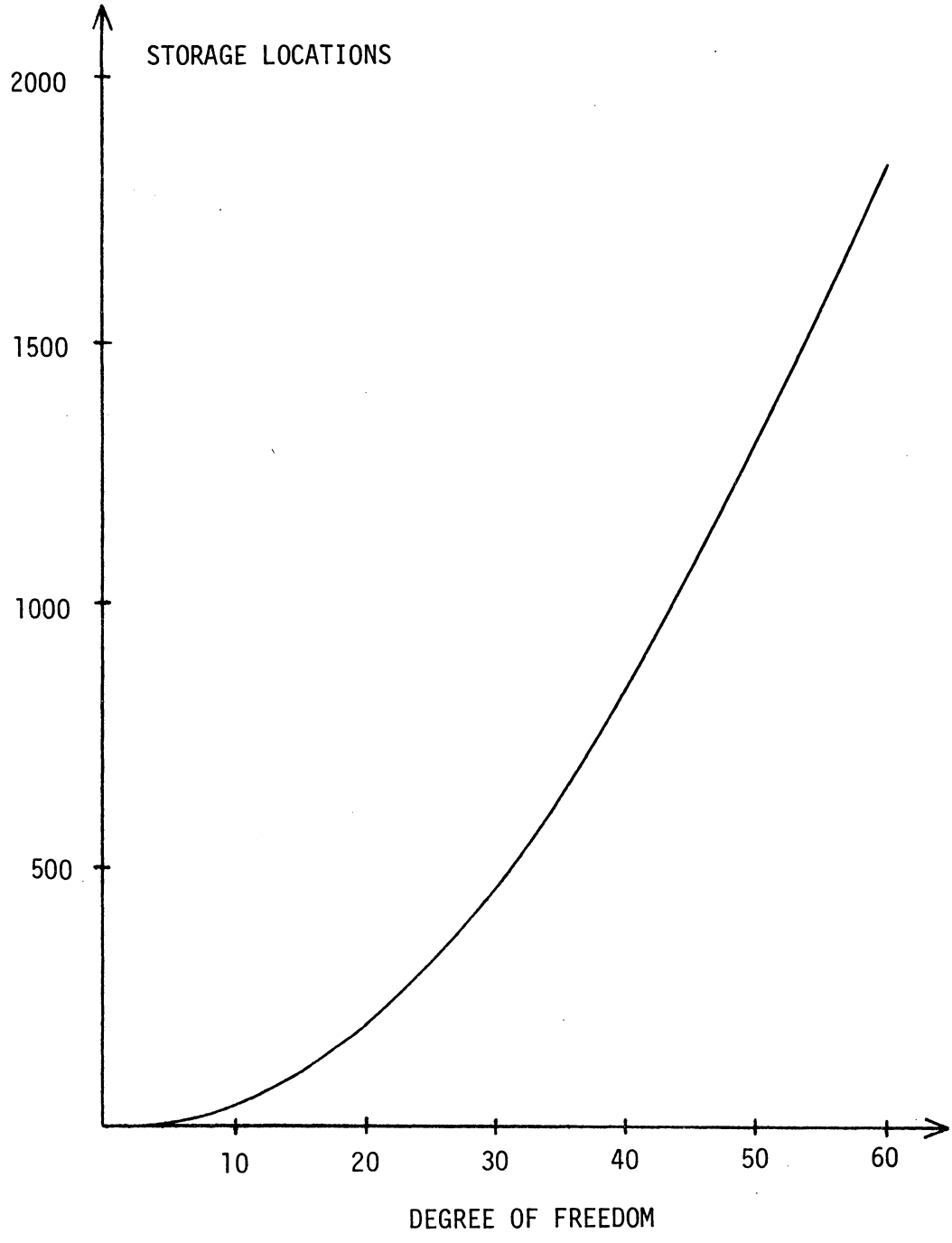



FIGURE 4.10 INCREASE IN COMPUTER STORAGE FOR $[\tilde{K}]^{-1}$ BASED ON D.O.F.

stiffness matrix that must be stored. This value is affected by configuration as well as d.o.f. It is determined by the numbering scheme used to identify the joints of the structure. The numbering scheme determines the semiband width and, therefore, required number of storage locations. The joints of all frames in this study have been optimally numbered. The difference in storage for the same d.o.f., but different configuration, is shown in Table 4.1. Table 4.1 also shows the excessive storage requirements of the inverse hessian used by ENERGY compared to STIFF requirements.

Figure 4.11 shows that the relation between execution time and storage requirement is closely linear. The execution time for class B frames with the same or nearly the same d.o.f. as class A frames shows they were analyzed more quickly as a result of smaller storage demands. A direct comparison between STIFF and ENERGY for each frame analyzed is shown in Fig. 4.8.

## 4.3 Results of Multiple Load Conditions

This section compares the rate of increase in execution time required by STIFF and ENERGY when several load conditions are applied to the same structure. The additional load vectors are increments of the original load vector except where noted differently. Only the DFP algorithm is used in minimizing the total potential. The FR algorithm retains no information that would improve the solution process when several load conditions are given. The property of using FR to estimate the inverse hessian (sec. 3.3) is lost due to the excessive

TABLE 4.1

MAJOR COMPUTER STORAGE REQUIREMENTS OF
ENERGY (DFP) AND STIFF

| FRAME | D.O.F. | $[K]^{-1}$ STORAGE | [K] STORAGE |
|---|---|---|---|
| 6A | 6 | 21 | 36 |
| 18A | 18 | 171 | 216 |
| 18B | 18 | 171 | 162 |
| 36A | 36 | 666 | 540 |
| 36B | 36 | 666 | 432 |
| 60A | 60 | 1830 | 1080 |
| 63B | 63 | 2016 | 756 |

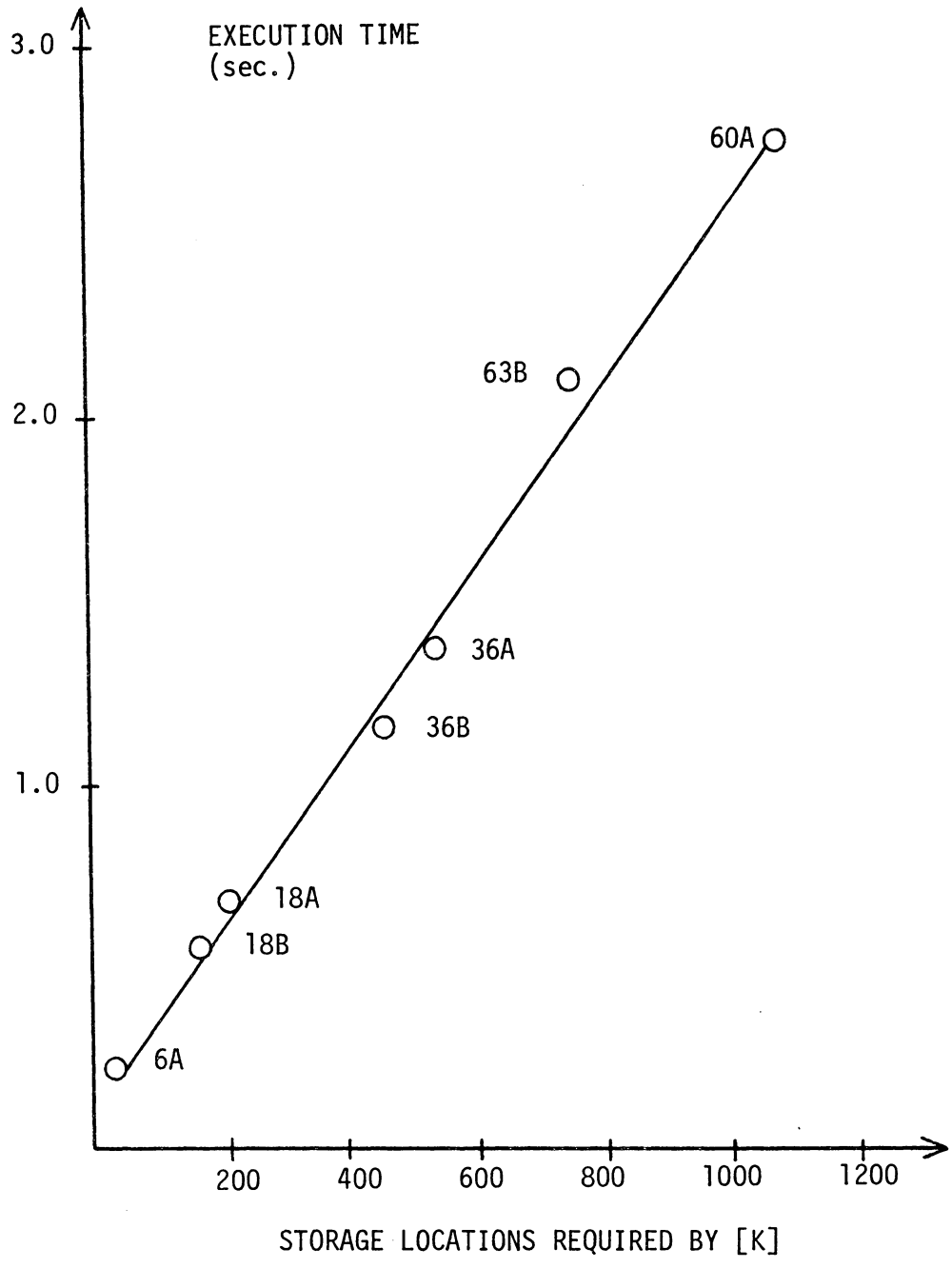



FIGURE 4.11 EFFECT OF PROBLEM SIZE ON STIFF EXECUTION TIME

number of iterations required to minimize the total potential (refer Fig. 4.9). Frame 18B was chosen for the comparison. From the previous solution, ENERGY (DFP) retains the estimate of the inverse hessian and uses the previously found minimum as the initial starting point, while STIFF retains the decomposed stiffness matrix computed during the first solution process and uses it when additional load vectors are applied.

The results, in terms of execution time versus the number of load increments, of applying load increments to frame 18B by both ENERGY (DFP) and STIFF are shown in Fig. 4.12. The increase in execution time when additional load vectors are applied using STIFF is linear. Figure 4.12 shows that requiring computation and decomposition of the stiffness matrix only once significantly reduces the time required for analysis using additional load vectors. The rate of increase in execution time, Fig. 4.12, using ENERGY (DFP) is greatly reduced after the solution using the initial load vector and even further reduced after the first load increment. After the first load increment there is a linear increase in execution time. The reason the increase in execution time for ENERGY (DFP) is not continuously linear after the initial load vector is demonstrated by Fig. 4.13a. Figure 4.13a shows the number of iterations required to minimize the total potential for each load vector. The initial load vector took 18 iterations and required 17 updates of the inverse hessian. The first load increment took 3 iterations, requiring two updates of the inverse hessian. After the first load increment,

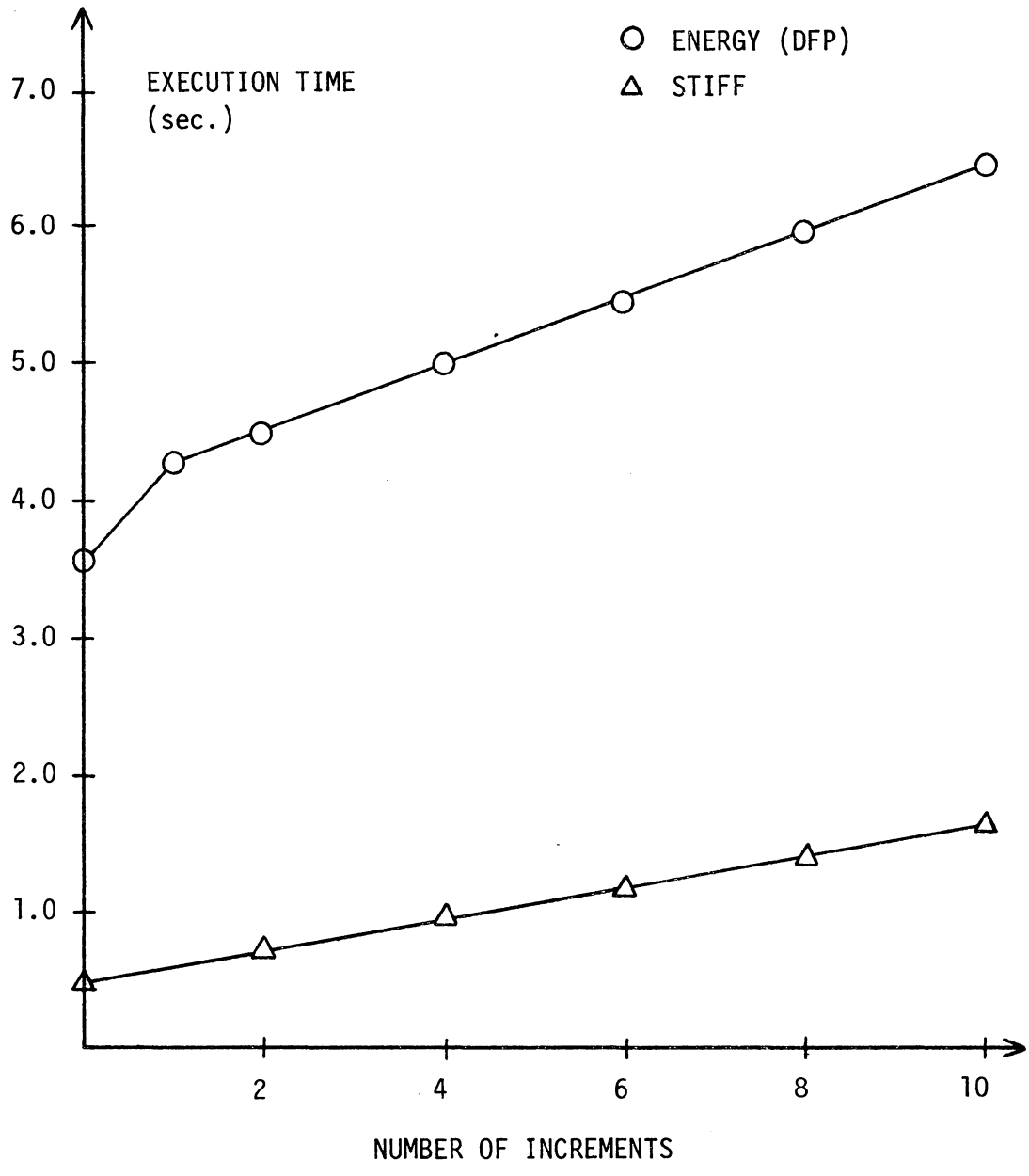
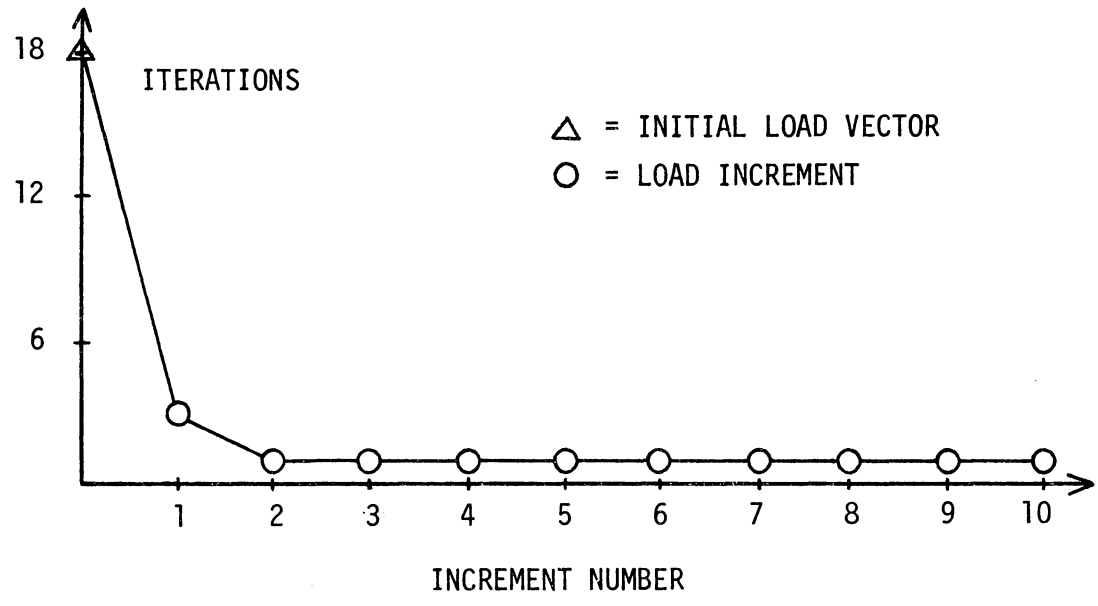
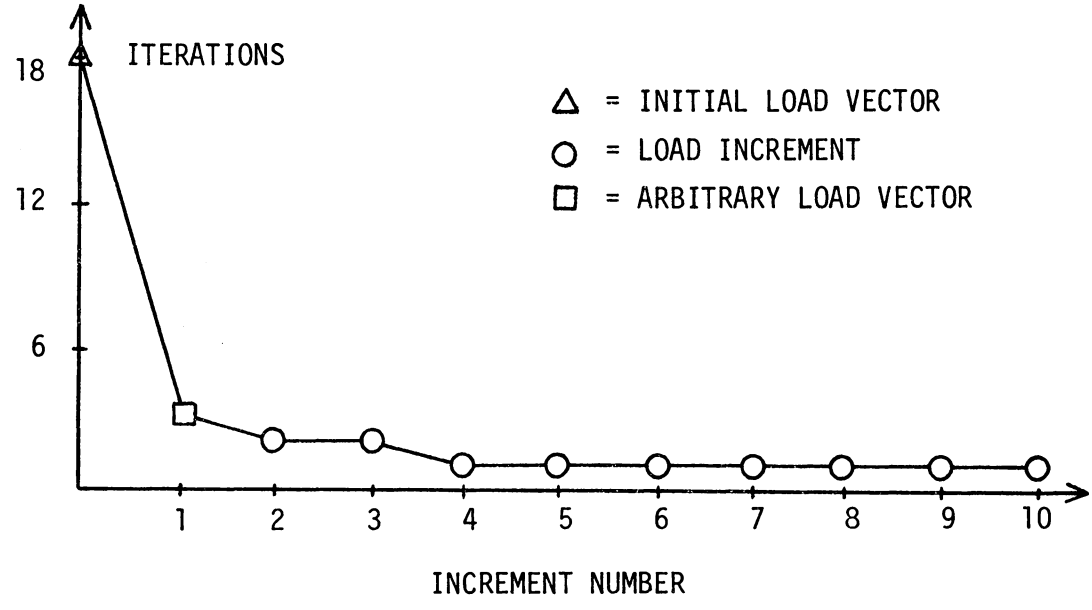



FIGURE 4.12 EFFECT OF LOAD INCREMENTS ON ENERGY (DFP) AND STIFF

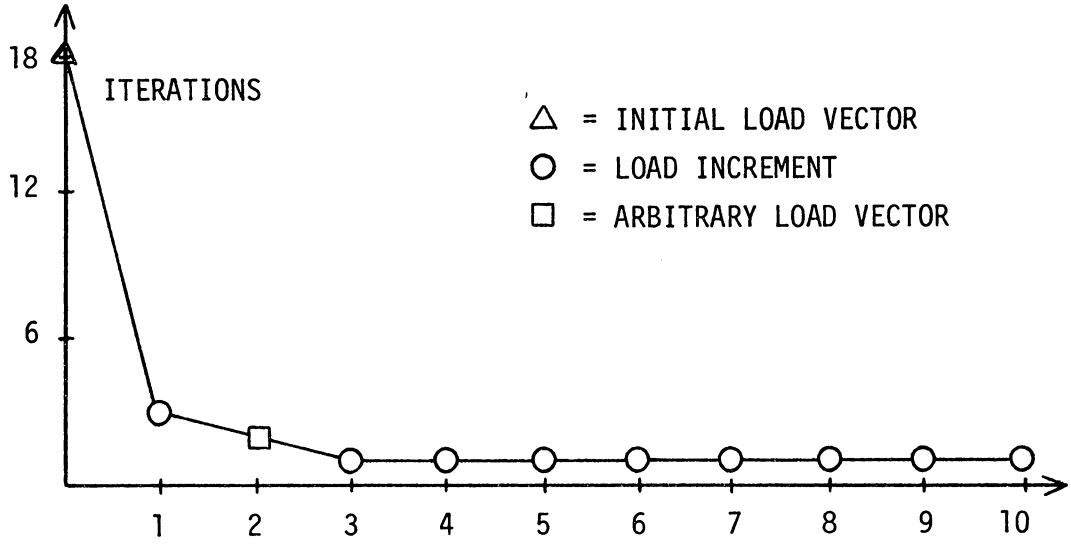



(a)

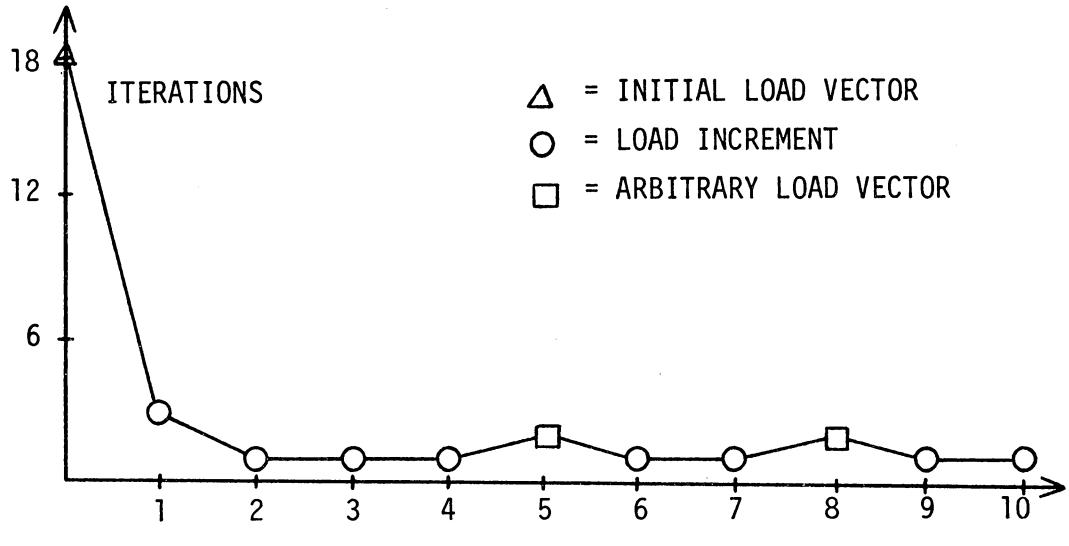



(b)

FIGURE 4.13 EFFECT OF ADDITIONAL LOAD VECTORS ON NUMBER OF ITERATIONS REQUIRED BY ENERGY (DFP)




(c)




(d)

FIGURE 4.13 (continued)

only 1 iteration was required to minimize the total potential and, therefore, no updates of the inverse hessian were required. This suggests the inverse hessian was not sufficiently well defined until after the first load increment. The number of iterations during each minimization gives an indication of the required execution time during that solution process. Figure 4.14 is a plot of the cumulative number of iterations during each load increment. The form of this graph indicates the trend in increase of execution time based on iterations required (refer Fig. 4.12 for DFP).

Since there is no guarantee the inverse hessian is sufficiently well defined, a convergence test must be made after each one dimensional minimization problem is solved. Because of this a direct solution by Eq. 3.16 can not be used. Using Eq. 3.16 would give the same results after increment number one as minimizing the function by the DFP algorithm, but there is no way of knowing this a priori. Therefore, after increment number one, using DFP results in many unnecessary operations (convergence tests) which could be avoided if the inverse hessian were known to be satisfactorily defined. Figures 4.13b through 4.13d show the effect on the number of iterations and the requirement that the inverse hessian may have to be updated at any time when an arbitrary load vector replaces an incremental load vector. This further indicates the need of having a convergence test after each iteration. The execution time for STIFF would be unaffected by the load type for the problems illustrated in Figs. 4.13b through 4.13d and would be the same as in Fig. 4.12.

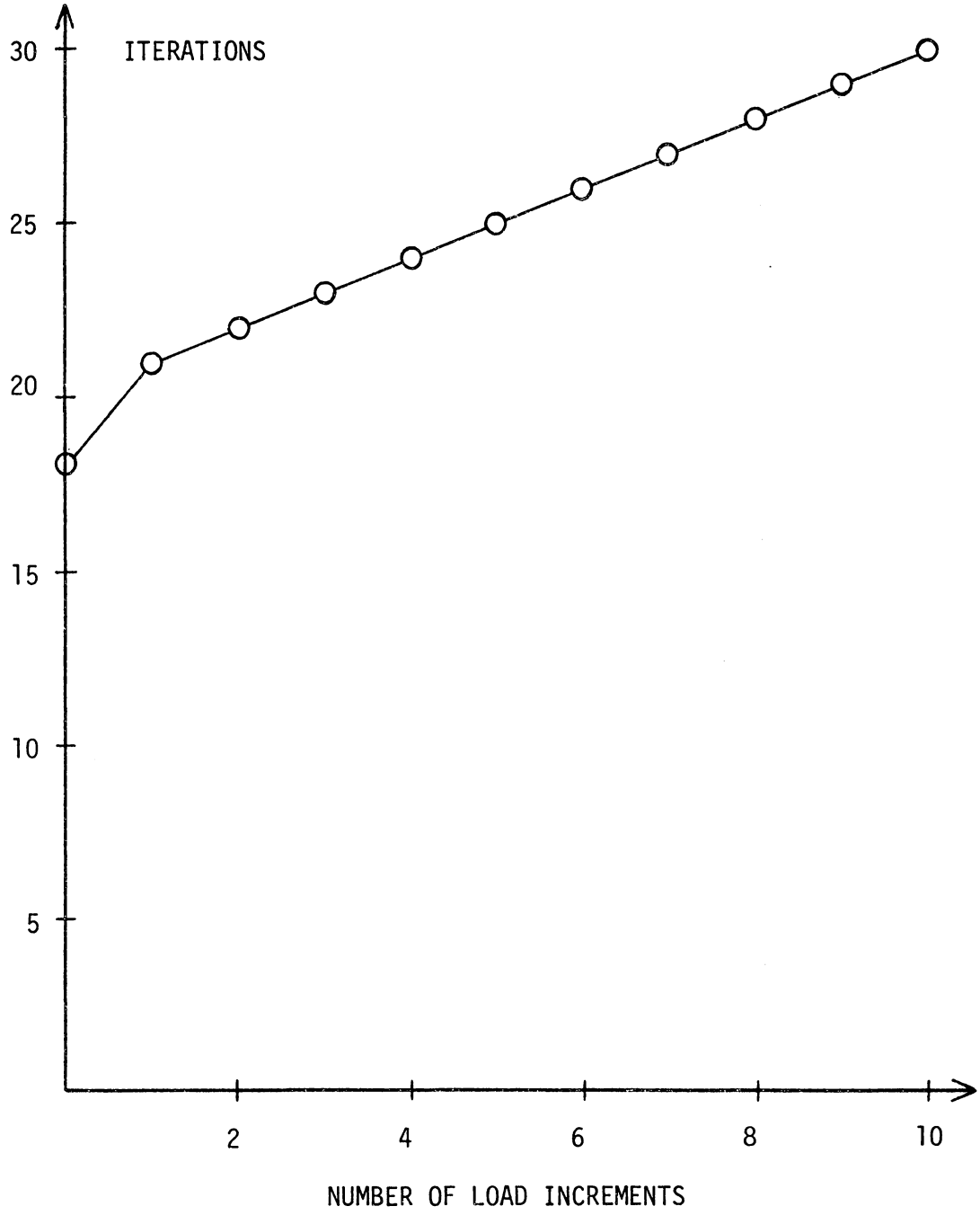



FIGURE 4.14 CUMULATIVE NUMBER OF ITERATIONS REQUIRED BY ENERGY (DFP) WHEN LOAD INCREMENTS ARE APPLIED

This is due to the direct method of solution of the equilibrium equations.

# Chapter 5

# SUMMARY

This chapter contains the conclusions arrived at based on the results of this study. Also included are recommendations for future study.

## 5.1 Conclusions

The performance of the stiffness method, STIFF, was superior to the energy method, ENERGY, in all comparisons based on execution time. This superiority is increasingly evident with only a small increase in problem size (6 d.o.f. compared to 18 d.o.f.). For larger problems (60 d.o.f.) the difference is so great further comparison based on execution time is meaningless. The accuracy of both methods was equivalent and, therefore, had no influence on the comparison.

The use of the FR and DFP algorithms to minimize the total potential indicates FR superior when only one load condition is considered. Although the convergence rate of DFP is superior to FR in all cases, the modest storage requirements of FR allows more iterations at a lower cost. The excessive number of iterations required by FR disallows the possibility of using it to estimate the inverse hessian.

When several load conditions were considered for the same structure ENERGY, using DFP, did compare more closely with STIFF after the initial solution process. Since DFP retains information from one solution process that will reduce the computational effort when additional load

vectors are applied, while FR does not, it is used when several load conditions are considered. However, the rate of increase in execution time by ENERGY (DFP) was uncertain and always greater than that of STIFF. From the indicated behavior of both methods it is expected this relation between ENERGY (DFP) and STIFF will continue for most problems.

The large storage requirements and iterative nature of ENERGY (DFP) creates the large gap between it and STIFF. An increase in d.o.f. means an increase in the size of the inverse hessian and the probable number of times it must be manipulated. Advantage is taken of the banded nature of the stiffness matrix by STIFF that keeps the storage requirements of the stiffness matrix from increasing as rapidly as the requirements of the inverse hessian. A convergence test is required during each iteration by ENERGY (DFP). This takes additional computational effort that is not required by STIFF since it uses a direct method of solution. The improvement in ENERGY (DFP) illustrated in Fig. 4.12 (Chapt. 4) still is not as efficient as STIFF, even when no updates of the inverse hessian are made. There still is the need to compute the gradient vector and the equivalent of a function evaluation in addition to a new search direction by ENERGY (DFP). This is required by the convergence criteria used.

The excessive storage demands and requirement of convergence tests at each iteration, even when a good initial estimate of the inverse hessian is available, makes ENERGY (DFP) inferior to STIFF for linear structural analysis.

## 5.2 Recommendations

The results of this study clearly show the superiority of the stiffness method over the method of minimizing the total potential energy for linear structural analysis. A future extension of this study should answer the question of how well the methods compare for nonlinear structural analysis. The material and results contained in this study give a reference point for such a study by giving insight into the performance of each method based on the linear problem. The performance of both methods should be much closer for nonlinear analysis. The superiority of the stiffness method in linear analysis does not necessarily indicate it superior for nonlinear analysis. When geometric and/or material nonlinearities are introduced, the force-displacement relation is no longer linear. Therefore the use of the stiffness method for nonlinear analysis becomes an iterative process. The stiffness matrix becomes a function of the generalized coordinates. If it is required for each iteration, that iteration uses approximately the same computational effort as solving a completely new problem.

The use of the energy method for nonlinear analysis is a natural extension of the linear analysis problem. Although linear analysis was done in this study, the total potential was a nonlinear function. Because it was quadratic, the variable metric algorithm used exhibited good convergence rates. (The use of a conjugate gradient algorithm is not recommended for nonlinear analysis due to the poor convergence rate for the linear case and the inability to retain information to reduce solution time when load increments are applied.) In some cases,

by using load increments in nonlinear analysis, the performance of the minimization algorithm may be expected to be similar to that reported in this study (if the initial load vector is small enough the total potential will be closely quadratic in form) when good gradient evaluations and line searches are available. Much of the material in this study, although heavily oriented toward quadratic functions, is also applicable to nonlinear analysis.

The literature survey included in this report indicates there are several variable metric algorithms that are preferred over the Davidon-Fletcher-Powell variable metric algorithm, used in this study, for minimization of general nonlinear functions. The use of the references cited in this survey is recommended as a guide to possibly selecting an algorithm superior to DFP. These references also indicate how the minimization code used in this study can be modified to handle general nonlinear functions.

# APPENDIX I

## REFERENCES


1. Allwright, J. C., "Conjugate Gradient Versus Steepest Descent," Journal of Optimization Theory and Application, Vol. 20, No. 1, September, 1976, pp. 129-134.

2. Bard, Y., "On Numerical Instability of Davidon-Like Methods," Mathematics of Computation, Vol. 22, No. 103, July, 1968, pp. 665-666.

3. Beckman, F. S., "The Solution of Linear Equations by the Conjugate Gradient Method," Mathematical Methods for Digital Computers, Ralston, A., and Wilf, H. S. (Eds.), John Wiley and Sons, Inc., New York, 1960.

4. Brodlie, K. W., "An Assessment of Two Approaches to Variable Metric Methods," Mathematical Programming, Vol. 12, No. 3, June, 1977, pp. 345-355.

5. Broyden, C. G., "Quasi-Newton Methods and Their Application to Function Minimization," Mathematics of Computation, Vol. 21, No. 99, July, 1967, pp. 368-381.

6. Cohen, A. I., "Rate of Convergence of Several Conjugate Gradient Algorithms," SIAM Journal on Numerical Analysis, Vol. 9, No. 2, June, 1972, pp. 248-259.

7. Cook, R. D., Concepts and Applications of Finite Element Analysis, John Wiley and Sons, Inc., New York 1974.

8. Davidon, W. C., "Variable Metric Method for Minimization," AEC Research and Development Report, ANL-5990 Rev., 1959.

9. Desai, C. S., and Abel, J. F., Introduction to the Finite Element Method, Van Nostrand Reinhold Co., New York, 1972.

10. Dixon, L. C. W., "Quasi-Newton Algorithms Generate Identical Points," Mathematical Programming, Vol. 2, No. 3, June, 1972, pp. 383-387.

11. Dixon, L. C. W., "Quasi Newton Techniques Generate Identical Points II: The Proofs of Four New Theorems," Mathematical Programming, Vol. 3, No. 3, December, 1972, pp. 345-358.

12. Fellipa, C. A., and Tocher, J. L., "Discussion: Efficient Solution of Load-Deflection Equations," Journal of the Structural Division, ASCE, Vol. 96, ST2, February, 1970, pp. 422-426.

13. Fletcher, R., and Reeves, C. M., "Function Minimization by Conjugate Gradients," Computer Journal, Vol. 7, 1964, pp. 149-154.

14. Fletcher, R. and Powell, M. J. D., "A Rapidly Convergent Descent Method for Function Minimization," Computer Journal, Vol. 6, No. 2, July, 1963, pp. 163-168.

15. Fletcher, R., "A New Approach to Variable Metric Algorithms," Computer Journal, Vol. 13, No. 3, August, 1970, pp. 317-322.

16. Fox, R. L., and Stanton, E. L., "Developments in Structural Analysis by Direct Energy Minimization," AIAA Journal, Vol. 6, No. 6, June, 1968, pp. 1036-1042.

17. Fox, R. L., Optimization Methods for Engineering Design, Addison-Wesley Publishing Company, Reading, Massachusetts, 1973.

18. Hestenes, M. R., and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems," Journal of Research of the National Bureau of Standards, Vol. 49, No. 6, December, 1952, pp. 409-436.

19. Himmelblau, D. M., Applied Nonlinear Programming, McGraw-Hill Book Company, New York 1972.

20. Holzer, S. M., "Lecture Notes on CE 4001: Design of Continuous Structures," Virginia Polytechnic Institute and State University, Winter Quarter, 1975, Blacksburg, Virginia.

21. Huang, H. Y., "Unified Approach to Quadratically Convergent Algorithms for Function Minimization," Journal of Optimization Theory and Applications, Vol. 5, No. 6, June, 1970, pp. 405-423.

22. Jenkins, W. M., Matrix and Digital Computer Methods in Structural Analysis, McGraw-Hill Publishing Company, Ltd., London, 1969.

23. Knight, N. F., Jr., "An Efficiency Assessment of Unconstrained Minimization Techniques as Applied to Nonlinear Structural Analysis," Thesis presented to Virginia Polytechnic Institute and State University, Blacksburg, Virginia, in 1977, in partial fulfillment of the requirements for the degree of Master of Science.

24. Langhaar, H. L., Energy Methods in Applied Mechanics, John Wiley and Sons, Inc., 1962.

25. Lenard, M. L., "Practical Convergence Conditions for the Davidon-Fletcher-Powell Method," Mathematical Programming, Vol. 9, No. 1, August, 1975, pp. 69-86.

26. Lenard, M. L., "Convergence Conditions for Restarted Conjugate Gradient Methods with Inaccurate Line Searches," Mathematical Programming, Vol. 10, No. 1, February, 1976, pp. 32-51.

27. Luenberger, D. G., Introduction to Linear and Nonlinear Programming, Addison-Wesley Publishing Company, Reading, Massachusetts, 1973.

28. Myers, G. E., "Properties of the Conjugate-Gradient and Davidon Methods," Journal of Optimization Theory and Applications, Vol. 2, No. 4, July, 1968, pp. 209-219.

29. Norris, D. O., and Gerken, J. D., "A Rank-One Algorithm for Unconstrained Function Minimization," Journal of Optimization Theory and Applications, Vol. 21, No. 3, March, 1977, pp. 261-275.

30. Oden, J. T., Mechanics of Elastic Structures, McGraw-Hill Book Company, Inc., New York, 1967.

31. Oren, S. S., "Self-Scaling Variable Metric Algorithms Without Line Search for Unconstrained Minimization," Mathematics of Computation, Vol. 27, No. 124, October, 1973, pp. 873-885.

32. Oren, S. S., and Luenberger, D. G., "Self-Scaling Variable Metric (SSVM) Algorithms Part I: Criteria and Sufficient Conditions for Scaling a Class of Algorithms," Management Science, Vol. 20, No. 5, January, 1974, pp. 845-862.

33. Oren, S. S., "Self-Scaling Variable Metric (SSVM) Algorithms Part II: Implementations and Experiments," Management Science, Vol. 20, No. 5, January, 1974, pp. 863-874.

34. Oren, S. S., and Spedicato, E., "Optimal Conditions of Self-Scaling Variable Metric Algorithms," Mathematical Programming, Vol. 10, No. 1, February, 1976, pp. 70-90.

35. Oren, S. S., "On Quasi-Newton and Pseudo-Newton Algorithms," Journal of Optimization Theory and Applications, Vol. 20, No. 2, October, 1976, pp. 155-170.

36. Powell, M. J. D., "Recent Advances in Unconstrained Optimization," Mathematical Programming, Vol. 1, No. 1, October, 1971, pp. 26-57.

37. Powell, M. J. D., "Some Convergence Properties of the Conjugate Gradient Method," Mathematical Programming, Vol. 11, No. 1, August, 1976, pp. 42-49.

38. Spedicato, E., "Computational Experience with Quasi-Newton Algorithms for Minimization Problems of Moderately Large Size," Centro Informazioni Studi Esperienzi, Report CISE-N-175, Segrate, Italy, 1975.

39. Vial, J. P., and Zang, I., "Unconstrained Optimization by Approximation of the Gradient Path," Mathematics of Operations Research, Vol. 2, No. 3, August, 1977, pp. 253-265.

# APPENDIX II

# USER's GUIDE FOR ENERGY

ENERGY is a computer code used for linear structural analysis. The analysis is performed by minimizing the total potential of the structure. The minimization is done by the Fletcher-Reeves conjugate gradient algorithm or the Davidon-Fletcher-Powell variable metric algorithm. The code is written in WATFIV computer language. A list of the subroutines used in ENERGY and a description of each is given below.

| SUBROUTINE | DESCRIPTION |
|---|---|
| MAIN | Reads data to determine array dimensions and specify minimization algorithm to use. |
| DIMN | Dimensions all arrays and calls all subroutines required for solution process. |
| INPUT | Reads data required to compute total potential and initializes starting values used by the minimization algorithms. |
| CONGRD | Minimizes the total potential by the FR algorithm and has option to estimate the inverse hessian. |
| DFP | Minimizes the total potential by the DFP algorithm. |
| COMPUT | Computes the gradient vector and helps in computing the step length during each iteration. |
| WLOAD | Prints solution data and reads additional load vectors when applied. |

The required data and input formats are given below.

CARD 1

KODE, COMENT (I5,9A8)

| | |
|---|---|
| CARD 2 | |
| N3,NM | (2I5) |
| CARD 3 | |
| (ICON(I),I=1,NJ3) | (80I1) |
| CARD 4 | |
| EP1,EP2 | (2D20.10) |
| CARD 5 | |
| LI,ICG,IDFP | (3I5) |
| CARD 6 | |
| ((XCORD(I,J),J=1,2),I=1,NJ) | (4D20.10) |
| CARD 7 | |
| A(I),E(I),FMI(I)  I=1,NM | (3D20.10) |
| CARD 8 | |
| ((INC(I,J),J=1,2),I=1,NM) | (16I5) |
| CARD 9 | |
| (P(I),I=1,NDF) | (4D20.10) |
| CARD 10 | |
| see below | |

When all data has been read leave CARD 10 blank to terminate code.
When additional load vectors are applied replace CARD 10 by CARD 9.
When a new problem begins go to CARD 1 and continue sequence.

The names used above have the following meanings

KODE  indicates solution process

KODE = 1  FR used ICG times to estimate the inverse hessian, final solution done by DFP

KODE = 2 FR used for entire solution process

KODE = 3 DFP used for entire solution process

| | |
|---|---|
| COMENT | any comment the user may want to identify the problem |
| NJ | number of joints |
| NM | number of members |
| ICON(I) | constraint matrix 0 = unconstrained<br>1 = constrained |
| NJ3 | NJ x 3 |
| EP1 | limit on gradient magnitude to define the global minimum |
| EP2 | limit on distance to next minimum to define the global minimum |
| LI | number of additional load vectors |
| ICG | limit on iterations by FR |
| IDFP | limit on iterations by DFP |
| XCORD(I,J) | joint coordinate matrix |
| A(I) | area of element I |
| E(I) | Young's modulus for element I |
| FMI(I) | moment of inertia for element I |
| INC(I,J) | member incidence matrix |
| P(I) | load vector |

# APPENDIX III

# USER'S GUIDE FOR STIFF

STIFF is a computer code used for linear structural analysis. The analysis is performed by the direct stiffness method. The equilibrium equations generated are solved by Gaussian elimination. The code is written in WATFIV computer language. A list of the subroutines used in STIFF and a description of each is given below.

| SUBROUTINE | DESCRIPTION |
|---|---|
| MAIN | Reads data to determine array dimensions and data for additional problems or load vectors. |
| DIMN | Dimensions all arrays and calls all subroutines required for solution process. |
| INPUT | Reads data on structure required to begin solution process and prints these values. |
| MPROP | Computes member properties required to generate the stiffness matrix and prints these values. |
| SSTIFF | Assembles the constrained stiffness matrix in banded form by degree of freedom. |
| SOLVE | Solves the equilibrium equations by Gaussian elimination and prints solution. |

The required data and input formats are given below.

CARD 1

KODE, COMENT (I5,9A8)

CARD 2

NJ,NM (2,I5)

CARD 3

((INC(I,J),J=1,2),I=1,NM) (16,I5)

| | |
|---|---|
| CARD 4 | |
| (ICON(I),I=1,NJB) | 80I1 |
| CARD 5 | |
| ((XCORD(I,J),I=1,2),I=1,NJ) | 4D20.10 |
| CARD 6 | |
| A(I),E(I),FMI(I) I=1,NM | 3D20.10 |
| CARD 7 | |
| (P(I),I=1,NDF) | 4D20.10 |
| CARD 8 | |
| KODE,B | I5,9A8 |

If KODE = 0 terminate code

IF KODE = 2 a new problem follows, return to CARD 1 and continued solution

If KODE = 5 go to CARD 7 and continue sequence

The names used above have the following meaning

| | | |
|---|---|---|
| COMENT | any comment the user may want to identify the problem | |
| NJ | number of joints | |
| NM | number of members | |
| INC(I,J) | member incidence matrix | |
| ICON(I,J) | constraint matrix | 0 = unconstrained |
| | | 1 = constrained |
| NJ3 | NJ x 3 | |
| XCORD(I,J) | joint coordinate matrix | |
| A(I) | area of element I | |

| | |
|---|---|
| E(I) | Young's modulus for element I |
| FMI(I) | moment of inertia for element I |
| P(I) | load vector |
| NDF | number of degrees of freedom |

# APPENDIX IV

```
C     READ DATA TO DETERMINE SOLUTION ALGORITHM TO BE USED AND          MAIN   1
C     ARRAY DIMENSIONS AND CALLS DIMN TO DIMENSION ARRAYS AND BEGIN     MAIN   2
C     ANALYSIS                                                          MAIN   3
C                                                                       MAIN   4
      IMPLICIT REAL*8 (A-H,O-Z)                                         MAIN   5
      COMMON /ONE/ F1,EP1,EP2,AK,SK                                     MAIN   6
      COMMON /TWO/ ICON(100),K,KODE,L1,NDF,NEL,NHMOD,NJ,NJ3,NLI,NM,NM2,1MAIN   7
     1CG,IDFP                                                           MAIN   8
      DIMENSION RD(5000), ID(500), COMENT(9)                            MAIN   9
10    READ (5,50) KODE,COMENT                                           MAIN  10
      IF (KODE.EQ.0) WRITE (6,70)                                       MAIN  11
      IF (KODE.EQ.0) STOP                                               MAIN  12
      READ (5,30) NJ,NM                                                 MAIN  13
      NM2=2*NM                                                          MAIN  14
      NJ3=NJ*3                                                          MAIN  15
      READ (5,40) (ICON(I),I=1,NJ3)                                     MAIN  16
      J=1                                                               MAIN  17
      DO 20 I=1,NJ3                                                     MAIN  18
      IF (ICON(I).NE.0) GO TO 20                                        MAIN  19
      J=J+1                                                             MAIN  20
      ICON(I)=J                                                         MAIN  21
20    CONTINUE                                                          MAIN  22
      NDF=J-1                                                           MAIN  23
      NEL=NDF*(NDF+1)/2                                                 MAIN  24
      WRITE (6,60) (COMENT(I),I=1,9)                                    MAIN  25
C                                                                       MAIN  26
C     VARIABLE ARRAY DIMENSIONS                                         MAIN  27
      N1=1                                                              MAIN  28
      N2=1+NEL                                                          MAIN  29
      N3=N2+NJ3                                                         MAIN  30
      N4=N3+NJ3                                                         MAIN  31
      N5=N4+NDF                                                         MAIN  32
```

```
      N6=N5+NDF                                                          MAIN  33
      N7=N6+NDF                                                          MAIN  34
      N8=N7+NDF                                                          MAIN  35
      N9=N8+NDF                                                          MAIN  36
      N10=N9+NDF                                                         MAIN  37
      N11=N10+NJ*2                                                       MAIN  38
      N12=N11+NJ3                                                        MAIN  39
      N13=N12+NM                                                         MAIN  40
      N14=N13+NM                                                         MAIN  41
      N15=N14+NM                                                         MAIN  42
      N16=N15+NM                                                         MAIN  43
      N17=N16+NM                                                         MAIN  44
      N18=1                                                              MAIN  45
      CALL DIMN (RD(N1),RD(N2),RD(N3),RD(N4),RD(N5),RD(N6),RD(N7),RD(N8)MAIN  46
     1,RD(N9),RD(N10),RD(N11),RD(N12),RD(N13),RD(N14),RD(N15),RD(N16),RDMAIN  47
     2(N17),ID(N18))                                                     MAIN  48
      GO TO 10                                                           MAIN  49
C                                                                        MAIN  50
30    FORMAT (16I5)                                                      MAIN  51
40    FORMAT (80I1)                                                      MAIN  52
50    FORMAT (I5,9A8)                                                    MAIN  53
60    FORMAT (1H1,12,9A8//)                                              MAIN  54
70    FORMAT (1H1)                                                       MAIN  55
      END                                                                MAIN  56
```

```
      SUBROUTINE DIMN (H,B,GD,G,GI,DG,D,P,X,XCORD,V,XL,A,FMI,SIN,COS,E,IDIMN   1
     1NC)                                                                    DIMN   2
C     DIMN DIMENSIONS ARRAYS AND CALLS SUBROUTINES FOR                       DIMN   3
C     SOLUTION PROCESS                                                       DIMN   4
C                                                                            DIMN   5
      IMPLICIT REAL*8(A-H,O-Z)                                               DIMN   6
      COMMON /ONE/ F1,EP1,EP2,AK,SK                                          DIMN   7
      COMMON /TWO/ ICON(100),K,KODE,LI,NDF,NEL,NHMOD,NJ,NJ3,NLI,NM,NM2,IDIMN   8
     1CG,IDFP                                                                DIMN   9
      DIMENSION H(NEL), B(NJ3), GD(NJ3), G(NDF), GI(NDF), DG(NDF), D(NDFDIMN  10
     1), P(NDF), X(NDF), XCORD(NJ,2), V(NJ3), XL(NM), A(NM), FMI(NM), SIDIMN  11
     2N(NM), COS(NM), E(NM), INC(NM,2)                                       DIMN  12
      CALL INPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)       DIMN  13
      IF (KODE.EQ.3) GO TO 30                                                DIMN  14
10    CALL CONGRD (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)      DIMN  15
      IF (KODE.EQ.1) GO TO 30                                                DIMN  16
      CALL WLOAD (G,D,P,X)                                                   DIMN  17
      IF (NLI.GT.LI) GO TO 40                                                DIMN  18
      K=2                                                                    DIMN  19
      CALL COMPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)      DIMN  20
      DO 20 I=1,NDF                                                          DIMN  21
20    D(I)=-GD(I)                                                            DIMN  22
      GO TO 10                                                               DIMN  23
30    CALL DFP (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)         DIMN  24
      CALL WLOAD (G,D,P,X)                                                   DIMN  25
      IF (NLI.GT.LI) GO TO 40                                                DIMN  26
      K=2                                                                    DIMN  27
      CALL COMPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)      DIMN  28
      GO TO 30                                                               DIMN  29
40    RETURN                                                                 DIMN  30
      END                                                                    DIMN  31
```

```
      SUBROUTINE INPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INPT   1
     1INC)                                                              INPT   2
C     INPUT READS INITIAL DATA AND COMPUTES ALL CONSTANTS REQUIRED      INPT   3
C     BY THE SOLUTION PROCESS AND INITIALIZES DATA REQUIRED BY THE      INPT   4
C     SOLUTION PROCESS                                                  INPT   5
C                                                                       INPT   6
      IMPLICIT REAL*8(A-H,O-Z)                                          INPT   7
      COMMON /ONE/ F1,EP1,EP2,AK,SK                                     INPT   8
      COMMON /TWO/ ICON(100),K,KODE,LI,NDF,NEL,NHMOD,NJ,NJ3,NLI,NM,NM2,IINPT   9
     1CG,IDFP                                                           INPT  10
      DIMENSION H(NEL), B(NJ3), GD(NJ3), G(NDF), GI(NDF), DG(NDF), D(NDFINPT  11
     1), P(NDF), X(NDF), V(NJ3), XL(NM), A(NM), FMI(NM), SIN(NM), COS(NMINPT  12
     2), E(NM), XCORD(NJ,2), INC(NM,2)                                  INPT  13
      READ (5,110) EP1,EP2                                              INPT  14
      READ (5,120) LI,ICG,IDFP                                          INPT  15
      READ (5,110) ((XCORD(I,J),J=1,2),I=1,NJ)                          INPT  16
      DO 10 I=1,NM                                                      INPT  17
10    READ (5,130) A(I),E(I),FMI(I)                                     INPT  18
      READ (5,120) ((INC(I,J),J=1,2),I=1,NM)                            INPT  19
      READ (5,110) (P(I),I=1,NDF)                                       INPT  20
C                                                                       INPT  21
C     INITIALIZE STARTING POINT DATA                                    INPT  22
      DO 20 I=1,NDF                                                     INPT  23
      X(I)=0.D0                                                         INPT  24
      D(I)=P(I)                                                         INPT  25
20    GD(I)=-P(I)                                                       INPT  26
      IF (KODE.EQ.2) GO TO 50                                           INPT  27
      DO 30 I=1,NEL                                                     INPT  28
30    H(I)=0.D0                                                         INPT  29
      IF (KODE.EQ.1) GO TO 50                                           INPT  30
      DO 40 I=1,NDF                                                     INPT  31
      G(I)=GD(I)                                                        INPT  32
```

```
      J=NDF*(I-1)+I-I*(I-1)/2                                          INPT  33
40    H(J)=1.D0                                                        INPT  34
50    CONTINUE                                                         INPT  35
      NLI=0                                                            INPT  36
C                                                                      INPT  37
C     COMPUTE MEMBER LENGTHS,SINES AND COSINES                         INPT  38
      DO 60 I=1,NM                                                     INPT  39
      J1=INC(I,1)                                                      INPT  40
      J2=INC(I,2)                                                      INPT  41
      XJ1=XCORD(J1,1)                                                  INPT  42
      XJ2=XCORD(J1,2)                                                  INPT  43
      XK1=XCORD(J2,1)                                                  INPT  44
      XK2=XCORD(J2,2)                                                  INPT  45
      DX1=XK1-XJ1                                                      INPT  46
      DX2=XK2-XJ2                                                      INPT  47
      XL(I)=DSQRT(DX1*DX1+DX2*DX2)                                     INPT  48
      COS(I)=DX1/XL(I)                                                 INPT  49
      SIN(I)=DX2/XL(I)                                                 INPT  50
60    CONTINUE                                                         INPT  51
      WRITE (6,140) EP1,EP2                                            INPT  52
      WRITE (6,150) LI,NM,NM2,NJ,NJ3,NDF,NEL,ICG,IDFP,KODE             INPT  53
      WRITE (6,180)                                                    INPT  54
      DO 70 I=1,NM                                                     INPT  55
70    WRITE (6,190) A(I),E(I),FMI(I),XL(I),INC(I,1),INC(I,2)           INPT  56
      WRITE (6,200)                                                    INPT  57
      DO 80 I=1,NM                                                     INPT  58
80    WRITE (6,220) SIN(I),COS(I)                                      INPT  59
      WRITE (6,210)                                                    INPT  60
      DO 90 I=1,NJ                                                     INPT  61
90    WRITE (6,220) XCORD(I,1),XCORD(I,2)                              INPT  62
      WRITE (6,160)                                                    INPT  63
      DO 100 I=1,NDF                                                   INPT  64
```

```
100   WRITE (6,170) I,P(I)                                                  INPT  65
      RETURN                                                                INPT  66
C                                                                           INPT  67
110   FORMAT (4D20.10)                                                      INPT  68
120   FORMAT (16I5)                                                         INPT  69
130   FORMAT (3D20.10)                                                      INPT  70
140   FORMAT (T18,3HEP1,T38,3HEP2/2D20.10)                                  INPT  71
150   FORMAT (/T4,2HLI,T9,2HNM,T13,3HNM2,T19,2HNJ,T23,3HNJ3,T28,3HNDF,T3INPT  72
     13,3HNEL,T38,3HICG,T42,4HIDFP,T47,4HKODE/10I5)                         INPT  73
160   FORMAT (/T2,19HDIRECTION OF D.O.F.,T37,4HP(1))                        INPT  74
170   FORMAT (I20,T21,D20.10)                                               INPT  75
180   FORMAT (/T17,4HA(I),T37,4HE(I),T55,6HFMI(I),T76,5HXL(I),T83,8HINC(INPT  76
     1I,1),T93,8HINC(I,2))                                                  INPT  77
190   FORMAT (D20.10,T21,D20.10,T41,D20.10,T61,D20.10,T81,I10,T91,I10,T1INPT  78
     101,I10)                                                               INPT  79
200   FORMAT (/T15,6HSIN(I),T35,6HCOS(I))                                   INPT  80
210   FORMAT (/T11,10HXCORD(I,1),T31,10HXCORD(I,2))                         INPT  81
220   FORMAT (2D20.10)                                                      INPT  82
      END                                                                   INPT  83
```

```
      SUBROUTINE CONGRD (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORDFRCG    1
     1,INC)                                                             FRCG    2
C     CONGRD USES THE FLETCHER-REEVES CONJUGATE GRADIENT ALGORITHM       FRCG    3
C     TO MINIMIZE THE TOTAL POTENTIAL                                    FRCG    4
C                                                                        FRCG    5
      IMPLICIT REAL*8 (A-H,O-Z)                                          FRCG    6
      COMMON /ONE/ F1,EP1,EP2,AK,SK                                      FRCG    7
      COMMON /TWO/ ICON(100),K,KODE,LI,NDF,NEL,NHMOD,NJ,NJ3,NLI,NM,NM2,IFRCG    8
     1CG,IDFP                                                           FRCG    9
      DIMENSION H(NEL), B(NJ3), GD(NJ3), G(NDF), GI(NDF), DG(NDF), D(NDFFRCG   10
     1), P(NDF), X(NDF), V(NJ3), XL(NM), A(NM), FMI(NM), SIN(NM), COS(NMFRCG   11
     2), E(NM), XCORD(NJ,2), INC(NM,2)                                  FRCG   12
      IZ=0                                                               FRCG   13
      NHMOD=0                                                            FRCG   14
      IF (KODE.NE.2.OR.NLI.EQ.0) GO TO 10                                FRCG   15
C                                                                        FRCG   16
C     COMPUTE STEPLENGTH FOR 1-D MINIMIZATION                            FRCG   17
10    GS=0.D0                                                            FRCG   18
      GK=0.D0                                                            FRCG   19
      DO 20 I=1,NDF                                                      FRCG   20
      GK=GK+GD(I)*GD(I)                                                  FRCG   21
20    GS=GS+GD(I)*D(I)                                                   FRCG   22
      K=1                                                                FRCG   23
      CALL COMPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)  FRCG   24
      AK=GS/F1                                                           FRCG   25
C                                                                        FRCG   26
C     VALUE OF MINIMUM ALONG SEARCH DIRECTION                            FRCG   27
      DO 30 I=1,NDF                                                      FRCG   28
30    X(I)=X(I)-AK*D(I)                                                  FRCG   29
      NHMOD=NHMOD+1                                                      FRCG   30
C                                                                        FRCG   31
C     COMPUTE GRADIENT AND DIRECTION VECTORS                             FRCG   32
```

```
      K=2                                                                    FRCG  33
      GK1=0.D0                                                               FRCG  34
      CALL COMPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)      FRCG  35
      DO 40 I=1,NDF                                                          FRCG  36
40    GK1=GK1+GD(I)*GD(I)                                                    FRCG  37
      BK1=GK1/GK                                                             FRCG  38
      DO 50 I=1,NDF                                                          FRCG  39
50    D(I)=BK1*D(I)-GD(I)                                                    FRCG  40
      IF (KODE.NE.1) GO TO 70                                                FRCG  41
C                                                                            FRCG  42
C     ESTIMATE INVERSE HESSIAN                                               FRCG  43
      M=0                                                                    FRCG  44
      DO 60 I=1,NDF                                                          FRCG  45
      DO 60 J=I,NDF                                                          FRCG  46
      M=M+1                                                                  FRCG  47
60    H(M)=H(M)+D(I)*D(J)/F1                                                 FRCG  48
      IF (NHMOD.EQ.ICG) GO TO 130                                            FRCG  49
      GO TO 10                                                               FRCG  50
70    CONTINUE                                                               FRCG  51
C                                                                            FRCG  52
C     CONVERGENCE TEST                                                       FRCG  53
      SK=0.D0                                                                FRCG  54
      I=0                                                                    FRCG  55
80    I=I+1                                                                  FRCG  56
      IF (DABS(GD(I)).GT.EP1) GO TO 110                                      FRCG  57
      IF (I.LT.NDF) GO TO 80                                                 FRCG  58
90    CONTINUE                                                               FRCG  59
      DO 100 I=1,NDF                                                         FRCG  60
100   SK=SK+D(I)*D(I)                                                        FRCG  61
      IF (DABS(AK)*DSQRT(SK).LT.EP2) GO TO 120                               FRCG  62
110   CONTINUE                                                               FRCG  63
C                                                                            FRCG  64
```

```
      K=3                                                          FRCG  65
      IF (NHMOD.EQ.ICG) GO TO 120                                  FRCG  66
      GO TO 10                                                     FRCG  67
120   CONTINUE                                                     FRCG  68
      IF (KODE.EQ.1) GO TO 130                                     FRCG  69
      IF (NHMOD.NE.ICG) GO TO 130                                  FRCG  70
      IZ=IZ+1                                                      FRCG  71
      IF (IZ.EQ.2) GO TO 130                                       FRCG  72
      IF (SK.EQ.0.D0) GO TO 90                                     FRCG  73
130   CONTINUE                                                     FRCG  74
      DO 140 I=1,NDF                                               FRCG  75
140   G(I)=GD(I)                                                   FRCG  76
      RETURN                                                       FRCG  77
      END                                                          FRCG  78
```

```
      SUBROUTINE DFP (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INDFPM   1
     1C)                                                                DFPM   2
C     DFP USES THE DAVIDON-FLETCHER-POWELL VARIABLE METRIC ALGORITHM     DFPM   3
C     TO MINIMIZE THE TOTAL POTENTIAL                                    DFPM   4
C                                                                        DFPM   5
      IMPLICIT REAL*8 (A-H,O-Z)                                          DFPM   6
      COMMON /ONE/ F1,EP1,EP2,AK,SK                                      DFPM   7
      COMMON /TWO/ ICON(100),K,KODE,LI,NDF,NEL,NHMOD,NJ,NJ3,NLI,NM,NM2,1DFPM   8
     1CG,IDFP                                                           DFPM   9
      DIMENSION H(NEL), B(NJ3), GD(NJ3), G(NDF), GI(NDF), DG(NDF), D(NDFDFPM  10
     1), P(NDF), X(NDF), V(NJ3), XL(NM), A(NM), FMI(NM), SIN(NM), COS(NMDFPM  11
     2), E(NM), XCORD(NJ,2), INC(NM,2)                                  DFPM  12
      NHMOD=0                                                           DFPM  13
      IJ=0                                                              DFPM  14
      KKK=0                                                             DFPM  15
      IF (KODE.EQ.3.AND.NLI.EQ.0) GO TO 70                              DFPM  16
10    CONTINUE                                                          DFPM  17
      DO 20 I=1,NDF                                                     DFPM  18
20    G(I)=GD(I)                                                        DFPM  19
30    CONTINUE                                                          DFPM  20
C                                                                       DFPM  21
C     COMPUTATION OF SEARCH DIRECTION                                   DFPM  22
      M=0                                                               DFPM  23
      DO 60 I=1,NDF                                                     DFPM  24
      D(I)=0.D0                                                         DFPM  25
      DO 40 J=I,NDF                                                     DFPM  26
      M=M+1                                                             DFPM  27
40    D(I)=D(I)-H(M)*GD(J)                                              DFPM  28
      IF (I.EQ.1) GO TO 60                                              DFPM  29
      J1=I-1                                                            DFPM  30
      DO 50 J=1,J1                                                      DFPM  31
      L=NDF*(J-1)+I-J*(J-1)/2                                           DFPM  32
```

```
50    D(I)=D(I)-H(L)*GD(J)                                                    DFPM  33
60    CONTINUE                                                                DFPM  34
70    CONTINUE                                                                DFPM  35
C                                                                             DFPM  36
C     COMPUTATION OF MAGNITUDE OF STEPLENGTH ALONG SEARCH DIRECTION           DFPM  37
      K=1                                                                     DFPM  38
      CALL COMPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)       DFPM  39
      GS=C.DO                                                                 DFPM  40
      DO 80 I=1,NDF                                                           DFPM  41
80    GS=GS+GD(I)*D(I)                                                        DFPM  42
      AK=GS/F1                                                                DFPM  43
      IF (KKK.EQ.1) GO TO 120                                                 DFPM  44
C                                                                             DFPM  45
C     VALUE OF MINIMUM ALONG SEARCH DIRECTION                                 DFPM  46
      DO 90 I=1,NDF                                                           DFPM  47
90    X(I)=X(I)-AK*D(I)                                                       DFPM  48
      NHMOD=NHMOD+1                                                           DFPM  49
C                                                                             DFPM  50
C     TEST IF GRADIENT VALUES ARE WITHIN PRESCRIBED LIMIT                     DFPM  51
      K=2                                                                     DFPM  52
      CALL COMPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORD,INC)       DFPM  53
      I=0                                                                     DFPM  54
100   I=I+1                                                                   DFPM  55
      IF (DABS(GD(I)).GT.EP1) GO TO 150                                       DFPM  56
      IF (I.NE.NDF) GO TO 100                                                 DFPM  57
      DO 110 I=1,NDF                                                          DFPM  58
110   GI(I)=D(I)                                                              DFPM  59
      F2=F1                                                                   DFPM  60
      KKK=1                                                                   DFPM  61
      GO TO 30                                                                DFPM  62
120   CONTINUE                                                                DFPM  63
C                                                                             DFPM  64
```

```
C       TEST TO DETERMINE IF DISTANCE TO NEXT MINIMUM IS WITHIN          DFPM  65
C       PRESCRIBED LIMITS                                                DFPM  66
        SK=0.D0                                                          DFPM  67
        DO 130 I=1,NDF                                                   DFPM  68
130     SK=SK+D(I)*D(I)                                                  DFPM  69
        IF (DABS(AK)*DSQRT(SK).LT.EP2) GO TO 220                         DFPM  70
        DO 140 I=1,NDF                                                   DFPM  71
140     D(I)=GI(I)                                                       DFPM  72
        F1=F2                                                            DFPM  73
        KKK=0                                                            DFPM  74
150     CONTINUE                                                         DFPM  75
        K=4                                                              DFPM  76
        IF (IDFP.LT.NHMOD) GO TO 220                                     DFPM  77
C                                                                        DFPM  78
C       MODIFICATION OF THE INVERSE HESSIAN (FLEXIBILITY MATRIX)         DFPM  79
        DO 160 I=1,NDF                                                   DFPM  80
160     DG(I)=GD(I)-G(I)                                                 DFPM  81
        BI=0.D0                                                          DFPM  82
        M=0                                                              DFPM  83
        DO 200 I=1,NDF                                                   DFPM  84
        B(I)=0.D0                                                        DFPM  85
        DO 170 J=I,NDF                                                   DFPM  86
        M=M+1                                                            DFPM  87
170     B(I)=B(I)+H(M)*DG(J)                                             DFPM  88
        IF (I.EQ.1) GO TO 190                                            DFPM  89
        J1=I-1                                                           DFPM  90
        DO 180 J=1,J1                                                    DFPM  91
        L=NDF*(J-1)+I-J*(J-1)/2                                          DFPM  92
180     B(I)=B(I)+H(L)*DG(J)                                             DFPM  93
190     BI=BI+B(I)*DG(I)                                                 DFPM  94
200     CONTINUE                                                         DFPM  95
        M=0                                                              DFPM  96
```

```
      DO 210 I=1,NDF                                                    DFPM  97
      DO 210 J=I,NDF                                                    DFPM  98
      M=M+1                                                             DFPM  99
210   H(M)=H(M)+D(I)*D(J)/F1-B(I)*B(J)/BI                               DFPM 100
      GO TO 10                                                          DFPM 101
220   CONTINUE                                                          DFPM 102
      DO 230 I=1,NDF                                                    DFPM 103
230   G(I)=GD(I)                                                        DFPM 104
      RETURN                                                            DFPM 105
      END                                                               DFPM 106
```

```
      SUBROUTINE COMPUT (H,B,GD,G,GI,DG,D,P,X,V,XL,A,FMI,SIN,COS,E,XCORDCOMP   1
     1,INC)                                                              COMP   2
C     COMPUT COMPUTES THE GRADIENT VECTOR AND HELPS IN COMPUTING         COMP   3
C     THE STEPLENGTH FOR 1-D MINIMIZATION                                COMP   4
C                                                                        COMP   5
      IMPLICIT REAL*8(A-H,O-Z)                                           COMP   6
      COMMON /ONE/ F1,EP1,EP2,AK,SK                                      COMP   7
      COMMON /TWO/ ICON(100),K,KODE,LI,NDF,NEL,NHMOD,NJ,NJ3,NLI,NM,NM2,ICOMP   8
     1CG,IDFP                                                            COMP   9
      DIMENSION H(NEL), B(NJ3), GD(NJ3), G(NDF), GI(NDF), DG(NDF), D(NDFCOMP  10
     1), P(NDF), X(NDF), V(NJ3), XL(NM), A(NM), FMI(NM), SIN(NM), COS(NMCOMP  11
     2), E(NM), XCORD(NJ,2), INC(NM,2)                                   COMP  12
      J=0                                                                COMP  13
      IF (K.NE.1) GO TO 20                                               COMP  14
      F1=0.D0                                                            COMP  15
      DO 10 I=1,NJ3                                                      COMP  16
      V(I)=0.D0                                                          COMP  17
      IF (ICON(I).EQ.1) GO TO 10                                         COMP  18
      J=J+1                                                              COMP  19
      V(I)=D(J)                                                          COMP  20
10    CONTINUE                                                           COMP  21
      GO TO 40                                                           COMP  22
20    CONTINUE                                                           COMP  23
      DO 30 I=1,NJ3                                                      COMP  24
      V(I)=0.D0                                                          COMP  25
      GD(I)=0.D0                                                         COMP  26
      IF (ICON(I).EQ.1) GO TO 30                                         COMP  27
      J=J+1                                                              COMP  28
      V(I)=X(J)                                                          COMP  29
30    CONTINUE                                                           COMP  30
40    CONTINUE                                                           COMP  31
      DO 60 I=1,NM                                                       COMP  32
```

```
      J3=3*INC(I,1)                                                     COMP  33
      J2=J3-1                                                           COMP  34
      J1=J2-1                                                           COMP  35
      J6=3*INC(I,2)                                                     COMP  36
      J5=J6-1                                                           COMP  37
      J4=J5-1                                                           COMP  38
      C=COS(I)                                                          COMP  39
      S=SIN(I)                                                          COMP  40
      IF (K.NE.1) GO TO 50                                              COMP  41
      FVAL=A(I)*E(I)/(2.D0*XL(I))*(V(J5)*S+V(J4)*C-V(J2)*S-V(J1)*C)**2+2COMP  42
     1.D0*E(I)*FMI(I)/XL(I)*(V(J3)*V(J3)+V(J3)*V(J6)+V(J6)*V(J6)-3.D0/XLCOMP  43
     2(I)*(V(J5)*C-V(J4)*S-V(J2)*C+V(J1)*S)*(V(J3)+V(J6))+3.D0/(XL(I)*XLCOMP  44
     3(I))*(V(J5)*C-V(J4)*S-V(J2)*C+V(J1)*S)**2)                        COMP  45
      F1=F1+2.D0*FVAL                                                   COMP  46
      GO TO 60                                                          COMP  47
50    CONTINUE                                                          COMP  48
      A1=6.D0*E(I)*FMI(I)/(XL(I)*XL(I))*S*(-V(J3)-V(J6)+2.D0/XL(I)*(V(J5COMP  49
     1)*C-V(J4)*S-V(J2)*C+V(J1)*S))-A(I)*E(I)*C/XL(I)*(V(J5)*S+V(J4)*C-VCOMP  50
     2(J2)*S-V(J1)*C)                                                   COMP  51
      A2=6.D0*E(I)*FMI(I)/(XL(I)*XL(I))*C*(V(J3)+V(J6)-2.D0/XL(I)*(V(J5)COMP  52
     1*C-V(J4)*S-V(J2)*C+V(J1)*S))-A(I)*E(I)*S/XL(I)*(V(J5)*S+V(J4)*C-V(COMP  53
     2J2)*S-V(J1)*C)                                                    COMP  54
      A3=2.D0*E(I)*FMI(I)/XL(I)*(2.D0*V(J3)+V(J6)-3.D0/XL(I)*(V(J5)*C-V(COMP  55
     1J4)*S-V(J2)*C+V(J1)*S))                                           COMP  56
      A4=2.D0*E(I)*FMI(I)/XL(I)*(2.D0*V(J6)+V(J3)-3.D0/XL(I)*(V(J5)*C-V(COMP  57
     1J4)*S-V(J2)*C+V(J1)*S))                                           COMP  58
      GD(J1)=GD(J1)+A1                                                  COMP  59
      GD(J2)=GD(J2)+A2                                                  COMP  60
      GD(J3)=GD(J3)+A3                                                  COMP  61
      GD(J4)=GD(J4)-A1                                                  COMP  62
      GD(J5)=GD(J5)-A2                                                  COMP  63
      GD(J6)=GD(J6)+A4                                                  COMP  64
```

```
60      CONTINUE                                                    COMP  65
        IF (K.EQ.1) GO TO 80                                        COMP  66
        DO 70 I=1,NJ3                                               COMP  67
        IF (ICON(I).EQ.1) GO TO 70                                  COMP  68
        ICI=ICON(I)-1                                               COMP  69
        GD(ICI)=GD(I)-P(ICI)                                        COMP  70
70      CONTINUE                                                    COMP  71
80      CONTINUE                                                    COMP  72
        RETURN                                                      COMP  73
        END                                                         COMP  74
```

```
      SUBROUTINE WLOAD (G,D,P,X)                                        LOAD   1
C     WLOAD PRINTS SOLUTION AND GRADIENT VECTOR AND READS VALUES OF     LOAD   2
C     ADDITIONAL LOAD VECTORS WHEN APPLIED                              LOAD   3
C                                                                       LOAD   4
      IMPLICIT REAL*8 (A-H,O-Z)                                         LOAD   5
      COMMON /ONE/ F1,EP1,EP2,AK,SK                                     LOAD   6
      COMMON /TWO/ ICON(100),K,KODE,LI,NDF,NEL,NHMOD,NJ,NJ3,NLI,NM,NM2,ILOAD   7
     1CG,IDFP                                                           LOAD   8
      DIMENSION G(NDF), D(NDF), P(NDF), X(NDF)                          LOAD   9
      NLI=NLI+1                                                         LOAD  10
      SK1=DABS(AK)*DSQRT(SK)                                            LOAD  11
      IF (KODE.EQ.1) GO TO 40                                           LOAD  12
      IF (KODE.EQ.2) GO TO 30                                           LOAD  13
      IF (K.NE.4) GO TO 20                                              LOAD  14
10    WRITE (6,100)                                                     LOAD  15
      GO TO 60                                                          LOAD  16
20    WRITE (6,150) NHMOD,NLI                                           LOAD  17
      GO TO 60                                                          LOAD  18
30    IF (K.EQ.3) GO TO 10                                              LOAD  19
      WRITE (6,130) NHMOD,NLI                                           LOAD  20
      GO TO 60                                                          LOAD  21
40    IF (NLI.NE.1) GO TO 50                                            LOAD  22
      WRITE (6,140)                                                     LOAD  23
50    IF (K.NE.4) GO TO 20                                              LOAD  24
      GO TO 10                                                          LOAD  25
60    WRITE (6,110)                                                     LOAD  26
      DO 70 I=1,NDF                                                     LOAD  27
70    WRITE (6,160) I,X(I),P(I),G(I)                                    LOAD  28
      IF (K.GT.2) GO TO 80                                              LOAD  29
      WRITE (6,120) SK1                                                 LOAD  30
80    CONTINUE                                                          LOAD  31
      IF (NLI.GT.LI) GO TO 90                                           LOAD  32
```

```
      READ (5,170) (P(I),I=1,NDF)                                          LOAD  33
90    RETURN                                                               LOAD  34
C                                                                          LOAD  35
C                                                                          LOAD  36
100   FORMAT (T2,79H***CONVERGENCE CRITERIA NOT SATISFIED WITHIN PRESCRILOAD  37
     18ED NUMBER OF ITERATIONS***/T2,81HVALUES OF MINIMUM ALONG SEARCH DLOAD  38
     2IRECTION PRIOR TO TERMINATION OF SOLUTION PROCESS)                   LOAD  39
110   FORMAT (/T2,9HDIRECTION,T20,19HDISPLACEMENT VECTOR,T45,11HLOAD VECLOAD  40
     1TOR,T70,15HGRADIENT VECTOR/T2,9HOF D.O.F.,T20,10HAT MINIMUM,T45,10LOAD  41
     2HAT MINIMUM,T70,10HAT MINIMUM)                                       LOAD  42
120   FORMAT (/T2,54HDISTANCE TO NEXT MINIMUM IF SOLUTION PROCESS CONTINLOAD  43
     1UED,T56,D20.10///)                                                   LOAD  44
130   FORMAT (T2,41HMINIMUM LOCATED BY CONJUGATE GRADIENTS IN,T43,I3,T47LOAD  45
     1,26HITERATIONS FOR LOAD VECTOR,T73,I3)                               LOAD  46
140   FORMAT (T2,78HINITIAL H INVERSE USED BY DFP VARIABLE METRIC ESTIMALOAD  47
     1TED BY CONJUGATE GRADIENTS)                                          LOAD  48
150   FORMAT (T2,41HMINIMUM LOCATED BY DFP VARIABLE METRIC IN,T43,I3,T47LOAD  49
     1,26HITERATIONS FOR LOAD VECTOR,T73,I3)                               LOAD  50
160   FORMAT (T6,I5,T11,3D25.10)                                           LOAD  51
170   FORMAT (4D20.10)                                                     LOAD  52
      END                                                                  LOAD  53
```

# APPENDIX V

```
C       READ DATA TO DETERMINE ARRAY DIMENSIONS AND CALL DIMN TO          MAIN   1
C       DIMENSION ARRAYS AND BEGIN ANALYSIS                               MAIN   2
C                                                                         MAIN   3
        IMPLICIT REAL*8 (A-H,O-Z)                                         MAIN   4
        COMMON INC(75,2),ICON(99),NM,NJ,NEQ,IBAND,KODE,NDF,LINC           MAIN   5
        DIMENSION S(5000), II(500)                                        MAIN   6
        DIMENSION B(9)                                                    MAIN   7
10      READ (5,60) KODE,B                                                MAIN   8
        IF (KODE.EQ.0) GO TO 50                                           MAIN   9
        WRITE (6,70) (B(J),J=1,9)                                         MAIN  10
        IF (KODE.EQ.5) GO TO 40                                           MAIN  11
        LINC=0                                                            MAIN  12
        READ (5,80) NJ,NM                                                 MAIN  13
        READ (5,80) ((INC(I,J),J=1,2),I=1,NM)                             MAIN  14
        NEQ=3*NJ                                                          MAIN  15
        READ (5,90) (ICON(I),I=1,NEQ)                                     MAIN  16
        NDF=0                                                             MAIN  17
        DO 20 I=1,NEQ                                                     MAIN  18
        IF (ICON(I).EQ.1) GO TO 20                                        MAIN  19
        NDF=NDF+1                                                         MAIN  20
        ICON(I)=NDF+1                                                     MAIN  21
20      CONTINUE                                                          MAIN  22
C                                                                         MAIN  23
C       COMPUTE MAXIMUM SEMI-BAND WIDTH                                   MAIN  24
        MAXDIF=0                                                          MAIN  25
        DO 30 I=1,NM                                                      MAIN  26
        MDIF=INC(I,2)-INC(I,1)                                            MAIN  27
30      IF (MDIF.GT.MAXDIF) MAXDIF=MDIF                                   MAIN  28
        IBAND=3*(MAXDIF+1)                                                MAIN  29
C                                                                         MAIN  30
C       VARIABLE ARRAY DIMENSIONS                                         MAIN  31
        N1=1                                                              MAIN  32
```

```
      N2=1+NEQ                                                          MAIN  33
      N3=N2+NJ*2                                                        MAIN  34
      N4=N3+NM                                                          MAIN  35
      N5=N4+NM                                                          MAIN  36
      N6=N5+NM                                                          MAIN  37
      N7=N6+NM                                                          MAIN  38
      N8=N7+NM                                                          MAIN  39
      N9=N8+NM                                                          MAIN  40
      N10=N9+NM                                                         MAIN  41
      N11=N10+NM                                                        MAIN  42
      N12=N11+IBAND*NDF                                                 MAIN  43
      N13=N12+NM*6                                                      MAIN  44
      N14=1                                                             MAIN  45
C                                                                       MAIN  46
40    CALL DIMN (S(N1),S(N2),S(N3),S(N4),S(N5),S(N6),S(N7),S(N8),S(N9),SMAIN  47
     1(N10),S(N11),S(N12),S(N13),II(N14))                               MAIN  48
      GO TO 10                                                          MAIN  49
50    WRITE (6,100)                                                     MAIN  50
      STOP                                                              MAIN  51
C                                                                       MAIN  52
60    FORMAT (I5,9A8)                                                   MAIN  53
70    FORMAT (1H1,T5,9A8//)                                             MAIN  54
80    FORMAT (16I5)                                                     MAIN  55
90    FORMAT (80I1)                                                     MAIN  56
100   FORMAT (1H1)                                                      MAIN  57
      END                                                               MAIN  58
```

```
      SUBROUTINE DIMN (P,X,FMI,E,A,XL,COS,SIN,ALPHA,BETA,AK,FTIL,PIJ,MCODIMN   1
     1DE)                                                              DIMN   2
C     DIMN DIMENSIONS ARRAYS AND CALLS SUBROUTINES                     DIMN   3
C     FOR SOLUTION PROCESS                                             DIMN   4
C                                                                      DIMN   5
      IMPLICIT REAL*8(A-H,O-Z)                                         DIMN   6
      COMMON INC(75,2),ICON(99),NM,NJ,NEQ,IBAND,KODE,NDF,LINC          DIMN   7
      DIMENSION P(NEQ), X(NJ,2), FMI(NM), E(NM), A(NM), XL(NM), COS(NM),DIMN   8
     1 SIN(NM), ALPHA(NM), BETA(NM), AK(NEQ,IBAND), FTIL(NM,6), PIJ(NDF)DIMN   9
     2, MCODE(NM,6)                                                   DIMN  10
      CALL INPUT (X,A,FMI,E,P,PIJ)                                     DIMN  11
      IF (KODE.EQ.5) GO TO 10                                          DIMN  12
      CALL MPROP (X,COS,SIN,E,FMI,A,XL,ALPHA,BETA)                     DIMN  13
      CALL SSTIFF (COS,SIN,ALPHA,BETA,XL,AK,MCODE)                     DIMN  14
10    CALL SOLVE (AK,P,PIJ)                                            DIMN  15
      RETURN                                                           DIMN  16
      END                                                              DIMN  17
```

```
      SUBROUTINE INPUT (X,A,FMI,E,P,PIJ)                                INPT   1
C     INPUT READS IN KNOWN DATA ON STRUCTURE AND LOADING                INPT   2
C     NJ= NUMBER OF JOINTS, NM= NUMBER OF MEMBERS                       INPT   3
C     A= AREA, E= YOUNG'S MODULUS, FMI= MOMENT OF INERTIA               INPT   4
C     P= (3*NJ) VECTOR SPECIFING APPLIED JOINT FORCES                   INPT   5
C     X= (NJ,2) MATRIX OF JOINT COORDINATES                             INPT   6
C     INC(NM,2)= JOINT NUMBER AT A AND B ENDS RESPECTIVELY              INPT   7
C     ICON(3*NJ)=MATRIX SPECIFING JOINT STATUS                          INPT   8
C        1= CONSTRAINED, 0= UNCONSTRAINED                               INPT   9
C                                                                       INPT  10
      IMPLICIT REAL*8 (A-H,O-Z)                                         INPT  11
      COMMON INC(75,2),ICON(99),NM,NJ,NEQ,IBAND,KODE,NDF,LINC           INPT  12
      DIMENSION X(NJ,2), A(NM), FMI(NM), E(NM), P(NEQ), PIJ(NDF)        INPT  13
      LINC=LINC+1                                                       INPT  14
      IF (KODE.EQ.5) GO TO 20                                           INPT  15
      READ (5,90) ((X(I,J),J=1,2),I=1,NJ)                              INPT  16
      DO 10 I=1,NM                                                      INPT  17
10    READ (5,90) A(I),E(I),FMI(I)                                      INPT  18
20    READ (5,90) (PIJ(I),I=1,NDF)                                      INPT  19
      J=0                                                               INPT  20
      DO 30 I=1,NEQ                                                     INPT  21
      P(I)=0.D0                                                         INPT  22
      IF (ICON(I).EQ.1) GO TO 30                                        INPT  23
      J=J+1                                                             INPT  24
      P(I)=PIJ(J)                                                       INPT  25
30    CONTINUE                                                          INPT  26
      IF (KODE.NE.5) GO TO 40                                           INPT  27
      WRITE (6,170) LINC                                                INPT  28
      WRITE (6,180) (P(I),I=1,NEQ)                                      INPT  29
      GO TO 80                                                          INPT  30
40    WRITE (6,100) NJ,NM,NDF,NEQ                                       INPT  31
      WRITE (6,110)                                                     INPT  32
```

```
      DO 50 I=1,NJ                                                      INPT  33
50    WRITE (6,120) I,X(I,1),X(I,2)                                     INPT  34
      WRITE (6,130)                                                     INPT  35
      DO 60 I=1,NM                                                      INPT  36
60    WRITE (6,140) I,INC(I,1),INC(I,2)                                 INPT  37
      WRITE (6,150)                                                     INPT  38
      DO 70 I=1,NJ                                                      INPT  39
      DO 70 M=1,3                                                       INPT  40
      J=3*(I-1)+M                                                       INPT  41
70    WRITE (6,160) I,M,ICON(J),I,M,P(J)                                INPT  42
80    RETURN                                                            INPT  43
C                                                                       INPT  44
90    FORMAT (4D20.10)                                                  INPT  45
100   FORMAT (T2,4HNJ =,T8,I3,T14,4HNM =,T20,I3,T26,5HNDF =,T33,I3,T39,5INPT  46
     1HNEQ =,T46,I3)                                                    INPT  47
110   FORMAT (//T25,23HJOINT COORDINATE MATRIX/T32,8H(INCHES)//T18,12HX INPT  48
     1COORDINATE,T43,12HY COORDINATE/)                                  INPT  49
120   FORMAT (T2,5HJOINT,T9,I2,T12,D23.16,T37,D23.16)                   INPT  50
130   FORMAT (//T12,23HMEMBER INCIDENCE MATRIX/T15,17H(NON DIMENSIONAL)/INPT  51
     1/T14,5HA END,T28,5HB END/)                                        INPT  52
140   FORMAT (T2,6HMEMBER,T9,I2,T16,I2,T30,I2)                          INPT  53
150   FORMAT (///T26,17HJOINT CONSTRAINTS,T75,20HAPPLIED JOINT FORCES/T2INPT  54
     16,17H(NON DIMENSIONAL),T78,14H(INCHES, KIPS)/)                    INPT  55
160   FORMAT (T2,5HJOINT,T9,I2,T13,I2,T15,9HDIRECTION,T34,I2,T50,5HJOINTINPT  56
     1,T57,I2,T61,I1,T63,9HDIRECTION,T74,D23.16)                        INPT  57
170   FORMAT (//T2,20HDATA FOR LOAD VECTOR,T22,I3/)                     INPT  58
180   FORMAT (D20.10)                                                   INPT  59
      END                                                               INPT  60
```

```
      SUBROUTINE MPROP (X,COS,SIN,E,FMI,A,XL,ALPHA,BETA)                MPRP   1
C     MPROP COMPUTES THE MEMBER PROPERTIES, COSINE, SINE, ALPHA, BETA,  MPRP   2
C     AND LENGTH, WHERE ALPHA=EI/L**3, BETA=(L/R)**2, XI(I)=LENGTH      MPRP   3
C                                                                       MPRP   4
      IMPLICIT REAL * 8 (A-H,O-Z)                                       MPRP   5
      COMMON INC(75,2),ICON(99),NM,NJ,NEQ,IBAND,KODE,NDF,LINC           MPRP   6
      DIMENSION X(NJ,2), COS(NM), SIN(NM), E(NM), FMI(NM), A(NM), XL(NM)MPRP   7
     1, ALPHA(NM), BETA(NM)                                             MPRP   8
      DO 10 I=1,NM                                                      MPRP   9
      J=INC(I,1)                                                        MPRP  10
      K=INC(I,2)                                                        MPRP  11
      XJ1=X(J,1)                                                        MPRP  12
      XJ2=X(J,2)                                                        MPRP  13
      XK1=X(K,1)                                                        MPRP  14
      XK2=X(K,2)                                                        MPRP  15
      DX1=XK1-XJ1                                                       MPRP  16
      DX2=XK2-XJ2                                                       MPRP  17
      XLEN=DSQRT(DX1**2+DX2**2)                                         MPRP  18
      XL(I)=XLEN                                                        MPRP  19
      COS(I)=DX1/XLEN                                                   MPRP  20
      SIN(I)=DX2/XLEN                                                   MPRP  21
      ALPHA(I)=E(I)*FMI(I)/XL(I)**3                                     MPRP  22
10    BETA(I)=(XL(I)/DSQRT(FMI(I)/A(I)))**2                             MPRP  23
      WRITE (6,40)                                                      MPRP  24
      DO 20 I=1,NM                                                      MPRP  25
20    WRITE (6,50) I,A(I),E(I),FMI(I),XL(I)                             MPRP  26
      WRITE (6,60)                                                      MPRP  27
      DO 30 I=1,NM                                                      MPRP  28
30    WRITE (6,70) I,COS(I),SIN(I),ALPHA(I),BETA(I)                     MPRP  29
      RETURN                                                            MPRP  30
C                                                                       MPRP  31
40    FORMAT (///T53,17HMEMBER PROPERTIES/T54,14H(INCHES, KIPS)///T21,4HMPRP  32
```

```
     1AREA,T38,21HMODULUS OF ELASTICITY,T65,17HMOMENT OF INERTIA,T92,14HMPRP  33
     2MEMBER LENGTHS//)                                                    MPRP  34
50    FORMAT (T2,6HMEMBER,T9,I2,T12,D23.16,T37,D23.16,T62,D23.16,T87,D23MPRP  35
     1.16)                                                                 MPRP  36
60    FORMAT (///T20,7HCOSINES,T46,5HSINES,T68,12HALPHA VALUES,T93,11HBEMPRP  37
     1TA VALUES/)                                                          MPRP  38
70    FORMAT (T2,6HMEMBER,T9,I2,T12,D23.16,T37,D23.16,T62,D23.16,T87,D23MPRP  39
     1.16)                                                                 MPRP  40
      END                                                                  MPRP  41
```

```
      SUBROUTINE SSTIFF (COS,SIN,ALPHA,BETA,XL,AK,MCODE)                STIF   1
C     STIFF COMPUTES THE ELEMENTS OF THE CONSTRAINED STIFFNESS MATRIX   STIF   2
C     CONTAINED IN THE SEMIBAND WIDTH                                   STIF   3
C                                                                       STIF   4
      IMPLICIT REAL*8 (A-H,O-Z)                                         STIF   5
      COMMON INC(75,2),ICON(99),NM,NJ,NEQ,IBAND,KODE,NDF,LINC           STIF   6
      DIMENSION COS(NM), SIN(NM), ALPHA(NM), BETA(NM), XL(NM), AK(NDF,IBSTIF   7
     1AND), MCODE(NM,6)                                                 STIF   8
      DIMENSION INDEX(6,6), AA(7)                                       STIF   9
      DATA INDEX/1,2,3,-1,-2,3,2,4,5,-2,-4,5,3,5,6,-3,-5,7,-1,-2,-3,1,2,STIF  10
     1-3,-2,-4,-5,2,4,-5,3,5,7,-3,-5,6/                                 STIF  11
      DO 10 I=1,NM                                                      STIF  12
      DO 10 J=1,6                                                       STIF  13
10    MCODE(I,J)=0                                                      STIF  14
      DO 20 I=1,NM                                                      STIF  15
      J1=3*INC(I,1)-2                                                   STIF  16
      J2=3*INC(I,1)-1                                                   STIF  17
      J3=3*INC(I,1)                                                     STIF  18
      J4=3*INC(I,2)-2                                                   STIF  19
      J5=3*INC(I,2)-1                                                   STIF  20
      J6=3*INC(I,2)                                                     STIF  21
      IF (ICON(J1).NE.1) MCODE(I,1)=ICON(J1)-1                          STIF  22
      IF (ICON(J2).NE.1) MCODE(I,2)=ICON(J2)-1                          STIF  23
      IF (ICON(J3).NE.1) MCODE(I,3)=ICON(J3)-1                          STIF  24
      IF (ICON(J4).NE.1) MCODE(I,4)=ICON(J4)-1                          STIF  25
      IF (ICON(J5).NE.1) MCODE(I,5)=ICON(J5)-1                          STIF  26
      IF (ICON(J6).NE.1) MCODE(I,6)=ICON(J6)-1                          STIF  27
20    CONTINUE                                                          STIF  28
      DO 30 I=1,NDF                                                     STIF  29
      DO 30 J=1,IBAND                                                   STIF  30
30    AK(I,J)=0.D0                                                      STIF  31
      DO 70 M=1,NM                                                      STIF  32
```

```
      ALPHAI=ALPHA(M)                                                   STIF   33
      BETAI=BETA(M)                                                     STIF   34
      S=SIN(M)                                                          STIF   35
      C=COS(M)                                                          STIF   36
      XLEN=XL(M)                                                        STIF   37
      AA(1)=ALPHAI*(BETAI*C*C+12.D0*S*S)                                STIF   38
      AA(2)=ALPHAI*C*S*(BETAI-12.D0)                                    STIF   39
      AA(3)=-6.D0*S*XLEN*ALPHAI                                         STIF   40
      AA(4)=ALPHAI*(BETAI*S*S+12.D0*C*C)                                STIF   41
      AA(5)=ALPHAI*6.D0*XLEN*C                                          STIF   42
      AA(6)=ALPHAI*4.D0*XLEN*XLEN                                       STIF   43
      AA(7)=AA(6)*0.5D0                                                 STIF   44
      DO 60 JM=1,6                                                      STIF   45
      J=MCODE(M,JM)                                                     STIF   46
      IF (J.EQ.0) GO TO 60                                              STIF   47
      DO 50 KM=JM,6                                                     STIF   48
      K=MCODE(M,KM)                                                     STIF   49
      IF (K.EQ.0) GO TO 50                                              STIF   50
      KB=K-J+1                                                          STIF   51
      L=INDEX(JM,KM)                                                    STIF   52
      IF (L.LT.0) GO TO 40                                              STIF   53
      AK(J,KB)=AK(J,KB)+AA(L)                                           STIF   54
      GO TO 50                                                          STIF   55
40    L=-L                                                              STIF   56
      AK(J,KB)=AK(J,KB)-AA(L)                                           STIF   57
50    CONTINUE                                                          STIF   58
60    CONTINUE                                                          STIF   59
70    CONTINUE                                                          STIF   60
      RETURN                                                            STIF   61
      END                                                               STIF   62
```

```
      SUBROUTINE SOLVE (AK,P,PIJ)                                       SOLV   1
C     SOLVE USES GAUSSIAN ELIMINATION TO SOLVE THE EQUILIBRIUM          SOLV   2
C     EQUATIONS AND PRINTS OUTPUT                                       SOLV   3
C                                                                       SOLV   4
      IMPLICIT REAL*8 (A-H,O-Z)                                         SOLV   5
      COMMON INC(75,2),ICON(99),NM,NJ,NEQ,IBAND,KODE,NDF,LINC           SOLV   6
      DIMENSION AK(NDF,IBAND), P(NEQ), PIJ(NDF)                         SOLV   7
      IF (KODE.EQ.5) GO TO 30                                           SOLV   8
C                                                                       SOLV   9
C     TRIANGULARZATION OF CONSTRAINED SYSTEM STIFFNESS MATRIX           SOLV  10
      NRS=NDF-1                                                         SOLV  11
      NR=NDF                                                            SOLV  12
      DO 20 N=1,NRS                                                     SOLV  13
      M=N-1                                                             SOLV  14
      MR=NR-M                                                           SOLV  15
      IF (IBAND.LT.MR) MR=IBAND                                         SOLV  16
      PIVOT=AK(N,1)                                                     SOLV  17
      DO 20 L=2,MR                                                      SOLV  18
      CP=AK(N,L)/PIVOT                                                  SOLV  19
      I=M+L                                                             SOLV  20
      J=0                                                               SOLV  21
      DO 10 K=L,MR                                                      SOLV  22
      J=J+1                                                             SOLV  23
10    AK(I,J)=AK(I,J)-CP*AK(N,K)                                        SOLV  24
20    AK(N,L)=CP                                                        SOLV  25
C                                                                       SOLV  26
C     REDUCTION OF THE LOAD VECTOR                                      SOLV  27
30    DO 40 N=1,NRS                                                     SOLV  28
      M=N-1                                                             SOLV  29
      MR=NR-M                                                           SOLV  30
      IF (IBAND.LT.MR) MR=IBAND                                         SOLV  31
      CP=PIJ(N)                                                         SOLV  32
```

```
      PIJ(N)=CP/AK(N,1)                                                    SOLV  33
      DO 40 L=2,MR                                                         SOLV  34
      I=M+L                                                                SOLV  35
40    PIJ(I)=PIJ(I)-AK(N,L)*CP                                             SOLV  36
      PIJ(NR)=PIJ(NR)/AK(NR,1)                                             SOLV  37
C                                                                          SOLV  38
C     SOLUTION OF REDUCED SYSTEM OF EQUATIONS BY BACK SUBSTITUTION         SOLV  39
      DO 50 I=1,NRS                                                        SOLV  40
      N=NR-I                                                               SOLV  41
      M=N-1                                                                SOLV  42
      MR=NR-M                                                              SOLV  43
      IF (IBAND.LT.MR) MR=IBAND                                            SOLV  44
      DO 50 K=2,MR                                                         SOLV  45
      L=M+K                                                                SOLV  46
50    PIJ(N)=PIJ(N)-AK(N,K)*PIJ(L)                                         SOLV  47
C                                                                          SOLV  48
      KK=0                                                                 SOLV  49
      DO 60 J=1,NEQ                                                        SOLV  50
      P(J)=0.D0                                                            SOLV  51
      IF (ICON(J).EQ.1) GO TO 60                                           SOLV  52
      KK=KK+1                                                              SOLV  53
      P(J)=PIJ(KK)                                                         SOLV  54
60    CONTINUE                                                             SOLV  55
      WRITE (6,80)                                                         SOLV  56
      DO 70 I=1,NJ                                                         SOLV  57
      DO 70 M=1,3                                                          SOLV  58
      J=3*(I-1)+M                                                          SOLV  59
      WRITE (6,90) I,M,P(J)                                                SOLV  60
70    CONTINUE                                                             SOLV  61
      RETURN                                                               SOLV  62
C                                                                          SOLV  63
80    FORMAT (//T26,27HGLOBAL JOINT DISPLACEMENTS /T31,16H(INCHES,RADIANSOLV  64
```

```
     1S)//)                                                              SOLV  65
90    FORMAT (T2,5HJOINT,T9,I2,T13,I1,T15,9HDIRECTION,T27,D20.10)        SOLV  66
      END                                                                SOLV  67
```

The vita has been removed from the scanned document

# COMPARISON OF ENERGY MINIMIZATION WITH DIRECT STIFFNESS FOR LINEAR STRUCTURAL ANALYSIS

by

David Thomas Griffith

(ABSTRACT)

This study compares energy minimization with direct stiffness for linear structural analysis. The energy minimization approach locates the generalized displacement vector by minimizing the total potential energy of the structure being analyzed. From the survey of variable metric and conjugate gradient algorithms included in this study, the Davidon-Fletcher-Powell variable metric algorithm and the Fletcher-Reeves conjugate gradient algorithm were chosen to minimize the total potential energy. A description of both algorithms is presented.

The direct stiffness method assembles the equilibrium equations of the structure beig analyzed. These equations are solved by Gaussian elimination to determine the generalized displacement vector.

Computer codes have been written for the energy minimization and direct stiffness methods. The comparison was based on computational effort, in terms of computer time, required for analysis. The results of this study show energy minimization is not competitive with direct stiffness for linear structural analysis. As the problem size increases by degree of freedom the direct stiffness method rapidly increases in superiority over the energy minimization method.