1973

Iterative solution of linear equations in finite element analysis, Ph.D. 1973

Erhard G. Schultchen

ITERATIVE SOLUTION OF LINEAR EQUATIONS IN FINITE ELEMENT ANALYSIS

by

Erhard G. Schultchen

A Dissertation

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Civil Engineering

Lehigh University
1973

CERTIFICATE OF APPROVAL

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

February 2, 1973
(Date)

Celal N. Kostem
Professor in Charge

Accepted March 2, 1973
(Date)

Special committee directing the doctoral work of Mr. Erhard Schultchen

Professor Lynn S. Beedle
Chairman

Professor Ti Huang

Professor Robert T. Folk

Professor David A. VanHorn
Ex Officio

ACKNOWLEDGMENTS

The author wishes to express his appreciation of the advice and assistance he received from Professor C. N. Kostem who supervised this dissertation. Sincere thanks are also extended to the members of the special committee, Professors L. S. Beedle (Chairman), T. Huang, R. T. Folk, and D. A. VanHorn.

Acknowledgments are due to the Lehigh University Computing Center for providing the technical facilities for the numerical study. The author also wishes to thank Miss Phyllis Raudenbush for her patience in typing the manuscript.

TABLE OF CONTENTS

PART II APPLICATION OF ITERATIVE METHODS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

A comparative study on the iterative solution of linear equations arising in the finite element analysis of structural problems is presented in this dissertation. Its primary objective is to determine the most suitable of those solution procedures in which the storage advantages of iterative methods can be exploited.

Part I of the dissertation contains a survey and classification of iterative and semi-iterative methods applicable to the solution of systems of equations with positive definite coefficient matrices. In addition, the survey includes various procedures for accelerating the convergence of iterative methods. The computational details of the algorithms are presented and a description of their specific properties is given. In order to investigate the convergence behavior and the efficiency of the solution procedures, numerical tests are carried out. As a result of the comparative study, the number of potentially useful algorithms is reduced to a total of three.

From additional numerical tests in Part II of the dissertation it is found that a particular version of the conjugate gradient method represents the most efficient solution procedure. Various means of improving its performance, such as scaling transformations, starting procedures, and algorithm modifications, are evaluated through numerical tests. The results indicate that a relatively simple diagonal scaling transformation allows an increase in its efficiency whereas other procedures are found to be of little or no practical value.

The dissertation also contains a survey of error prediction methods used for the termination of iterative solution processes. For the conjugate gradient method a new prediction procedure is proposed which provides comparatively accurate, conservative estimates of the relative error.

In addition, the practical application of the conjugate gradient algorithm is investigated in various aspects. Numerical tests are performed in order to study the effects of initial guesses and roundoff errors as well as specific problems in the solution of large systems of equations.

1. INTRODUCTION

1.1 Background

The development of electronic digital computers has made it possible to solve rather complex problems in the field of science and engineering by means of numerical methods. Among these procedures the finite element method has become a very versatile tool for solving a wide range of practical problems of structural analysis. As in many other numerical methods the results of the analysis are obtained from a system of linear equations whose solution may involve a rather large percentage of the total computational effort.

In its most widely used stiffness formulation, the finite element method requires the solution of a large, sparse system of linear equations of the form (Ref. 93)

$$Ku = f \tag{1.1}$$

where K is the global stiffness matrix of the discretized structure, u is the unknown displacement vector, and f represents the nodal point load vector. The various methods for solving such systems of equations can be categorized as:

(1) direct methods,

(2) iterative methods,

(3) semi-iterative methods, and

(4) Monte-Carlo methods.

Except for roundoff errors, direct methods yield the exact solution of the equations after a finite, predictable number of numerical operations. Iterative methods, on the other hand, yield sequences of approximate solutions which approach the exact solution asymptotically. Semi-iterative methods could be considered as a particular type of direct methods since in the absence of roundoff errors the exact solution is obtained within a finite number of numerical operations whose maximum can be predicted. However, the computational scheme of these methods as well as their behavior in the presence of roundoff errors are very similar to those of certain iterative methods. Monte-Carlo methods, which yield only statistical estimates of the solution, have found little application in the field of structural analysis and, therefore, are not discussed here.

The main problems in the solution of large sparse systems of linear equations are related to the storage of the coefficient matrix K and the computational effort for the solution process. Direct methods, which all involve some type of transformation of the coefficient matrix, require only a limited amount of numerical operations. The disadvantage of these methods is that the transformed coefficient matrix contains more non-zero elements and, therefore, requires more storage than K in its original form. Both the storage requirements as well as the computing time are strongly affected by the ordering of the equations, i.e. by the sequence in which the nodal points are enumerated. Therefore, various researchers have developed elaborate computational schemes in order to keep the storage requirements, the number of data transfers, and the computing time to a minimum (Refs. 46, 55, 69, 80, 85). In comparison with direct methods, iterative and semi-iterative methods have the advantage that the storage of K is essentially restricted to its non-zero elements and does not depend on the enumeration scheme. For most of these methods the global stiffness matrix does not even have to be available in assembled form, although the latter approach usually requires additional numerical operations. The major disadvantage of iterative methods is that the required computational effort is not only unpredictable, but may be rather large, too, at least for some of the less efficient algorithms.

Aside from storage and computing time, other factors such as roundoff errors and the simultaneous treatment of multiple load vectors may affect a decision on which type of solution method is to be preferred. For large problems, however, this decision will always depend on the size of the system of equations in relation to the available computing facilities. The continued interest in the iterative solution of linear equations indicates that the storage advantages of these methods are of considerable practical value for both small computers as well as large facilities used in time sharing mode (Refs. 28, 48, 61, 90). In order to take full advantage of the preferable aspects of iterative methods it is necessary to know which of the numerous iterative algorithms is efficient enough to be useful for practical applications. Therefore, a main objective of this investigation is to determine such a suitable method for systems of linear equations arising from finite element analysis.

1.2 Previous Work

The iterative solution of linear equations represents a problem of numerical analysis whose history goes back as far as the early nineteenth century and in which extensive research has been carried out (Ref. 7). The development of truly new iterative algorithms was essentially completed at the time when electronic digital computers came into practical use. Later contributions were made primarily in the field of acceleration procedures (Chapter 4) and in the development of specialized algorithms applicable only to particular type of coefficient matrices.

During the 1950's and early 1960's, when computers were used on an ever larger scale, the practical application of iterative methods was investigated extensively, resulting in a large number of publications in this field. The research almost exclusively concentrated on systems of equations arising from finite difference approximations of partial differential equations. The investigations dealt not only with convergence proofs, but also with the prediction of convergence rates and the determination of optimum acceleration factors. Since it is virtually impossible to give even a brief survey of these publications, reference is made to a number of books which summarize most of the research findings of this period. An extensive treatment of general iterative methods is given by Bodewig, Faddeev-Faddeeva, and Westlake (Refs. 7, 21, 84), whereas Forsythe-Wasow, Milne, and Varga (Refs. 25, 56, 82) deal with the iterative solution of finite difference equations. Throughout this study individual findings will be quoted mainly from these standard references rather than from less accessible original publications.

Many of these investigations include limited comparative studies of iterative solution methods. Apparently, however, only one major numerical investigation, covering most of the better known iterative methods, has been published (Ref. 17). This study as well as many previous publications deal with systems of equations which are of less general nature than those arising from finite element analysis (Appendix 1). Therefore, only few results of these investigations are directly applicable to the iterative solution of finite element equations. Within the last few years the solution of such more general problems has been investigated to a certain extent (Refs. 28, 61, 90), although no comparative studies have been published so far.

1.3 Purpose and Scope of Investigation

The subject of this investigation is the iterative solution of systems of n linear equations of the form

$$Ku = f \tag{1.2}$$

which arise in the stiffness formulation of the finite element method. A detailed description of the specific properties of these equations is given in Appendix 1. Since iterative methods are primarily used because of their potentially smaller storage requirements, the study covers only those solution procedures which actually allow such storage savings. In the presentation and discussion of iterative algorithms it is assumed that the coefficient matrix K is symmetric and positive definite. No attention is paid to specific problems which arise in connection with other types of coefficient matrices.

The first part of the dissertation contains a classification, description, and comparative study of numerous iterative, semi-iterative, and acceleration procedures. The majority of these algorithms are tested numerically in order to determine a suitable iterative solution method which should meet the following requirements:

(1) The convergence of the solution method should be stable and rapid enough to be useful for practical applications.

(2) Except for a limited number of additional vectors, the storage requirements of the algorithm should be essentially restricted to the non-zero elements of the original coefficient matrix. Moreover, the storage requirements should be independent of the ordering of the equations.

(3) The algorithm may not involve unknown quantities such as eigenvalue bounds unless simple "a priori" estimates requiring only a limited amount of numerical computations are available.

(4) If the algorithm contains acceleration parameters, well-defined limits within

which convergence is guaranteed to
occur, must be known for these quantities.
In addition, certain information on the
optimum range of these parameters should
be available.

The purpose of this comparative study is to reduce the total number of potentially useful algorithms to a minimum.

The second part of the investigation contains additional numerical tests of these remaining methods in order to identify the most suitable solution procedure. Various means of improving the performance of the selected method, such as scaling procedures and modifications of the basic algorithm, are investigated. Additional numerical tests are carried out in order to study the effects of initial guesses, roundoff errors, nodal point enumeration, and other factors of influence. The purpose of this second part is to guide the potential user in the practical application of iterative methods and to indicate possible effects on the behavior of the solution process.

PART I

COMPARATIVE STUDY OF ITERATIVE SOLUTION METHODS

2. EXECUTION OF NUMERICAL TESTS

2.1 Test Examples and Procedures

A comparative study of the efficiency of iterative methods for solving finite element equations necessarily has to be carried out numerically since theoretical comparisons of rates of convergence are not possible for such general type matrices. The numerical tests of this first part of the investigation are performed with a total of six problems of linear elastic finite element analysis of plane stress problems. Simple CST-elements (Constant Strain Triangles) with two degrees of freedom per nodal point are used in the discretization of the sample structures (Ref. 93). The test examples themselves, as well as various properties of the corresponding stiffness matrices, are described in Appendix 2.

In order to keep the total computational effort within reasonable limits, relatively small size systems of equations are chosen. Despite their small size and the relatively simple element type, the coefficient matrices of the test examples exhibit all the properties of general finite element matrices (Appendix 1). Since the performance of iterative methods is primarily affected by the condition rather than the size of the system of equations, the test examples are believed to be general enough for comparison purposes. Any decisions on the suitability of a particular method are based on the assumption that a method which shows instabilities or fails to converge rapidly enough for these test examples, cannot be expected to perform satisfactorily for larger practical problems.

All iterative solutions are started with zero initial guesses since better starting vectors are usually not available or difficult to generate. The numerical calculations are carried out on a CDC 6400 computer with a word length of 60 and a mantissa length of 48 bits, which allow the representation of a single precision number by approximately 14 significant digits.

2.2 Presentation of Results

Since the main objective of this first part of the investigation is to compare the efficiency of various iterative methods, it is necessary to define an appropriate measure of their performance. In order to be generally applicable, such a measure has to give reliable estimates of the amount of error reduction in relation to the required computational effort.

All results on the performance of iterative methods are, therefore, presented in the form of $m_{0.1}$-values, which represent the number of iteration cycles necessary to reduce the relative error of the maximum nodal point displacement to a value of 0.1%. A simple but realistic measure of the total computational effort for the solution process is obtained by multiplying these m-values by the number of matrix-vector products a particular iteration requires per iteration cycle (Appendix 3). For iterations which do not converge monotonically the m-values represent the number of iteration cycles after which the relative error does not exceed the specified value anymore. In cases where the m-values are not actually reached because of slow convergence, approximate values are determined by extrapolation of available data.

As most other convergence indicators, the m-values are, to a certain degree, affected by irregularities of the iteration process, particularly for iterative solutions which do not converge monotonically. Nevertheless, it is believed that these effects do not obscure those general trends which are the main concern of this investigation. It is realized that better, theoretically more meaningful measures, such as asymptotic rates of convergence (Section 3.2.1), are available for a number of iterative methods. However, for a comparative study such quantities are not suitable since they cannot be defined for certain types of solution methods.

Because of space limitations the presentation of numerical results is restricted to a small number of representative examples which constitute only a fraction of those results on which the observations and conclusions are based.

3. ITERATIVE METHODS

3.1 General Discussion

3.1.1 Definitions and Classification

Iterative solution methods can be defined as computational rules for operating on previous approximations u_c in order to obtain improved approximations u_{c+1}. In the absence of roundoff errors iterative methods yield the solution vector u after an infinite number of iteration cycles, provided the solution process does not diverge.

A general iterative algorithm for the solution of a system of linear equations

$$Ku = f \tag{3.1}$$

can be written in the form

$$u_{c+1} = I[K, f, u_c, u_{c-1}, \ldots, c] \tag{3.2}$$

The large number of iterative methods included in this general algorithm can be categorized on the basis of several different criteria (Refs. 17, 84). In this presentation only two such criteria are used for classifying all the iterative methods under investigation:

(a) linear/nonlinear iterations

(b) stationary/nonstationary iterations.

An iteration is said to be linear if the new approximation u_{c+1} is a linear function of the previous approximations u_c, u_{c-1},... On the other hand, an iteration is called stationary if the cycle counter c enters the algorithm only as a subscript for identifying previous approximations, but not as a variable or as the order of a polynomial. Specifically, the following types of algorithms are considered:

(a) linear stationary algorithms

(b) nonlinear stationary algorithms

(c) linear nonstationary algorithms.

The fourth possible combination is omitted since examples for nonlinear nonstationary algorithms are not found in the literature. Above classification is chosen since it allows an appropriate description of the nature of the algorithm and applies equally well to the classification of acceleration procedures (Chapter 4).

In order to describe the convergence behavior of iterative methods it is necessary to define suitable measures for the deviation between the exact and the approximate solution vectors. The following two vector quantities are commonly used for this purpose:

$$\begin{aligned} e_c &= u-u_c && \text{error vector} \\ r_c &= f-Ku_c = Ke_c && \text{residual vector} \end{aligned} \tag{3.3}$$

Since the exact solution vector u is the objective of the solution process, the error vector e_c remains generally unknown whereas the residual vector r_c can be calculated for any approximate solution. For systems of linear equations arising from finite element analysis the elements of this residual vector can be interpreted as unbalanced forces of the nodal point equilibrium equations. The nature of the relationship between the error vector and the residual vector (Eq. 3.3) does not imply that small unbalanced forces necessarily correspond to small errors in the displacement vector. This fact makes a prediction of the error of the approximate solution on the basis of the residual vector rather difficult (cf. Chapter 8).

The nature of an iterative algorithm is defined by its so-called iteration matrix T_c which relates the error vectors of two consecutive approximations.

$$e_c = T_c \, e_{c-1} \tag{3.4}$$

The total reduction of the initial error e_o can be expressed as

$$e_c = T_c T_{c-1} \dots T_2 T_1 e_o = \prod_{i=1}^{c} T_i e_o = M_c e_o \qquad (3.5)$$

where M_c represents the so-called error (matrix) polynomial. The solution process is guaranteed to converge if the largest eigenvalue of M_c is less than 1.0 in absolute value. Aside from this very general formulation it is usually possible to establish more practical, explicit convergence conditions for specific iterative algorithms.

Based on the spectral radius $\|M_c\|_{sr}$ and the spectral norm $\|M_c\|_2$ of the error polynomial it is possible to define the following rates of convergence

$$\begin{aligned} \rho_c &= -\frac{\log\|M_c\|_2}{c} && \text{average rate of convergence} \\ \rho_\infty &= -\frac{\log\|M_c\|_{sr}}{c} \quad \text{as } c\to\infty && \text{asymptotic rate of convergence} \end{aligned} \qquad (3.6)$$

The use of latter quantity is restricted to iterations which show an asymptotic convergence behavior whereas the average rate of convergence is free of such limitations. Above quantities represent theoretical measures of the performance of iterative algorithms and as such play a central role in the theory of iterative matrix methods (Refs. 82, 84). However, their practical importance is limited since the numerical evaluation of these quantities is difficult.

The convergence behavior of iterative solution methods is governed by the nature of the corresponding iteration matrices. In describing this behavior two special types of convergence, namely geometrical and linear, can be distinguished (Ref. 84). An iteration is said to converge geometrically if two consecutive error vectors are related by a constant scalar factor, that is, if the iteration matrix T_c can be written in the form

$$\begin{gathered} T_c = \tau I \\ e_c = T_c e_{c-1} = \tau e_{c-1} \end{gathered} \tag{3.7}$$

where τ is a constant factor less than 1.0. On the other hand, an iteration is said to have linear convergence if the iteration matrix is of the form

$$T_c = \tau I + \overline{T}_c \tag{3.8}$$

where $\overline{T}_c$ is a matrix whose elements approach zero for c approaching infinity. Most iterative methods discussed here exhibit linear convergence whereas geometric convergence occurs only asymptotically for linearly converging iterations.

3.1.2 Principles of Derivation

In order to establish a unifying concept for the large number of iterative solution methods, various investigators have suggested certain basic principles from which groups of such algorithms can be derived (Refs. 17, 42, 74, 84). Apparently, however, there exists no single derivation scheme which is general enough to be applicable to all iterative algorithms. The purpose of this section is to review some of the more important concepts of derivation and to give an indication of their limitations.

The first derivation scheme to be discussed here is based on the minimization of so-called error functions (Ref. 84). Among various possible types of such functions the following low order Schwarz constants play a dominant role (Ref. 73).

$$\begin{aligned} \phi_1(u_c) &= e_c^T e_c \\ \phi_2(u_c) &= e_c^T K e_c = e_c^T r_c \\ \phi_3(u_c) &= e_c^T K K e_c = r_c^T r_c \end{aligned} \qquad (3.9)$$

Other objective functions for the minimization process (Refs. 17, 23, 37, 73) have apparently found no practical application in the derivation of iterative methods for the solution of linear equations. Above error functions satisfy the relationships

$$\begin{aligned} \phi(u_c) &\geq 0 \quad \text{for arbitrary } u_c, \\ \phi(u_c) &= 0 \quad \text{only if } u_c = u, \end{aligned} \tag{3.10}$$

provided the coefficient matrix K is nonsingular, and in case of ϕ_2, also positive definite. For systems of linear equations arising from structural analysis, error function ϕ_2 is of particular importance since it is closely related to the total potential energy Π of the structure

$$\phi_2(u_c) = e_c^T K e_c = (u-u_c)^T K(u-u_c) = u^T f + u_c^T K u_c - 2u_c^T f$$

$$\Pi(u_c) = \frac{1}{2} u_c^T K u_c - u_c^T f, \tag{3.11}$$

therefore $\phi_2(u_c) = u^T f + 2\Pi(u_c)$.

From the above expressions it can be shown that both functions assume their minimum for the same vector $u_c = u$. However, the magnitude of error function ϕ_2 remains generally unknown, whereas the potential energy Π can be computed for any approximate solution vector.

The derivation scheme is applicable to those iterative algorithms which can be presented in the form

$$u_{c+1} = u_c+\gamma_c d_c \tag{3.12}$$

where d_c is a so-called direction vector and γ_c represents a scalar factor. The numerical value for γ_c is determined in such a way that the new approximation u_{c+1} minimizes one of the three error functions

$$\phi(u_{c+1}) = \phi(u_c+\gamma_c d_c) = \text{minimum} \tag{3.13}$$

By carrying out the minimization process the following γ_c-values are obtained (Ref. 84).

$$\begin{aligned} \phi_1: \quad \gamma_c &= \frac{r_c^T K^{-1} d_c}{d_c^T d_c} \\ \phi_2: \quad \gamma_c &= \frac{r_c^T d_c}{d_c^T K d_c} \\ \phi_3: \quad \gamma_c &= \frac{r_c^T K d_c}{d_c^T K K d_c} \end{aligned} \tag{3.14}$$

Any iterative algorithm of this kind is, therefore, completely defined by its direction vector and the selected error function. Typical choices for the direction vector d_c are r_c, Kr_c, e_i, and Ke_i, where e_i is a vector which corresponds to the i-th column of the identity matrix. A generalization of the above concept (Ref. 41) is based on the use of direction matrices rather than vectors and applies to certain block iterative methods (Section 3.2 and 3.3). The minimization of error functions allows the derivation of most linear and nonlinear iterations as well as accelerations; however, it does not apply to nonstationary algorithms. Essentially the same iterative methods can be obtained from the closely related principle of projection methods (Refs. 41, 42).

A second important concept of derivation is based on the theory of orthogonal polynomials (Refs. 21, 74, 75). The derivation scheme involves again the minimization of certain error measures, but unlike above, the minimization process is carried out over more than one iteration cycle. The results of this process are given as sets of γ-like scalar factors which usually contain the cycle counter c as a variable. Orthogonal polynomials are primarily used in the derivation of nonstationary iterations and accelerations, although they also apply to the derivation of semi-iterative solution methods.

The analogy between iterative processes and certain time dependent phenomena, commonly described by parabolic or hyperbolic differential equations, forms the basis of a third concept of derivation. Depending on the particular form of interpretation, this analogy can be applied to the derivation of iterative methods (Refs. 39, 82, 87), to the derivation of accelerations and accelerated iterations (Refs. 9, 39, 60, 87) as well as to the estimation of optimum acceleration factors (Refs. 18, 19, 32). The discussion of derivation concepts is restricted to these three major types, although several additional principles of more limited applicability are given in the literature.

Among various types of iterative solution methods, common principles can be found, not only in their derivation, but also in the way in which iterative algorithms are modified. Two of the most widely used modification procedures are based on so-called Gauss transformations of systems of linear equations (Ref. 21). The first Gauss transformation consists of pre-multiplying the original matrix equation (Eq. 3.1) by K and results in the following equivalent system

$$K_1 u_1 = f_1$$

where

$$\begin{aligned} K_1 &= KK \\ u_1 &= u \\ f_1 &= Kf \end{aligned} \qquad (3.15)$$

A similar transformed system of equations is obtained by the second Gauss transformation

$$K_2u_2 = f_2$$

where

$$\begin{aligned} K_2 &= KK \\ u_2 &= K^{-1}u \\ f_2 &= f \end{aligned} \tag{3.16}$$

The coefficient matrices of both transformed systems are necessarily positive definite, since for any nonsingular matrix A the matrix product A^TA has this particular property (Ref. 21). The basic purpose of Gauss transformations is, therefore, to allow those iterative algorithms, which converge only for positive definite coefficient matrices, to be applied to more general systems of equations. The transformations form the basis for re-formulating the original iterative procedure in order to obtain a new algorithm which does not require the explicit formation of the matrix product KK. Since the transformed coefficient matrix is more ill-conditioned than K itself, the above transformations are expected to have a detrimental effect on the performance of iterative methods (Ref. 84).

Both Gauss transformations can be considered as special cases of the following general transformation

$$K_3u_3 = f_3$$

where

$$K_3 = P_1KP_2$$

$$u_3 = P_2^{-1}u$$

$$f_3 = P_1 f \tag{3.17}$$

P_1, P_2 = transformation matrices

This third modification procedure also includes so-called "preparations" of systems of equations (Ref. 21) as well as various scaling transformations (Section 7.2).

It should be pointed out that these seemingly unrelated principles of deriving and modifying iterative algorithms are not strictly independent since it is possible to arrive at the same algorithm by means of different derivation schemes.

3.2 Linear Stationary Iterations

3.2.1 Classification

Among all iterative solution methods, linear stationary iterations form not only the largest group of algorithms, but also include some of the oldest, best known iterative methods. The particular nature of this type of iterations is reflected in the general operator

$$u_{c+1} = I[K, f, u_c] \tag{3.18}$$

in which the new approximation u_{c+1} is a linear function of u_c. All iterative methods discussed in this section are included in the following general algorithm

$$\begin{aligned} Q_1u_{c+1} &= Q_2u_c+f \\ K &= Q_1-Q_2 \end{aligned} \tag{3.19}$$

The matrices Q_1 and Q_2 represent so-called "splittings" of the coefficient matrix (Ref. 82) and define the nature of a particular algorithm. Since the new approximation u_{c+1} has to be obtained explicitly, the matrix Q_1 must be of such a form that a simple recursive calculation of the elements of u_{c+1} is possible.

Throughout this discussion an alternative presentation of the above algorithm is used which can be written as

$$\begin{aligned} u_{c+1} &= u_c+Sr_c \\ r_c &= f-K\,u_c \end{aligned} \tag{3.20}$$

where r_c represents the residual vector. In comparison with the previous formulation, the matrix S is related to Q_1 simply by

$$S = Q_1^{-1} \tag{3.21}$$

A main characteristic of linear stationary iterations is that the corresponding iteration matrix T depends only on the particular splitting of the coefficient matrix and remains constant throughout the iteration process. The error polynomial M_c (Section 3.1.1) can, therefore, be written in the form

$$\begin{aligned} M_c &= T^c \\ e_c &= M_c e_o = T^c e_o \end{aligned} \tag{3.22}$$

whereas the iteration matrix itself can be expressed as

$$T = I\text{-}SK \tag{3.23}$$

Because T remains constant, the definitions of the average and asymptotic rates of convergence (Section 3.1.1) assume the following simpler form

$$\begin{aligned} \rho_c &= -\frac{\log \|T^c\|_2}{c} \\ \rho_\infty &= -\log \|T\|_{sr} \end{aligned} \tag{3.24}$$

A necessary and sufficient condition for the convergence of a linear stationary iteration is given by

$$\|T\|_{sr} = \max_{i=1...n} |\lambda_i(T)| < 1 \qquad (3.25)$$

where $\lambda_i(T)$ represents the i-th eigenvalue of the corresponding iteration matrix. For most iterative methods of this type it is possible, however, to define more explicit convergence criteria which depend only on the nature of the coefficient matrix K (Section 3.2.2).

In order to categorize the large number of linear stationary iterations, a total of five basic groups of algorithms will be described first. In defining the corresponding S-matrices the following notation for splittings of the matrices K and KK is used:

$$(1) \quad K = L+D+L^T \qquad (3.26a)$$

where D is a diagonal matrix containing the diagonal elements of K, whereas L and L^T are the corresponding lower and upper triangular matrices;

$$(2) \quad KK = L_1+D_1+L_1^T \qquad (3.26b)$$

where D_1 and L_1 represent a similar splitting of the matrix KK; and

$$(3) \quad K_2 = L_2+D_2+L_2^T \tag{3.26c}$$

where D_2 is a quasi-diagonal matrix containing n_b principle submatrices of K of (not necessarily constant) size n_s, whereas L_2 and L_2^T are, again, the corresponding lower and upper triangular matrices.

The first basic group of linear stationary iterations, designated as Successive Approximation (Ref. 21), is defined by

$$\begin{aligned} S &= I \\ u_{c+1} &= u_c+r_c \end{aligned} \tag{3.27}$$

The algorithm represents the simplest, most fundamental type of iteration in the sense that all other basic groups can be reduced to this form by a general transformation of type 3 (Section 3.1.2).

The second basic group, named after Jacobi, encompasses some of the most widely known iterative solution methods and is defined by

$$\begin{aligned} S &= D^{-1} \\ u_{c+1} &= u_c+D^{-1}r_c \end{aligned} \tag{3.28}$$

The iterations of the Jacobi group can be derived by minimizing error function ϕ_2 with e_i as direction vector, where e_i corresponds to the i-th column of the identity matrix (Section 3.1.2). During each iteration cycle c the index i assumes all values from 1 to n in cyclic order. For the basic algorithm described here it is understood that the residual vector r_c is re-calculated only at the end of a complete iteration cycle, not after each single component of u_{c+1} is determined. The same special provisions, which represent a deviation from the usual minimization procedure, apply to the derivation of the remaining basic groups.

The algorithm for a third basic group of iterations, attributed to de la Garza, can be written as

$$\begin{aligned} S &= D_1^{-1}K \\ u_{c+1} &= u_c+D_1^{-1}Kr_c \end{aligned} \tag{3.29}$$

It is possible to derive the iterative methods of this group either by minimizing error function ϕ_3 with e_i as direction vector or by applying the first Gauss transformation to the corresponding algorithms of the Jacobi group (Section 3.1.2).

Similarly, a fourth group of iterations, named after Kaczmarz and defined by

$$S = KD_1^{-1}$$
$$u_{c+1} = u_c+KD_1^{-1}r_c \tag{3.30}$$

can be derived either by applying the second Gauss transformation to the corresponding methods of the Jacobi group or by minimizing error function ϕ_1 using Ke_i as direction vector.

The fifth basic group, designated as Block Jacobi, represents a modification of the Jacobi group in the sense that the diagonal matrix D (Eq. 3.28) is replaced by a quasi-diagonal matrix D_2 which contains principal submatrices of K rather than single diagonal elements.

$$S = D_2^{-1}$$
$$u_{c+1} = u_c+D_2^{-1}r_c \tag{3.31}$$

The iterative methods of this group can be derived from error function ϕ_2 by using direction matrices (rather than vectors) consisting of neighboring columns of the identity matrix. The algorithms of the Jacobi group could be considered as special cases of the corresponding block versions with block size n_s equal to one.

Although it is possible to apply similar block modifications to the de la Garza and Kaczmarz iterations

(Ref. 40), this possibility was not considered here. Additional iterative methods could also be derived either by applying Gauss transformations to the Successive Approximation group (Ref. 56) or by using other types of error functions and direction vectors for the minimization process. However, since none of these possible modifications have found practical applications, they are not included in this comparative study.

The S- and T-matrices of the above five basic groups as well as their corresponding algorithms are summarized in Table 1. Each of these iterative procedures can be subjected to a number of modifications whose nature is discussed in the following paragraphs.

The basic algorithms, from here on designated as Versions A, belong to a group of iterative methods which are commonly described as total-step iterations (Ref. 82). Their main characteristic is that every element of the new approximation u_{c+1} is computed solely on the basis of u_c or its corresponding residual. This fact is illustrated by the algorithm for Version A of the Jacobi group, which is presented here in index as well as matrix notation

$$(u)_i^{c+1} = (u)_i^c + \frac{1}{(K)_{ii}}\left[(f)_i - \sum_{j=1}^{n} (K)_{ij}(u)_j^c\right] \qquad (3.32a)$$

$$i = 1 \ldots n$$

$$u_{c+1} = u_c + D^{-1} r_c \qquad (3.32b)$$

The behavior of total-step iterations is not affected by the sequence in which the individual equations are treated or by the particular form of nodal point enumeration. Consequently, the stiffness matrix K does not have to be available in assembled form since the residual vector r_c can be computed on the basis of element stiffness matrices alone (Appendix 3).

The first type of modification (Version B) involves the re-formulation of the basic computational procedures as so-called Seidel processes. The essential feature of the modified algorithms is that the elements of the new approximate vector u_{c+1} are calculated on the basis of the most recently available approximations. Iterative methods of this type are designated as single-step methods (Ref. 82) and can be found only among linear stationary iterations. The application of the Seidel process to Version A of the Jacobi group results in the following algorithm (Gauss-Seidel iteration)

$$(u)_i^{c+1} = (u)_i^c + \frac{1}{(K)_{ii}} \left[(f)_i - \sum_{j=1}^{i-1} (K)_{ij} (u)_j^{c+1} - \sum_{j=i}^{n} (K)_{ij} (u)_j^c \right] \qquad (3.33a)$$

$$i = 1 \ldots n$$

$$u_{c+1} = u_c + (D+L)^{-1} r_c \qquad (3.33b)$$

The nature of the above algorithm indicates that the convergence behavior of single-step iterations depends on the sequence in which the individual equations are treated. In order to allow an efficient computer implementation of such algorithms, the stiffness matrix K has to be available in assembled form. As far as their derivation from error functions is concerned, single-step methods differ from total-step iterations by the fact that the elements of the residual vector have to be re-calculated after each single step, not only after the completion of a full iterative cycle (p. 32). The comparatively minor modification, thus, has important consequences not only for the convergence and the derivation of the iterative algorithms, but even for the storage of the coefficient matrix. From Eq. 3.33 it can also be seen that the presentation of single-step iterations in matrix notation bears unfortunately no resemblance to the actual computational scheme (Eq. 3.33a).

A second type of modification (Version C) is obtained by applying a so-called Aitken process to the basic iterative algorithms (Refs. 26, 84). The computational procedure essentially consists of two single-step iteration cycles, where the sequence in which the equations are treated is reversed during the second cycle. As an example, the

C-version of the Jacobi group (Aitken iteration) can be written as

$$(u)_i^{c+\frac{1}{2}} = (u)_i^{c}+\frac{1}{(K)_{ii}}\left[(f)_i-\sum_{j=1}^{i-1}(K)_{ij}(u)_j^{c+\frac{1}{2}}-\sum_{j=i}^{n}(K)_{ij}(u)_j^{c}\right]$$

$$i = 1...n$$

(3.34a)

$$(u)_i^{c+1} = (u)_i^{c+\frac{1}{2}}+\frac{1}{(K)_{ii}}\left[(f)_i-\sum_{j=1}^{i}(K)_{ij}(u)_j^{c+\frac{1}{2}}-\sum_{j=i+1}^{n}(K)_{ij}(u)_j^{c+1}\right]$$

$$i = n...1$$

$$u_{c+1} = u_c+(D+L)^{-1^T}D(D+L)^{-1}r_c \qquad (3.34b)$$

The purpose of reversing the equation sequence is to obtain an iterative algorithm whose iteration matrix is guaranteed to have only real eigenvalues (p. 40).

A total of three additional modifications (Versions D, E, and F) can be obtained by applying a linear stationary acceleration (Chapter 4) of the form

$$u_{c+1} = u_c+\omega[I(u_c)-u_c] \qquad (3.35)$$

to any of the versions discussed so far. In the above expression $I(u_c)$ represents a linear stationary iteration of type A, B, or C, whereas ω is a constant acceleration factor greater than zero. Since this particular acceleration procedure is so closely related to iterative algorithms, it is justifiable to treat the acceleration not as a separate algorithm (Section 4.2), but directly in context with the respective iterative methods. By applying the acceleration to the basic algorithm of the Jacobi group the following iterative method (extrapolated Jacobi iteration) results

$$(\bar{u})_i^{c+1} = (u)_i^c + \frac{1}{(K)_{ii}}\left[(f)_i - \sum_{j=1}^{n} (K)_{ij}(u)_j^c\right] \tag{3.36a}$$

$$(u)_i^{c+1} = (u)_i^c + \omega\left[(\bar{u})_i^{c+1} - (u)_i^c\right] \qquad i = 1\ldots n$$

$$(u)_i^{c+1} = (u)_i^c + \omega\frac{1}{(K)_{ii}}\left[(f)_i - \sum_{j=1}^{n} (K)_{ij}(u)_j^c\right] \tag{3.36b}$$

$$u_{c+1} = u_c + \omega D^{-1} r_c \tag{3.36c}$$

For single-step iterations it is understood that the acceleration is applied after each individual iteration step, not only at the end of a full iterative cycle. As an illustration, the accelerated form of the Gauss-Seidel iteration, usually designated as overrelaxation method, can be derived in the following way

$$(\bar{u})_i^{c+1} = (u)_i^c + \frac{1}{(K)_{ii}}\left[(f)_i - \sum_{j=1}^{i-1} (K)_{ij}(u)_j^{c+1} - \sum_{j=i}^{n} (K)_{ij}(u)_j^c\right] \tag{3.37a}$$

$$(u)_i^{c+1} = (u)_i^c + \omega[(\bar{u})_i^{c+1} - (u)_i^c] \qquad i = 1\ldots n$$

$$(u)_i^{c+1} = (u)_i^c+\omega\frac{1}{(K)_{ii}}\left[(f)_i-\sum_{j=1}^{i-1}(K)_{ij}(u)_j^{c+1}-\sum_{j=i}^{n}(K)_{ij}(u)_j^c\right] \tag{3.37b}$$

$$u_{c+1} = u_c+\omega(D+\omega L)^{-1}r_c \tag{3.37c}$$

The accelerated form of the corresponding C-version is obtained in a similar manner.

From Eq. 3.35 it can be seen that the acceleration is suppressed if the ω-factor assumes a value of 1.0. The unaccelerated algorithm versions A, B, and C could, therefore, be considered as special cases of the corresponding versions D, E, and F with ω set to this particular value.

The designation and basic features of the different types of modifications discussed above are explained schematically in Table 2. Table 3, on the other hand, contains the S-matrices of all linear stationary iterations as well as the commonly used names for some of the algorithms. In connection with Eq. 3.20, these S-matrices define the computational details of all iterative algorithms under consideration.

In addition, the iteration matrices for Versions D, E, and F of the five basic groups are listed in Table 4. By setting the acceleration factor ω equal to 1.0, the T-matrices of the corresponding unaccelerated versions are obtained. Since the convergence behavior of iterative solution methods strongly depends on the nature of their iteration matrices, it is important to know whether these matrices have real or complex eigenvalues. From Table 4 it can be seen that the T-matrices of Versions A, C, D, and F are either symmetric or their unsymmetric second part can be expressed as a product of symmetric and symmetric positive definite matrices. According to Theorems 11.3 and 11.14 of Ref. 21, in both cases the eigenvalues of the iteration matrices are real. On the other hand, the T-matrices for Versions B and E have real eigenvalues only for sufficiently small acceleration factors, whereas higher ω-values result in complex eigenvalues.

3.2.2 Algorithms

The computational characteristics of 30 linear stationary iterations are defined in Table 3 by means of so-called S-matrices. In order to provide some additional information on these algorithms, certain theoretical as well as practical aspects of their application are summarized in this section. For greater clarity the discussion is given separately for each of the five basic algorithm groups.

(1) Successive Approximation (Refs. 56, 73, 91)

Most iterative algorithms of the Successive Approximation group have found practical application only in cases where they are derived from the corresponding versions of the Jacobi group (Ref. 17) by a general transformation of type 3 (Section 3.1.2). An exception is made by Version D which is frequently used in connection with various acceleration procedures. One of the reasons for the limited importance of this group of iterations is that the unaccelerated Versions A, B, and C converge only if the largest eigenvalue of the coefficient matrix is in the order of 1. Since this condition is rarely satisfied for systems of equations arising from finite element analysis, the algorithms have to be considered as not suitable.

The iteration matrix for Version D of the Successive Approximation group

$$T = I-\omega K \tag{3.38}$$

is known to have real eigenvalues which are directly related to those of the coefficient matrix.

$$\lambda_i(T) = 1-\omega\lambda_i(K) \tag{3.39}$$

Among all iterative solution methods, the above algorithm represents the only case for which such a simple relationship between the eigenvalues of T and K can be established. From the general convergence criterion for linear stationary iterations (Eq. 3.25) it follows that the algorithm will converge for ω-values in the range

$$\begin{aligned} & & 0 < \omega < \frac{2}{\lambda_{max}(K)} & \\ \text{or} & & 0 < \omega' < 2 & \qquad (3.40) \\ \text{where} & & \omega = \frac{\omega'}{\lambda_{max}(K)} & \end{aligned}$$

The highest asymptotic rate of convergence is obtained for ω_{opt}, which can be expressed in terms of the eigenvalues of the coefficient matrix (Ref. 73)

$$\omega_{opt} = \frac{2}{\lambda_{max}(K)+\lambda_{min}(K)} \qquad (3.41)$$

For ω-values smaller than ω_{opt}, the dominant eigenvalue of T (i.e. the eigenvalue of largest absolute value) is positive, whereas it becomes negative for $\omega>\omega_{opt}$. The fact that $\lambda_{max}(K)$ remains generally unknown does not impair the applicability of the algorithm since relatively close upper bounds for this quantity can be easily calculated (Appendix 1). On the basis of such eigenvalue bounds it is possible to determine "safe" ω-values for which convergence is guaranteed to occur. However, the optimum acceleration factor will usually lie outside this particular range of ω-values.

(2) Jacobi (Refs. 17, 21, 25, 82, 84)

During the long history of their practical application the algorithms of the Jacobi group have become probably the best understood iterative methods. Numerous investigations have been carried out in order to study various aspects of this particular group of iterations. The basic algorithm (Jacobi iteration) does not, in general, converge for the type of linear equations which are considered here. The reason is that these equations satisfy none of the various convergence conditions which have been established in connection with certain forms of finite difference equations (Refs. 82, 84). However, convergence of the closely related Version D (extrapolated Jacobi iteration) can be achieved by selecting sufficiently small values of the acceleration factor. In particular, it is possible to determine "safe" ω-values on the basis of upper bounds for the spectral radius of the Jacobi A iteration matrix. The acceleration factor for Version D has a similar effect on the eigenvalues of the iteration matrix as for the corresponding version of the Successive Approximation group (p. 42). Unlike the previous case, however, it is generally not possible to express these eigenvalues directly in terms of $\lambda_i(K)$-values. Only if all diagonal elements of the coefficient matrix are identical, will Versions A and D of the Jacobi group coincide with Successive Approximation Version D.

For positive definite coefficient matrices the convergence of Version E of the Jacobi group (overrelaxation method) is guaranteed for any ω-value in the range from 0.0 to 2.0 (Ref. 84). This range also includes the corresponding unaccelerated Version B, commonly designated as Gauss-Seidel iteration. For low ω-values the eigenvalues of the iteration matrix are real, whereas higher acceleration factors cause at least some of the eigenvalues to become complex. As in all single-step iterations, the eigenvalues of the iteration matrix are not only affected by the acceleration factor, but also by the ordering of the equations.

The highest asymptotic rate of convergence is obtained for ω_{opt} which is defined as the ω-value for which the spectral radius of the iteration matrix assumes a minimum. Apparently, this ω-value coincides with the point at which the dominant eigenvalue of T becomes complex. A mathematical proof for this identity, however, can only be given for special types of coefficient matrices (Ref. 17). The optimum acceleration factor is usually greater than 1.0 and approaches the limiting value 2.0 for increasingly ill-conditioned problems.

The numerical evaluation of ω_{opt} has been the subject of a large number of investigations. Aside from purely experimental determination, essentially three different types of methods have been developed for this purpose. The first approach utilizes special properties of the coefficient matrices, such as consistent orderings, diagonal dominance, and property A (Appendix 1), in order to establish a relationship between ω_{opt} and the spectral radius of the iteration matrix for Version A of the Jacobi group (Refs. 17, 25, 82). Since linear equations arising from finite element analysis do not exhibit the above properties, the method has to be considered as not feasible. A second approach rests on the assumption that a lower bound for the smallest eigenvalue of the undiscretized elasticity problem can be established by certain means. Based on this quantity it is possible to determine approximate values of ω_{opt} which are applicable for sufficiently small mesh spacings (Refs. 18, 33, 82). The fact that such eigenvalue bounds are difficult to find for all but the simplest problems, makes this method useless for practical purposes. The behavior of the approximate solutions u_c is used by a third group of empirical techniques in order to improve estimates of ω_{opt} during the course of the iteration (Ref. 63). Unfortunately, these rather general methods are not applicable if acceleration procedures are applied to the iteration process. For the given type of linear equations it, therefore, has to be concluded that suitable methods for the determination of ω_{opt} are not available, at least not if overrelaxation is used in connection with acceleration procedures.

In a similar way as for Version E, the convergence of Version F (extrapolated Aitken iteration) is guaranteed for ω-values in the range from 0.0 to 2.0 (Refs. 26, 84). The eigenvalues of the corresponding iteration matrix, however, remain real throughout the full range of acceleration factors. Therefore, the iteration process does not exhibit the irregular convergence behavior which is associated with complex dominant eigenvalues. Although methods for the determination of ω_{opt} have also been developed for this particular iteration (Refs. 20, 54), their applicability is as limited as in the case of overrelaxation. In comparison with the remaining algorithms of the Jacobi group, Versions C and F require approximately twice as many numerical operations per iteration cycle (Appendix 3).

(3) de la Garza (Refs. 21, 42, 84)

In the discussion of basic groups it was shown that the iterative methods of the de la Garza group can be derived from the corresponding Jacobi versions by a Gauss transformation of the first kind (Section 3.1.2). In other words, the de la Garza iterations applied to the given system of equations (Eq. 3.1) are identical to the corresponding Jacobi algorithms applied to a similar transformed system of equations

$$KKu = Kf \tag{3.42}$$

The discussion of theoretical aspects of the Jacobi versions, therefore, applies equally well to the algorithms of the de la Garza group. The only major difference arises from the fact that in the application of the de la Garza versions matrix-vector products of the form KKu have to be formed instead of the usual Ku-products. The computational effort per iteration cycle will, therefore, be approximately twice as large as that of the corresponding Jacobi versions (Appendix 3). For the single-step versions of the de la Garza group certain computational problems arise in calculating the new iterate u_{c+1} on the basis of the most recently available approximations. In order to avoid a drastic increase in the number of arithmetic operations, the intermediate elements of the residual vector have to be calculated recursively. In this way it is possible to reduce the computational effort to two matrix-vector products per iteration cycle. However, the recursive calculation necessarily increases the danger that roundoff errors may affect the solution process.

(4) Kaczmarz (Refs. 7, 42, 84)

Since the iterative methods of the Kaczmarz group can be derived from the corresponding Jacobi versions by a second Gauss transformation, it can be concluded that the basic properties of the Jacobi, de la Garza, and Kaczmarz iterations are essentially the same. This correlation is illustrated by the fact that the iteration matrices for the de la Garza versions, T_{III}, are related to those of the Kaczmarz group, T_{IV}, by the following similarity transformation

$$T_{IV} = KT_{III}K^{-1} \tag{3.43}$$

As the eigenvalues of a matrix remain unchanged under a similarity transformation, the asymptotic behavior and the asymptotic rate of convergence of both types of iterations will be identical (cf. Ref. 4).

The iterative methods of the Kaczmarz group require a computational effort per iteration cycle which is about twice as large as that of the corresponding Jacobi versions (Appendix 3). However, for the single-step versions it is not necessary to rely on a recursive calculation of the residual vector as it is the case for the de la Garza versions.

(5) Block Jacobi (Refs. 17, 25, 82, 84)

Because of the close relationship between both types of methods, the previous discussion of various aspects of the Jacobi iterations applies equally well to the corresponding Block Jacobi versions. The main purpose of using these block modifications is to increase the rate of convergence of the iterative solution process. The amount of improvement largely depends on the selected block size n_s. Theoretically, n_s may assume any value from 1, in which case the original Jacobi versions result, to n, the total size of the system of equations. Various different choices for the block size n_s have found practical application in the literature. For instance, the so-called "alternate component iteration" (Ref. 61) is equivalent to block overrelaxation with n_s equal to the total number of nodal points. Other special cases, such as one- or two-line overrelaxation (Refs. 81, 82), are frequently used for the solution of certain types of finite difference equations. If Version A or B of the Block Jacobi group are used with n_s equal to n, a so-called "iterated direct method" for improving the solution of a system of equations, initially solved by direct methods, is obtained (Ref. 84). Although the effects of the block size n_s on the storage requirements and the computational effort per iteration cycle depend on the particular form of algorithm implementation, it nevertheless can be said that either one or both of these requirements will generally increase for increasing values of n_s.

3.2.3 Numerical Tests

In order to study the performance of linear stationary solution methods, numerical tests were carried out with most of the iterative algorithms described in the previous section. The results of these tests as well as various observations on the convergence behavior are summarized in the following discussion.

Among the five basic groups of linear stationary iterations, the algorithms of the Block Jacobi group are used as the basis for the comparison with other iterative methods. In the numerical tests of these block iterations the coefficient matrix K is partitioned in such a way that each principal submatrix contains the stiffness components of a single nodal point. The selected block size, therefore, corresponds to the number of degrees of freedom per nodal point, which is two for the given element type. This particular n_s-value was chosen since it offers certain advantages as far as the algorithm implementation is concerned. At the same time, it represents the maximum value for which the storage requirements of the modified coefficient matrix $D_2^{-1}K$ (Eq. 3.31) do not exceed those of the original K-matrix. The selected block size, therefore, complies with the requirements for suitable iterative solution methods stated in Section 1.3.

(1) Block Jacobi

Numerical tests with Version D of the Block Jacobi group (extrapolated Block Jacobi iteration) show that the iteration exhibits all the characteristics of a monotonic linear convergence. After an initially steeper decrease, the logarithm of the relative error ε_c becomes a linear function of the cycle counter c (Fig. 1), whereas the maximum element of the solution vector $(u)^c_{max}$ approaches its final value asymptotically without ever exceeding this quantity (Fig. 2). The behavior is characteristic of iterative processes which are dominated by a real, positive eigenvalue of the iteration matrix. The rate of convergence, which is directly related to the slope of the $\log(\varepsilon_c)$ vs. c curve, increases for increasing values of the acceleration factor until it reaches its maximum for ω_{opt} (Table 5). For the given type of linear equations it was found that the optimum acceleration factor is very close to the ω-value for which the iteration diverges. The convergence behavior of the iteration process was, therefore, not investigated within this particular range of acceleration factors. For most of the test examples, the solution process starts to diverge for ω-values smaller than 1.0. An exception is made by example A1 which satisfies one of the convergence conditions for the Jacobi iteration (Appendix 2). In this particular case divergence occurs for ω-values slightly larger than 1.0.

A similar behavior as that described for Version D can be observed for Version E of the Block Jacobi group (block overrelaxation method) provided the acceleration factor remains sufficiently small. Under this condition the iteration exhibits monotonic linear convergence, indicating that the dominant eigenvalue of the iteration matrix is real and positive. The similarity can be found in the behavior of both, the relative error ε_c (Fig. 3) as well as the maximum displacement $(u)^c_{max}$ whose variation with respect to c resembles the deflection-time diagram of an overdamped vibration (Fig. 4). In comparison with the previous iteration, the only minor difference occurs in the form of initial irregularities in the behavior of ε_c and $(u)^c_{max}$ which are caused by small complex eigenvalues of T.

A drastically different nature of the iteration process can be observed if the acceleration factor exceeds a certain problem-dependent value, ω_{opt}, for which the dominant eigenvalue of the iteration matrix becomes complex (p. 44). For this range of ω-values the relative error ε_c undergoes irregular cyclic oscillations which occur in constant cycle intervals c_o. Similarly, the maximum element of the solution vector oscillates about its final value and assumes the characteristics of a damped vibration. The nature of these oscillations does not only depend on the acceleration factor, but also on the condition of the given system of equations. Short cycle intervals c_o and rather erratic oscillations are observed particularly for well-conditioned problems and ω-values in the neighborhood of 2.0. The transition between monotonic and oscillating convergence, which occurs within a relatively narrow range of ω-values close to ω_{opt}, is illustrated in Fig. 5. For acceleration factors $\omega \geq 2.0$ the solution process diverges, in which case the variation of $(u)^c_{max}$ with respect to c resembles a vibration with zero or negative damping.

The above observations indicate that the convergence behavior of the solution process is strongly affected by the magnitude of the acceleration factor. A similar strong effect on the performance of the iteration is reflected in the convergence rates (Fig. 3 and 5) as well as the $m_{0.1}$-values of the test examples (Table 7). In general, the asymptotic rate of convergence increases until ω reaches the optimum acceleration factor. In the vicinity of this value the convergence rate assumes a sharp maximum within a relatively narrow range of ω-values. Unfortunately, this behavior is not accurately reflected in the results of Table 7 since m-values are to a certain degree affected by oscillations and irregularities in ε_c. The optimum acceleration factors of the test examples were found to lie in the range from 1.58 for example A2 up to 1.98 for example A3. Both the location of ω_{opt} as well as the magnitude of the attainable convergence rate, are influenced by the condition of the system of equations. The $m_{0.1}$-values of Table 7 show that the block overrelaxation method converges significantly faster than Version D of the Block Jacobi group.

Following a suggestion by Sheldon (Ref. 71), a slightly modified form of the block overrelaxation method was also tested numerically. The iteration differs from the usual algorithm only in the fact that ω is set equal to 1.0 during the first iteration cycle. Contrary to Sheldon's conjectures it was found that the modification has a minor detrimental, if any, effect on the performance of the iteration.

Numerical tests with Version F of the Block Jacobi group (extrapolated Block Aitken iteration) indicate that the solution process retains the characteristics of monotonic linear convergence (p. 51) throughout the full range of admissible acceleration factors. Even for ω-values greater than ω_{opt}, the relative error ε_c does not exhibit any kind of oscillations or irregularities in its behavior. From the test examples it is found that the optimum acceleration factor is greater than 1.0 and approaches this value for increasingly ill-conditioned problems (Table 6). In contrast to block overrelaxation, the rate of convergence of Version E changes only gradually in the vicinity of the optimum ω-value. By comparing the $m_{0.1}$-values of Tables 6 and 7, it can be concluded that the Block Aitken iteration is less efficient than block overrelaxation, especially since the former method requires twice as many numerical operations per iteration cycle. However, the Block Aitken iteration converges generally faster than Version D of the Block Jacobi group (cf. Table 5).

(2) Successive Approximation

The same type of monotonic linear convergence that was already described in connection with Block Jacobi Version D can also be observed for the corresponding version of the Successive Approximation group. For the given systems of equations, the optimum acceleration factors, defined by Eq. 3.41, nearly coincide with those ω'-values for which the iteration diverges. In certain cases, however, convergence can be observed for acceleration factors beyond the theoretical limit of $\omega' = 2.0$ (example A6). This abnormal behavior occurs if the initial error vector e_o is orthogonal to those eigenvectors of the coefficient matrix which correspond to the largest eigenvalues.

The performance of Version D of the Successive Approximation group is described in Table 8 for ω-values in the range from $1/\lambda_{max}(K)$ to $2/\lambda_{max}(K)$. Since the numerical value for $\lambda_{max}(K)$ remains generally unknown, the practical application of the iteration is restricted to the following range of acceleration factors

$$0 < \omega < \frac{2}{b_K} \qquad (3.44)$$

In the above expression, b_K represents an "a priori" upper bound for the maximum eigenvalue of the coefficient matrix (Appendix 1). The $m_{0.1}$-values of Table 8 clearly indicate that Version D of the Successive Approximation group is less efficient than the block overrelaxation method. At the same time, the iteration converges slower than other iterative methods whose T-matrices have only real eigenvalues (Block Jacobi Versions D and F). However, the difference in the performance of these methods are comparatively small.

(3) <u>Jacobi</u>

Numerical tests with Versions E and F of the Jacobi group show that the convergence behavior is very similar to that of the corresponding Block Jacobi versions. The previous discussion of various effects on the convergence, therefore, applies equally well to this group of iterative solution methods. The similarity between both types of algorithms is also reflected in the performance of the iterations (Tables 6 and 7). The test results indicate that the relatively small block size of the Block Jacobi methods allows only minor improvements in the convergence rates. For the E-versions, noticeable effects can be observed only in the vicinity of the optimum acceleration factors (example A4). Greater differences in the performance of Jacobi and Block Jacobi iterations could be expected, if larger values of the block size were chosen. For test example A1, which only involves nodal points with one degree of freedom, both types of iterations become virtually identical.

(4) de la Garza

The $m_{0.1}$-values of Table 9 indicate that Version D of the de la Garza group exhibits a slower convergence than any of the other iterative methods discussed so far. The difference becomes even more pronounced if it is taken into account that the algorithm requires two matrix-vector products per iteration cycle instead of the usual single product. The results clearly show that the Gauss transformation, by which the iteration is obtained from the corresponding version of the Jacobi group, has a detrimental effect on the performance (Section 3.1.2). Because of the very slow convergence it is not possible to give an accurate description of the convergence behavior. However, there are indications that the behavior resembles that of other D-versions applied to very ill-conditioned problems.

(5) Kaczmarz

The fact that the eigenvalues of the iteration matrix for Version D of the Kaczmarz group (Cimmino iteration) are identical to those of the corresponding de la Garza version (p. 48) is reflected in nearly identical $m_{0.1}$-values for both iterations (Table 9). The slow convergence of the solution makes both methods unsuitable for practical applications. Numerical tests with Version E of the Kaczmarz group indicate a better performance for this iterative method, although the results are not comparable with those for block overrelaxation. Since the iteration process is affected by initial irregularities within the tested range of iteration cycles, it is not possible to extrapolate $m_{0.1}$-values for the solution method. As in the previous case, the slow convergence of the iteration does not allow definite conclusions to be drawn on the convergence behavior.

From the results of this numerical investigation of linear stationary iterations, it is possible to draw a number of conclusions. The investigation clearly shows that Gauss transformations, by which the iterations of the de la

Garza and Kaczmarz group are obtained from the corresponding Jacobi versions, have a strong detrimental effect on the performance. The numerical tests also indicate that block overrelaxation represents the most efficient linear stationary iteration, although the corresponding "point" version is only slightly less efficient for the given block size. Among those iterative methods whose T-matrices are known to have real eigenvalues, the extrapolated Block Aitken iteration exhibits the fastest convergence. However, the differences with several other iterations, such as the D-versions of the Block Jacobi and Successive Approximation groups, are relatively small. It can also be concluded that the Gauss-Seidel iteration, either in point or block form, exhibits the best performance among the unaccelerated iteration versions.

In view of these results, the following three linear stationary iterations were selected for additional numerical tests in connection with acceleration procedures (Chapter 4):

(a) Block overrelaxation,

(b) Block Gauss-Seidel, and

(c) Successive Approximation Version D .

The iteration matrices for both, block overrelaxation and Block Gauss-Seidel iteration, are known to have complex eigenvalues (p. 44). Since most acceleration procedures are based on the assumption that these eigenvalues are real, an additional method was included which satisfies the above requirement. Version D of the Successive Approximation group was selected since its algorithm is simpler than that of Block Jacobi Version F. At the same time, "safe" ω-values for which convergence is guaranteed can be established more easily than in the case of Block Jacobi Version D.

3.3 Nonlinear Stationary Iterations

3.3.1 Algorithms

The general operator for nonlinear stationary iterations (or gradient methods as they are frequently called) can be expressed in the following form

$$u_{c+1} = I[K,f,u_c,u_{c-1},\ldots,s] \qquad (3.45)$$

The main characteristic of this group of iterative methods is that the new approximate vector u_{c+1} is obtained as a nonlinear function of the previous approximations. Among the large number of possible algorithms, the following basic iterations have found practical application in the literature

(1)	Steepest descent	(Ref. 21,84)
(2)	Krasnoselskii	(Ref. 21,74)

(3)	Householder	(Ref. 41,42)
(4)	Cauchy	(Ref. 56,84)
(5)	Gastinel-Householder	(Ref. 42)
(6)	Gastinel	(Ref. 33,42)

By comparing the computational forms of the above iterations (Table 10) with Eq. 3.12, it can be seen that the minimization of error functions is applicable to the derivation of all six algorithms (Section 3.1.2). The specific error functions and direction vectors used in deriving the iterations are, therefore, included in Table 10. As an alternative form of derivation, Householder's and Cauchy's methods could also be obtained by applying Gauss transformations to the algorithm of the steepest descent method.

If the γ-values of the first two algorithms are kept constant during the iteration process, both methods become identical to Version D of the Successive Approximation group (Section 3.2.2). Gastinel's method, or more precisely the idea of using the non-algebraic direction vector t_c

$$(t)_i^c = \text{sign}[(r)_i^c] = \frac{(r)_i^c}{|(r)_i^c|} \qquad (3.46)$$

is closely related to a so-called "block relaxation" suggested by Stiefel (Ref. 73). Except for the use of t_c in their direction vectors, the Gastinel-type iterations 5 and 6 are identical to algorithms 1 and 3 of Table 10. Two additional iterations could be obtained by applying similar modifications to Krasnoselskii's and Cauchy's methods. However, this possibility was not considered in this investigation.

In the application of most of the algorithms, a considerable amount of computational effort can be saved by making use of recursive relationships for the calculation of the new residual vector r_{c+1} (Appendix 3). For the steepest descent method such recursions allow a 50% reduction in the required number of matrix-vector products per iteration cycle.

$$\begin{aligned} u_{c+1} &= u_c + \frac{r_c^T r_c}{r_c^T K r_c} r_c \\ r_{c+1} &= f - Ku_{c+1} = f - Ku_c - \frac{r_c^T r_c}{r_c^T K r_c} K r_c \\ r_{c+1} &= r_c - \frac{r_c^T r_c}{r_c^T K r_c} K r_c \end{aligned} \tag{3.47}$$

However, in addition to somewhat higher storage requirements, the recursive evaluation of r_{c+1} has the disadvantage that roundoff errors may affect the solution process.

In order to increase the convergence rate of nonlinear stationary iterations, a number of investigators suggested various modifications of the basic algorithms discussed so far. In the remainder of this section several of these modifications are presented and their relationship with other iterative methods is discussed.

(1) Kantorovich's s-Step Gradient Method (Ref. 21)

The computational scheme for Kantorovich's s-step gradient method can be expressed in the following form

$$u_{c+1} = u_c + \sum_{i=1}^{s} \gamma_i K^{i-1} r_c \tag{3.48a}$$

where the γ_i-values are obtained by solving the following subsystem of s linear equations

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1s} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2s} \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_{s1} & \alpha_{s2} & \cdots & \alpha_{ss} \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \cdots \\ \gamma_s \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \cdots \\ \beta_s \end{bmatrix} \tag{3.48b}$$

$$\begin{aligned} \alpha_{ij} &= r_c^T K^{i+j-1} r_c = \alpha_{ji} \\ \beta_i &= r_c^T K^{i-1} r_c = \alpha_{io} \end{aligned} \tag{3.48c}$$

The iteration can be derived from error function ϕ_2 by using direction matrices, rather than vectors, for the minimization process (Section 3.1.2). This fact indicates a conceptual similarity between s-step gradient methods and block modifications of linear stationary iterations (Section 3.2.2). For s = 1 the above algorithm becomes identical to that of the steepest descent method, whereas for s = n a direct solution of the system of equations is obtained. Since the algorithm requires the simultaneous storage of s vectors of the form $K^i r_c$, its application is restricted to rather small values of $s \ll n$.

In Ref. 21 it is shown that Kantorovich's method is identical to Version A of the conjugate gradient method, provided the solution process is restarted every s iteration cycles (Chapter 5). Since the latter method yields identical results in recursive form and without practical restrictions on s, no advantage is gained by using the rather uneconomical algorithm of Kantorovich's method.

(2) Khabaza's Method (Ref. 50)

A second s-step gradient method, which could be considered as a "block version" of Krasnoselskii's iteration, was developed by Khabaza (Ref. 50). The computational details of the algorithm differ from those given in Eq. 3.48 only in the definition of the α- and β-coefficients.

$$\alpha_{ij} = r_c^T K^{i+j} r_c = \alpha_{ji}$$
$$\beta_i = r_c^T K^i r_c = \alpha_{io} \tag{3.49}$$

As in the previous case, the iteration corresponds to a particular form of conjugate gradient method (Version B), restarted every s iteration cycles. The practical application of Khabaza's method is, therefore, limited for the same reasons that were mentioned above.

(3) Almost Optimum Steepest Descent Method (Ref. 24,72)

The almost optimum steepest descent method can be derived by applying a linear stationary acceleration (Section 4.2) of the form

$$u_{c+1} = u_c + \omega[I(u_c) - u_c] \tag{3.50}$$

to the original steepest descent algorithm (Table 10). As a result the following computational form is obtained

$$u_{c+1} = u_c + \omega\gamma_c r_c$$
$$\gamma_c = \frac{r_c^T r_c}{r_c^T K r_c} \tag{3.51}$$

ω = constant acceleration factor

The convergence of the iteration process is guaranteed for ω-values in the range from 0.0 to 2.0. Naturally, the same type of acceleration could also be applied to other nonlinear iterations (Ref. 56), although this possibility is not considered here.

(4) Accelerated Steepest Descent Method (Ref. 24)

In order to increase the rate of convergence of the steepest descent method, Forsythe and Motzkin suggested the following nonlinear stationary acceleration procedure (Section 4.3)

$$\begin{aligned} u_{c+1} &= u_c+\gamma_c(u_c-u_{c-p}) \\ \gamma_c &= \frac{r_c^T(u_c-u_{c-p})}{(u_c-u_{c-p})^T K(u_c-u_{c-p})} \\ p &= 2 \end{aligned} \tag{3.52}$$

If the acceleration is applied in intervals of two iteration cycles, the accelerated steepest descent method becomes identical to Kantorovich's 2-step gradient method and, therefore, identical to conjugate gradient Version A, restarted every two cycles. No direct correspondence to a conjugate gradient method can be established for any other cycle interval or value of p.

3.3.2 Numerical Tests

The numerical investigation of basic nonlinear stationary iterations was restricted to the first 5 algorithms listed in Table 10. Gastinel's method was not included in this study since its efficiency, based on experience with method 5, appears to be doubtful. Numerical tests were also carried out with several modified nonlinear iterations, although only the almost optimum steepest descent method is discussed at this point. S-step gradient methods are treated in context with semi-iterative solution methods (Chapter 5), whereas the application of Forsythe's acceleration procedure is described in Section 4.3. As a summary of the numerical results, various observations on the convergence behavior and the performance of the tested methods are described in the following paragraphs.

(1) Steepest Descent, Krasnoselskii, Householder, and Cauchy Iterations

The convergence behavior of the above four iterations exhibits the same characteristics of monotonic linear convergence that were observed for certain linear stationary algorithms (Section 3.2.3). During the initial phase of the solution process the relative error ε_c decreases at a comparatively high, although gradually diminishing rate. For the remainder of the iteration, $\log(\varepsilon_c)$

vs. c becomes a linear relationship whose slope is directly related to the asymptotic rate of convergence. The γ-values of all iterations oscillate from cycle to cycle and approach different asymptotic values for even and odd cycle numbers. It was observed that these asymptotic values of γ_c are related to the eigenvalues of the coefficient matrix by the following expressions

$$\frac{1}{\gamma_c} + \frac{1}{\gamma_{c+1}} = \lambda_{min}(K)+\lambda_{max}(K) \qquad (3.53a)$$

for the method of steepest descent and Krasnoselskii's iteration, whereas

$$\frac{1}{\gamma_c} + \frac{1}{\gamma_{c+1}} = \lambda^2_{min}(K)+\lambda^2_{max}(K) \hat{=} \lambda_{min}(KK)+\lambda_{max}(KK) \qquad (3.53b)$$

in case of Householder's and Cauchy's methods.

The results of Table 11 indicate that the method of steepest descent and Krasnoselskii's iteration have nearly identical rates of convergence. Similar observations can be made for Householder's and Cauchy's iteration, although both methods converge considerably slower than those of the first group. Since both, Householder's as well as Cauchy's iteration are derived from the method of steepest descent by means of Gauss transformations, it can be concluded that these transformations have an identical detrimental effect on the rate of convergence. A comparison with Successive Approximation Version D indicates that for $\omega' = 2.0$ the iteration converges at practically the same rate as the method of steepest descent or Krasnoselskii's iteration (Tables 8 and 11).

(2) Gastinel-Householder Method

The behavior of the Gastinel-Householder iteration differs from that of the previous methods in so far as the convergence is only approximately linear. The relative error ε_c does not decrease strictly monotonically and exhibits variations in its rate of reduction. Similarly, the γ_c-values show a gradual, although irregular decrease during the iteration process.

The convergence of the Gastinel-Householder iteration is usually slower than that of the method of steepest descent (Table 11). However, exceptions may occur for systems of equations, where the elements of the solution vector are of the same magnitude (example A1).

(3) Almost Optimum Steepest Descent Method

In comparison with the method of steepest descent, a drastic change in the nature of convergence can be observed for ω-values less than 1.0 (Fig. 6). Although the relative error ε_c decreases monotonically, its variation with respect to c is characterized by sudden drops which occur at irregular cycle intervals. At the same time, the rate of error reduction becomes very irregular and does not approach any kind of asymptotic value. On the other hand, for acceleration factors greater than 1.0, the iteration retains all the characteristics of monotonic linear convergence that were already observed for the original steepest descent method.

The $m_{0.1}$-values of Table 12 indicate that for ω-values smaller than 1.0 the iteration converges considerably faster than its unaccelerated version. For the test examples a maximum amount of error reduction can be observed for acceleration factors in the vicinity of 0.90. However, due to the irregular nature of convergence a clearly defined optimum value does not exist. For ω-values greater than 1.0 the rate of convergence remains largely unaffected by the magnitude of the acceleration factor.

In summarizing the results of these tests it can be concluded that the almost optimum steepest descent method is the fastest converging algorithm within this particular group of iterative methods. However, the iteration converges noticeably slower than, for instance, block overrelaxation with optimum or near optimum values of ω (Table 7).

3.4 Linear Nonstationary Iterations

3.4.1 Algorithms

Nonstationary iterations are characterized by the fact that the cycle counter c is directly part of the algorithm, either in form of a variable or as the order of a polynomial. Aside from the quantity c, the algorithm may also involve the cycle interval q after which the iteration process is restarted. The general operator for linear nonstationary iterations can, therefore, be written in the form

$$u_{c+1} = I[K,f,u_c,u_{c-1},\ldots,c,q] \tag{3.54}$$

where u_{c+1} is a linear function of the previous approximations. Included in this class of iterative solution procedures are the following individual methods

(1) Lanczos' method

(2) Hypergeometric relaxation

(3) Bellar's method.

In addition, several closely related algorithms are discussed among linear nonstationary accelerations (Section 4.4). The computational details of the above iterations as well as certain explanatory remarks are given in the remainder of this section.

(1) Lanczos' Method (Ref. 21,51,84)

For the purpose of generating suitable starting vectors for a certain type of conjugate gradient method, Lanczos developed the following iterative algorithm

$$g_c = Bg_{c-1} - g_{c-2} + \frac{(1+c)^2}{b_K} r_o$$

$$u_{c+1} = u_o + \frac{4}{(c+2)^2} g_c \tag{3.55a}$$

where $c = 1 \ldots. (q-1)$

$$B = 2I - \frac{4}{b_K} K \tag{3.55b}$$

$$b_K \geq \lambda_{max}(K)$$

In order to start the iteration process, the vector g has to be initialized as follows

$$g_{-1} = 0, \quad g_o = \frac{1}{b_K} r_o \tag{3.55c}$$

A description of the procedure by which Lanczos' iteration can be derived from the theory of orthogonal polynomials is given in Ref. 21 and 51. The second reference also contains a modified version of Lanczos' method which essentially corresponds to a first Gauss transformation of the above algorithm.

As in the case of other nonstationary iterations, the solution process may be restarted after q iteration cycles by using u_q as the new initial approximation. For q = 1 Lanczos' iteration becomes identical to Version D of the Successive Approximation group (Section 3.2.2).

(2) Hypergeometric Relaxation (Ref. 74,75)

The computational scheme for Stiefel's method of hypergeometric relaxation can be presented in the form

$$\begin{aligned} u_1 &= u_o + \frac{1}{\nu_o}\frac{1}{b_K} r_o \\ u_{c+1} &= u_c + \frac{1}{\nu_c}\frac{1}{b_K} r_c + \frac{\mu_c}{\nu_c}(u_c - u_{c-1}) \end{aligned} \tag{3.56a}$$

where $c = 1 \ldots . (q-1)$

$$\mu_c = \frac{1}{4}\left[1-\frac{(\sigma_1-\sigma_2+1)(\sigma_1+\sigma_2+1)}{(2c+\sigma_1+\sigma_2+1)} + \frac{(\sigma_1-\sigma_2)(\sigma_1+\sigma_2)}{(2c+\sigma_1+\sigma_2)}\right]$$

$$\nu_c = \frac{1}{4}\left[1+\frac{(\sigma_1-\sigma_2+1)(\sigma_1+\sigma_2+1)}{(2c+\sigma_1+\sigma_2+1)} - \frac{(\sigma_1-\sigma_2)(\sigma_1+\sigma_2)}{(2c+\sigma_1+\sigma_2+2)}\right] \qquad (3.56b)$$

$$b_K \geq \lambda_{max}(K)$$

Aside from the cycle interval q, the algorithm involves two additional parameters, σ_1 and σ_2, which are subject to the following conditions

$$\sigma_1 > 0, \quad \sigma_2 \geq -\frac{1}{2} \qquad (3.56c)$$

The derivation of the iterative method as well as a description of the effects of σ_1 and σ_2 on its convergence are given by Stiefel in Ref. 74.

(3) Bellar's Method (Ref. 6,84)

In order to improve the efficiency of Lanczos' iteration, Bellar suggested the following modification of the algorithm

$$g_c = Bg_{c-1} - (1-\frac{a_K}{b_K})^2 g_{c-2}+4\frac{\mu_c}{b_K}\, r_o$$

$$u_{c+1} = u_o + \frac{1}{\mu_{c+1}}\, g_c \qquad (3.57a)$$

where $$c = 1....(q-1)$$

$$B = 2\left(1 + \frac{a_K}{b_K}\right) I - \frac{4}{b_K} K$$

$$\mu_c = \left(1 + \sqrt{\frac{a_K}{b_K}}\right)^{2c} + \left(1 - \sqrt{\frac{a_K}{b_K}}\right)^{2c} \qquad (3.57b)$$

$$0 < a_K \leq \lambda_{min}(K)$$

$$b_K \geq \lambda_{max}(K)$$

At the start of the solution process, the vector g has to be initialized in the following form

$$g_{-1} = 0, \quad g_o = \frac{4}{b_K} r_o \qquad (3.57c)$$

The main difference between both nonstationary iterations lies in the fact that Bellar's method involves a lower bound, a_K, for the smallest eigenvalue of the coefficient matrix (Appendix 1).

3.4.2 Numerical Tests

Numerical tests of linear nonstationary iterations were restricted to two of the three algorithms discussed in the previous section. Bellar's method was not included in this study since it involves a numerical quantity which is not available in practice (p. 75). The remaining iterations were tested with two different upper bounds for the maximum eigenvalue of the coefficient matrix: $b_K = \|K\|_1$ and $b_K = \lambda_{max}(K)$. These particular values were chosen since Gershgorin's estimate $\|K\|_1$ represents the least conservative "a priori" bound (Appendix 1), whereas $\lambda_{max}(K)$ constitutes the limiting b_K-value for which convergence is guaranteed.

(1) Lanczos' Method

If the iteration is performed without restart (i.e. for $q = \infty$), the relative error ε_c exhibits cyclic oscillations which resemble a series of convex parabolas with gradually decreasing vertices (Fig. 7). The cycle interval of these oscillations, c_o, remains constant throughout the iteration process. In contrast to observations with other iterative methods, the maximum displacement apparently never exceeds its final value when ε_c passes through a minimum.

If the iteration is restarted in intervals of q cycles, a clearly visible change in behavior can be observed. Essentially, after each restart the log (ε_c) vs. c relationship assumes a form which is very similar to the initial branch of the curve (Fig. 7). The most rapid over-all convergence is obtained if the iteration is restarted after $q = c_o$ iteration cycles (Table 13). Since the increments of the maximum displacement reverse their sign in the vicinity of c_o, it is possible to determine this optimum value of q during the iteration process. However, even with this restarting procedure the over-all rate of convergence could not exceed the average rate of the first c_o iteration cycles.

The numerical tests indicate that the iteration converges somewhat faster for $b_K = \lambda_{max}(K)$ than for $b_K = \|K\|_1$, although the amplitudes of the ε_c-oscillations remain unchanged. The effect of the eigenvalue bound b_K on the convergence behavior could, therefore, be described as that of scaling the c-axis of the log (ε_c) vs. c relationship.

Comparisions between $m_{0.1}$-values for block over-relaxation and Lanczos' method show that latter iteration is less efficient even for optimum values of q (Tables 7 and 13). Therefore, the suggested restart procedure would not be effective enough to make Lanczos' method competitive with some of the faster converging linear stationary iterations.

(2) Hypergeometric Relaxation

The numerical investigation of an iterative method involving three independent parameters q, σ_1, and σ_2 requires a considerable amount of computations if all effects on the convergence behavior should be adequately studied. The situation is particularly complicated by the fact that the parameters σ_1 and σ_2 are of identical nature. Therefore, a rather strong interaction between the effects of these two quantities can be expected (Refs. 74, 75). However, in view of the poor performance of hypergeometric relaxation in preliminary tests for relatively few selected values of σ_1, σ_2, q, and b_K, such an effort is not justifiable. The test results indicate that the rates of convergence of this method are generally low and cannot be compared with those of block overrelaxation (Table 7). For low values of σ_1 and high values of σ_2 the iteration may diverge, particularly if the iteration process is restarted. On the other hand, the convergence for the recommended "safe" values $\sigma_1 = +0.5$ and $\sigma_2 = -0.5$ is slow (Ref. 74). Although these tests give only an incomplete picture of the nature of convergence, they nevertheless illustrate the difficulties encountered in the application of hypergeometric relaxation.

Summarizing the results of this section, it can be concluded that the investigated linear nonstationary iterations do not represent efficient solution procedures.

4. ACCELERATION PROCEDURES

4.1 General Discussion

In a broad sense, acceleration procedures can be defined as algorithms for improving the average rate of convergence of iterative methods. The main difference between iterations and accelerations arises from the fact that the solution of a system of equations can be obtained either by iterations or accelerated iterations, but not by acceleration procedures alone. Despite their different purpose, both types of algorithms often have a similar computational form. Therefore, acceleration procedures can be classified as linear or nonlinear, stationary or nonstationary according to the same criteria that were defined for iterative methods (Section 3.1.1). The formalistic similarity between iterations and accelerations also extends to their derivation, since most of the basic principles discussed in Section 3.1.2 are applicable to both types of algorithms.

Theoretically it is possible to apply any acceleration procedure to any iterative method, although only certain combinations are of practical importance. In many cases acceleration and iteration algorithms can be combined in such a way that they form a computational unit. Combined algorithms of this type are discussed, for instance, among linear stationary iterations (Section 3.2). It is also possible to accelerate the convergence of an iteration by applying two (or even more) acceleration procedures simultaneously. Such combinations are normally restricted to cases where one of the accelerations is linear stationary.

Usually an acceleration procedure involves a number of parameters whose numerical values have to be chosen in such a way that the over-all rate of convergence of the accelerated iteration is maximized. Only in exceptional cases are these parameters pre-set for computational reasons (cf. Section 4.3, Irons-Tuck acceleration).

4.2 Linear Stationary Accelerations

4.2.1 Algorithms

Linear stationary acceleration procedures, which represent the simplest form of accelerations, can be symbolized by the following general operator

$$u_{c+1} = A[u_c, u_{c-1}, \dots, \omega] \qquad (4.1)$$

where u_{c+1} is a linear function of the previous approximations. The parameter ω represents a constant acceleration factor which has to be specified prior to the beginning of the solution process. The linear stationary group of accelerations includes only two algorithms which are of practical importance. As mentioned in Section 3.1.2, both accelerations can be derived by considering the iterative solution process as a time dependent phenomenon, whose analysis is carried out by means of step-wise integration techniques (Refs. 9, 39, 87).

(1) Algorithm I

The most frequently used type of linear stationary accelerations can be written in the form of

$$u_{c+1} = u_c+\omega[I(u_c)-u_c] \tag{4.2}$$

In the above expression, $I(u_c)$ represents any of the stationary iterative algorithms described in Sections 3.2 and 3.3. For single-step versions of linear stationary iterations it is understood that the acceleration is applied after each individual iteration step, not only at the end of a full iterative cycle (p. 38). If the iteration $I(u_c)$ itself converges, that is, if the dominant eigenvalue of the iteration matrix is less than 1.0 in absolute value, the accelerated iteration usually converges for ω-values in the range from 0.0 to 2.0. Specific convergence conditions, however, can only be established in context with a particular iterative algorithm. From Eq. 4.2 it can be seen that the acceleration is suppressed if ω assumes the value of 1.0.

The application of this algorithm to various linear and nonlinear stationary iterations is extensively described in Sections 3.2 and 3.3. Therefore, a separate discussion of the acceleration procedure is not given at this point.

(2) Algorithm II (Faddeev I Acceleration)

In Ref. 21 a large number of linear stationary acceleration procedures are described which all employ the following type of algorithm

$$u_{c+1} = I(u_c)+\omega[I(u_c)-u_{c-1}] \qquad (4.3)$$

The various forms of this acceleration differ

(a) in the definition of their objective, that is in the criterion used to establish the optimum value of ω,

(b) in the way ω_{opt} is related to the eigenvalues of the iteration matrix, and

(c) in the starting procedure for the accelerated iteration.

For the given type of linear equations it is not possible to establish "a priori" bounds for the eigenvalues of the iteration matrix. Therefore, ω has to be considered as purely empirical parameter and any differences in the derivation of its optimum value become immaterial. From previous experience it is known that the way in which a linearly accelerated iteration is started has no significant effect on its over-all convergence (p. 54). Therefore, it was considered adequate to adopt the simplest starting procedure for the present purpose. Its algorithm can be written as

$$\begin{aligned} u_1 &= I(u_o) \\ u_2 &= I(u_1)+\omega[I(u_1)-u_o] \end{aligned} \qquad (4.4)$$

and is equivalent to suppressing the acceleration during the first cycle by setting $\omega = 0$. Two alternative procedures are described in Ref. 21. Under conditions similar to those defined for Algorithm I, the accelerated iteration converges for any ω-value in the range $-1.0 < \omega < +1.0$. However, specific convergence conditions can, again, be established only in context with a particular iterative method. As far as the storage requirements are concerned, the Faddeev I acceleration has the disadvantage of requiring at least one additional vector in comparison with the previous algorithm.

In order to illustrate the implementation of the acceleration procedure, a number of examples are given in the following paragraphs. The application of the Faddeev I acceleration to the iteration

$$I(u_c) = u_c + \frac{2}{a_K+b_K} r_c \quad \begin{array}{l}\text{(Successive}\\ \text{Approximation}\\ \text{Version D)}\end{array} \tag{4.5a}$$

with

$$\omega = \frac{(\sqrt{a_K}-\sqrt{b_K})^2}{(\sqrt{a_K}+\sqrt{b_K})^2}$$

$$0 < a_K \leq \lambda_{min}(K) \tag{4.5b}$$

$$b_K \geq \lambda_{max}(K)$$

leads to the following combined algorithm, attributed to Frankel (Ref. 17)

$$u_{c+1} = u_c - \frac{4}{(\sqrt{a_K}+\sqrt{b_K})^2} r_c + \frac{(\sqrt{a_K}-\sqrt{b_K})^2}{(\sqrt{a_K}+\sqrt{b_K})^2}(u_c-u_{c-1}) \tag{4.5c}$$

Similarly, a particular version of the "Dynamic Relaxation" method (Ref. 39)

$$u_{c+1} = \frac{1}{\left(\frac{\mu_1}{\nu^2} + \frac{\mu_2}{\nu}\right)}\left[\frac{2\mu_1}{\nu^2}u_c-\left(\frac{\mu_1}{\nu^2} - \frac{\mu_2}{\nu}\right)u_{c-1}+r_c\right] \tag{4.6a}$$

is obtained by applying the acceleration procedure to the iteration

$$I(u_c) = u_c + \frac{\nu^2}{2\mu_1} r_c$$

with
$$\omega = \frac{\left(\frac{\mu_1}{\nu^2} - \frac{\mu_2}{\nu}\right)}{\left(\frac{\mu_1}{\nu^2} + \frac{\mu_2}{\nu}\right)} \qquad (4.6b)$$

Various other forms of dynamic relaxation can be derived by using a (diagonal) matrix of acceleration factors instead of single ω-values (Refs. 9, 11, 39).

So far the discussion of linear stationary accelerations covered only two basic types of algorithms. A third type, suggested by Abramov and described in Ref. 21, could be considered a modification of the Faddeev I acceleration. The computational forms of both methods are identical except that during the execution of Abramov's acceleration, ω does not remain constant but may assume two different values: $\omega = 0$ (i.e. no acceleration) and $\omega = 1.0$. Since the sequence in which these ω-values are to be chosen is arbitrary, the procedure as such is of no particular practical value. However, following a suggestion by Faddeev (Ref. 21), an appropriate sequence could be established by computing separate new vectors u_{c+1} for both ω-values and by chosing the one which gives the smaller length of the residual vector r_c. The calculation of these residuals represents, of course, a considerable increase in the number of arithmetic operations. Despite this additional computational effort it is very unlikely that the procedure would yield average rates of convergence which are higher than those for the Faddeev I acceleration with optimum values of ω. Abramov's acceleration procedure was, therefore, not included in the numerical tests.

4.2.2 Numerical Tests

The application of Algorithm I to various linear and nonlinear iterations was extensively described in direct context with these iterative methods (Section 3.2 and 3.3). Numerical tests of linear stationary accelerations were, therefore, restricted to Algorithm II in combination with the Block Gauss-Seidel method. This particular iteration was selected since it exhibits the fastest convergence of all unaccelerated iterative methods (p. 59).

The numerical tests indicate that the nature of convergence of the accelerated iteration is identical to that of block overrelaxation whose algorithm represents a combination of Block Gauss-Seidel and Algorithm I (Section 3.2). By comparing the $m_{0.1}$-values of both iterations (Tables 7 and 14) it can be concluded that not only the nature but also the rate of convergence is nearly the same, provided the following relationship between the ω-values of Algorithms I and II is assumed

$$\omega_I = 1.0+\omega_{II} \qquad (4.7)$$

Relatively minor differences in the performance of both methods occur only in the vicinity of the optimum acceleration factors. The numerical results on the convergence of the iterations thus concur with theoretical findings reported by various investigators (Refs. 9, 11, 82). Although it is unlikely that the accelerating effect of both algorithms will be the same for all iterative methods, it nevertheless can be concluded that the Faddeev I acceleration has no distinct advantage as far as its efficiency is concerned. Taking into account that Algorithm I requires less storage space (two vectors of size n in case of Block Gauss-Seidel), the Faddeev I acceleration has to be considered as less suitable.

4.3 Nonlinear Stationary Accelerations

4.3.1 Algorithms

The general algorithm for nonlinear stationary acceleration procedures can be expressed as

$$u_q = A[K,f,u_L,u_{L-1},\ldots.,p,q] \tag{4.8}$$

where u_q represents a nonlinear function of the previous approximations. Acceleration procedures of this type differ from the remaining algorithms in at least two aspects of their application. Whereas other accelerations are used in alternating order with iterative methods, nonlinear stationary accelerations are normally applied in intervals of several iteration cycles. In addition, the algorithms usually involve approximate vectors u_c which are computed several cycles before the acceleration is carried out. Therefore, it becomes necessary to introduce an auxiliary notation for identifying these previous approximations and for specifying the cycle interval in which the acceleration is repeated. The following three quantities, already contained in the general algorithm of Eq. 4.8, are used for this purpose:

q = cycle length of the acceleration interval, equivalent to the total number of approximate vectors u_c computed during a single acceleration interval, ($q \geq 2$),

L = index of the last iterate of the acceleration interval, equivalent to the total number of iteration cycles carried out during an acceleration interval, ($L = q-1 \geq 1$),

p = cycle interval for identifying previous approximations in relation to u_L ($p \geq 1$).

All three quantities could be considered part of a local cycle counter system which is independent of the global cycle counter c.

The nonlinear stationary group of acceleration procedures includes the following individual methods

(1) Wilson
(2) Forsythe
(3) Aitken
(4) Ishibashi
(5) Dyer I
(6) Milne I
(7) Modified Aitken
(8) Irons-Tuck
(9) Rashid
(10) Dyer II
(11) Milne II

The computational details of these accelerations are given in Table 15. The remainder of this section contains a number of explanatory remarks on the nature and the derivation of each algorithm.

(1) Wilson (Ref. 86)

By rewriting the computational form of the acceleration procedure (Table 15) in the following way

$$u_q = u_L+\overline{\gamma}_L u_L$$

with
$$\bar{\gamma}_L = \frac{r_L^T u_L}{u_L^T K u_L} \tag{4.9}$$

it is possible to show that the algorithm can be derived by minimizing error function ϕ_2 with u_L as direction vector (Section 3.1.2). A modification of the above acceleration with γ_L defined by

$$\gamma_L = \frac{f^T K u_L}{u_L^T K K u_L} \tag{4.10}$$

is obtained if error function ϕ_3 is chosen for the minimization process. Of all nonlinear acceleration procedures, Wilson's algorithm represents the only case in which no more than one previous approximation is required.

(2) <u>Forsythe</u> (Ref. 24,73)

The original algorithm (with $p = 2$) was developed by Forsythe and Motzkin for the acceleration of the steepest descent method (Section 3.3.1), although a similar procedure, called "Pauschalkorrektur" (lump sum correction), was also suggested by Stiefel (Ref. 73). The algorithm can be derived by minimizing error function ϕ_2 with $(u_L - u_{L-p})$ as direction vector (Section 3.1.2). Therefore, Forsythe's acceleration procedure becomes identical to Wilson's algorithm if the vector u_{L-p} is assumed to be zero.

(3) Aitken (Ref. 26,60,84)

Aitken's acceleration procedure is a method for determining the asymptotic limit of a geometrically converging scalar sequence. In applying this procedure to the solution of systems of linear equations, each element $(u)_i^c$ of the solution vector is treated as an independent quantity. Since the convergence of the iterative methods to be accelerated is not geometrical but linear, the extrapolated value $(u)_i^q$ represents, at best, an improved approximation of the limiting value $(u)_i$. A necessary, but not sufficient condition for such an improvement is satisfied if the dominant eigenvalue of the iteration matrix is real. Certain generalizations of Aitken's procedure are also applicable for complex dominant eigenvalues (Refs. 23, 26, 70). However, such generalized algorithms are not suitable for the given purpose since they require an even larger number of intermediate vectors to be stored.

A different type of generalization can be made by applying Aitken's procedure to a sequence of vectors rather than scalars. Essentially, the generalization consists of establishing a vector equivalent of the quotient which defines the individual γ_{i_L}-values of Aitken's acceleration. Several possible forms of such vector extrapolation methods are listed in Table 15 as algorithms 4 through 8. All accelerations of this type are similar to Aitken's procedure in the sense that they become identical if u_c is assumed to be a scalar quantity.

(4) Ishibashi (Ref. 45)

Ishibashi's acceleration procedure, which belongs to the group of vector extrapolation methods, is based on the assumption that all elements of the solution vector converge at approximately the same rate as an arbitrarily selected element $(u)_k^c$. The method requires less storage than Aitken's acceleration since only the k-th element of the vector u_{L-2p} has to be stored.

(5) Dyer I, Dyer II (Ref. 79)

Both acceleration procedures were originally developed by Dyer for the purpose of accelerating the convergence of Kaczmarz' iteration (Section 3.2.2). The first of the two algorithms represents a generalization of Aitken's acceleration and, therefore, is included in the group of vector extrapolation methods. In comparison with other nonlinear stationary accelerations, the Dyer II algorithm requires a considerable amount of computational effort, totalling approximately two matrix-vector products per acceleration interval (Appendix 3).

(6) Modified Aitken

The modified Aitken procedure can be derived from Algorithm 3 by reformulating the γ_{i_L}-values of Aitken's acceleration, using the following definition of the "inverse" of a vector (Ref. 89)

$$V^{-1} = \frac{1}{V^T V} V \qquad (4.11)$$

(7) Irons-Tuck (Ref. 43)

Unlike other nonlinear stationary accelerations, the Irons-Tuck procedure involves no acceleration parameters and is applied in alternating sequence with iterative algorithms. In order to start the solution process, the following initial values for γ and u are suggested in Ref. 43

$$\begin{aligned} &\gamma_o = 0 \\ &u_{-1} = 0, \quad u_{-2} = \infty \end{aligned} \qquad (4.12)$$

The starting procedure has the effect of suppressing the acceleration during the first interval. Since the length of the acceleration interval q is pre-set to a value of 2, the algorithm does not require the separate storage of all previous approximations u_{L-1}, u_{L-2}, and u_{L-3}.

(8) Rashid (Ref. 61)

Rashid's extrapolation procedure is a method for determining the (k+L)-th value of a geometrically converging scalar sequence. As in the case of Aitken's acceleration, each element $(u)_i^c$ of the solution vector is treated as an independent quantity. Both acceleration procedures become, therefore, mathematically identical if the parameter k approaches infinity. Since the convergence of the iterative methods to be accelerated is not geometrical but linear, the extrapolated value $(u)_i^q$ represents, at best, an approximation of the (k+L)-th iterate $(u)_i^{k+L}$.

(9) Milne I, Milne II (Ref. 56)

An approximate solution of a system of linear equations can be improved by means of the following general procedure suggested by Milne (Ref. 56)

$$u_q = \frac{1}{1+\sum_{j=1}^{k}\alpha_j}(u_L+\alpha_1 u_{L-1}+\alpha_2 u_{L-2}+\ldots.+\alpha_k u_{L-k})$$

$$u_q = u_L-\gamma_1(u_L-u_{L-1})-\ldots.-\gamma_k(u_L-u_{L-k}) \qquad (4.13a)$$

with $$\gamma_i = \frac{\alpha_i}{1+\sum_{j=1}^{k}\alpha_j}$$

The coefficients α_i are obtained from a least squares solution of an auxiliary system of equations defined by

$$(u_L-u_{L-1})+\alpha_1(u_{L-1}-u_{L-2})+\alpha_2(u_{L-2}-u_{L-3})+\ldots \\ \ldots+\alpha_k(u_{L-k}-u_{L-k-1}) = 0 \qquad (4.13b)$$

The resulting algorithm bears a strong formalistic resemblance to the computational scheme of s-step gradient methods (Section 3.3.1). In the derivation of Milne's acceleration it is assumed that the eigenvalues of the iteration matrix are real. For k = n the new approximation u_q coincides with the solution vector u, whereas the α_i-values become identical to the coefficients of the characteristic equation of the iteration matrix. Storage requirements make it necessary, though, to restrict the parameter k to relatively small values. The Milne I acceleration, which belongs to the group of vector extrapolation methods, is obtained by setting k equal to 1, whereas a k value of 2 leads to Algorithm 11 of Table 15.

A different form of Milne's acceleration procedure is obtained if the α_i-values are determined directly by calculating the low order coefficients of the characteristic equation of T (Ref. 21). However, this alternative, which can only be applied to total-step versions of linear stationary iterations (Section 3.2.1), was not considered in this investigation.

The derivation of most nonlinear stationary acceleration procedures is based on the assumption that the iterative methods to be accelerated exhibit geometrical or linear convergence. Since the acceleration itself causes a disruption of the convergence process, the parameters p and q have to be chosen in such a way that sets of previous approximations u_L, u_{L-p}, u_{L-2p} are part of the same acceleration interval and, thus, do not overlap. An exception is made by Algorithm 8 (Irons-Tuck acceleration) which was explicitly developed for overlapping sets of previous approximations.

4.3.2 Numerical Tests

All nonlinear stationary acceleration procedures discussed in the previous section were tested numerically in connection with block overrelaxation as well as Successive Approximation Version D (Section 3.2). In addition, Algorithms 1 and 2 were also applied to the almost optimum steepest descent method (Section 3.3). The selection of block overrelaxation as a basic iterative method was based on the fact that the algorithm exhibits the fastest convergence of all linear stationary iterations (p. 59). It is realized that the presence of complex eigenvalues of the iteration matrix may cause numerical instabilities since most acceleration procedures are only applicable to iterations with real $\lambda_i(T)$-values. In order to eliminate this potential source of difficulties, the accelerations were also tested in combination with Successive Approximation Version D, whose iteration matrix is known to have only real eigenvalues.

Because of the large number of acceleration procedures, it was necessary to keep the discussion on the convergence behavior and the performance of the accelerated iterations to a minimum. A relatively detailed description is given only for two of the more promising methods (Wilson's and Forsythe's accelerations), whereas the discussion of the remaining algorithms is more or less restricted to brief comments on their suitability. The following first part of the presentation of test results covers the application of nonlinear stationary accelerations to block overrelaxation.

(1) Wilson

Within the range of ω-values for which the basic iteration exhibits monotonic linear convergence, the application of the acceleration procedure results in a sharp reduction of the relative error ε_c. During intermediate iterations, however, the value of ε_c usually increases, thus giving the $\log(\varepsilon_c)$ vs. c relationship a saw-tooth-like appearance (Fig. 8, 9). For higher values of ω, the convergence of the accelerated iteration becomes irregular, particularly for well-conditioned systems of equations. As the relative error ε_c decreases in magnitude, the γ_L-factor of Wilson's acceleration approaches a value of 1.0.

The average rate of convergence of the solution process is a comparatively smooth and regular function of the acceleration parameters. From the $m_{0.1}$-values of Table 16 it can be concluded that the convergence rate increases for increasing values of ω and q until a poorly defined maximum is reached. For higher values of the acceleration parameters the solution process converges at a lower rate which approaches that of the unaccelerated iteration. The range of ω-q values for which a maximum or near maximum rate of convergence occurs is relatively wide, thus offering a major advantage in comparison with block overrelaxation itself. For the test examples the optimum values were found to be in the neighborhood of $\omega = 1.9$ and $q = 10$.

The results of Table 16 indicate that the accelerated iteration converges significantly faster than block overrelaxation for corresponding values of the acceleration factor. Exceptions may occur for well-conditioned systems of equations if ω is close to its optimum value. The acceleration is particularly effective for ill-conditioned problems where even roughly estimated ω-q values allow a faster convergence than that obtained by block overrelaxation under optimum conditions. Numerical tests also show that a possible modification of Wilson's acceleration procedure (Eq. 4.10) is considerably less efficient than the original algorithm.

(2) Forsythe

As in the case of Wilson's acceleration the convergence behavior of the accelerated iteration is dominated by the ω-factor, while the effect of the parameters p and q is comparatively small. For low values of ω the solution process converges in a smooth and regular way, whereas higher acceleration factors cause a rather irregular form of convergence. In general, the average convergence rate of the accelerated iteration increases for increasing values of ω as well as q and for decreasing values of p. The highest rate is usually obtained for p equal to 1, for comparatively high values of q, and for ω-values which are smaller than the optimum value of the basic iteration. However, these generalizations describe only over-all trends and may not hold for certain ω-p-q combinations.

For optimum values of the acceleration parameters the solution process converges rather fast, especially for well-conditioned systems of equations. Under these conditions the convergence rate can be as high as or even higher than that of Wilson's acceleration. However, the optimum range of parameters is usually very narrow and even small deviations may cause a drastic decrease in the rate of convergence. In this respect Forsythe's acceleration compares unfavorably with Wilson's algorithm, where a near maximum convergence is obtained for a considerably wider range of ω-q values.

(3) Aitken

Numerical tests indicate that divergence or a very irregular form of convergence at a low average rate may occur for a wide range of ω-values. This behavior can frequently be observed after an initial period of smooth and regular convergence during which the acceleration is comparatively ineffective. Because of the numerical instabilities, the application of Aitken's acceleration to block overrelaxation cannot be considered suitable.

(4) Ishibashi

For high values of ω the accelerated iteration exhibits a very irregular, slow convergence which is characterized by frequent changes in the magnitude of $(u)^c_{max}$. On the other hand, for low ω-values the solution process may diverge, particularly for low values of q and high values of p. If convergence occurs for latter range of ω-values, the average rate may be relatively high for optimum combinations of p and q. However, in view of the numerical instabilities, the combination of Ishibashi's acceleration procedure with block overrelaxation has to be considered unsuitable. In all numerical tests the quantity $(u)^c_k$, used for calculating the acceleration factor γ, was assumed to be identical to the maximum nodal point displacement $(u)^c_{max}$.

(5) Dyer I, Milne I, Modified Aitken

The convergence of any of these three vector-extrapolation methods shows a similar behavior as that observed for Aitken's acceleration. A normally smooth and regular convergence occurs for low values of ω, whereas higher ω-values cause irregular oscillations during which $(u)^c_{max}$ may undergo frequent changes in sign and magnitude. Numerical instabilities of this type are most likely to occur for the Dyer I acceleration and may result in divergence of the solution process. The average rate of convergence of any of these methods is a very irregular function of the parameters ω, p, and q. The numerical tests indicate that the modified Aitken acceleration is not only the most stable but also the most efficient of these algorithms. However, with very few exceptions, all three acceleration procedures converge noticeably slower than Wilson's acceleration for corresponding values of ω and q.

(6) Irons-Tuck

In a relatively large number of numerical tests it was observed that the convergence of the accelerated iteration stagnates at a certain point without probably ever regaining any measurable amount of error-reduction. This type of behavior may occur for any value of ω and for any condition of the system of equations, but it is almost certain to occur for higher ω-values. Although the average rate of convergence is a rather irregular function of the acceleration factor, its maximum seems to occur for ω-values close to or below 1.0. Even if the convergence does not stagnate, Wilson's acceleration is considerably more efficient than the Irons-Tuck acceleration in combination with block-overrelaxation.

(7) Rashid

From the nature of Rashid's acceleration procedure (p. 94) it can be concluded that a maximum rate of convergence will be obtained for high values of k, low values of q, and for ω-values close to ω_{opt} of the basic iteration. Numerical tests indicate, however, that for these ω-k-q values the accelerated iteration frequently diverges, whereas other, less optimum ω-k-q combinations cause a rather slow convergence. Therefore, Rashid's algorithm cannot be considered suitable for the acceleration of block overrelaxation.

(8) Dyer II

Although the accelerated iteration does not seem to diverge, its nature of convergence is very irregular, particularly for well-conditioned systems of equations. It may occur that the acceleration consistently has a detrimental effect on the convergence, such that the "accelerated" iteration actually converges slower than its unaccelerated form. Under certain conditions the method may converge to a "wrong" solution in the sense that the acceleration exactly off-sets the amount of error reduction achieved in the intermediate iterations. In none of the test examples does the performance of the Dyer II acceleration come even close to that of Wilson's acceleration for corresponding values of ω and q.

(9) Milne II

The nature of convergence of the accelerated iteration is essentially the same as that described for the Milne I algorithm (p. 101). In comparing these two methods, the Milne II acceleration procedure appears to be less susceptible to numerical instabilities and practically always exhibits a faster convergence for identical values of ω, p, and q. However, the average rate is considerably lower than that obtained by Wilson's acceleration except for well-conditioned problems and low values of ω.

The second part of the presentation of test results covers the application of nonlinear stationary accelerations to Version D of the Successive Approximation group. From preliminary numerical tests it was found that the convergence behavior of the accelerated iterations is essentially the same as that described previously for low values of ω. In particular, the same type of numerical instabilities occur for certain accelerations (Aitken, Ishibashi, Rashid), indicating that they are not caused by complex eigenvalues of the iteration matrix in the case of block overrelaxation (p. 97). Wilson's and Forsythe's acceleration procedures are, again, found to be the most efficient algorithms, although the average rates of convergence are significantly lower than those obtained in the previous case.

In view of the discouraging results of these initial tests, a detailed investigation of acceleration procedures applied to Successive Approximation Version D was not carried out.

In a third group of numerical tests, Forsythe's and Wilson's acceleration procedures were studied in connection with the almost optimum steepest descent method. As mentioned in Section 3.3.1, Forsythe's acceleration (with $p = 2$ and $q = 3$) applied to the method of steepest descent ($\omega = 1.0$) is equivalent to Kantorovich's 2-step gradient method and, therefore, equivalent to a certain type of conjugate gradient algorithm. Numerical tests show that this particular combination of ω-p-q values results in rates of convergence which are only insignificantly higher than those of the unaccelerated steepest descent method. For $\omega = 1.0$ better results are obtained by selecting q-values greater than 3 and in some cases by choosing small, even p-values other than 2. A choice of odd p-values, however, has a detrimental effect on the rate of convergence. In particular, if p is set equal to 1, the acceleration procedure leaves the approximate solution vector u_L unchanged (Ref. 24).

If Forsythe's acceleration is applied to the almost optimum steepest descent method (i.e. for $\omega < 1.0$), the average rate of convergence becomes a very irregular function of the parameters ω, p, and q. It occurs frequently that the accelerated iteration converges slower than its unaccelerated form, and rarely is the rate of convergence higher than that obtained for $\omega = 1.0$ and a proper choice of p-q values.

Noticeably better results can be obtained by applying Wilson's acceleration to the almost optimum steepest descent method. The combined algorithm has the advantage that its average rate of convergence is a rather smooth and regular function of the parameters ω and q. However, a comparison with results obtained by applying the same acceleration to block overrelaxation indicates that the latter solution process is more efficient.

In summarizing the results of these numerical tests it can be concluded that Wilson's acceleration in combination with block overrelaxation offers a number of advantages which, as a whole, make this procedure the only one suitable for general application:

(a) Convergence occurs for any admissible value of the parameters ω and q,

(b) the accelerated iteration exhibits a comparatively fast convergence, and

(c) the optimum range of ω-q values is relatively wide.

A more detailed study on the performance of the accelerated iteration is described in Part II of this investigation.

4.4 Linear Nonstationary Accelerations

4.4.1 Algorithms

Nonstationary accelerations as well as iterations are characterized by the fact that the cycle counter c is directly part of the algorithm, either in the form of a variable or as the order of a polynomial. Aside from c, the algorithm may also contain the parameter q which represents the cycle interval for restarting the solution process. The general operator for linear nonstationary accelerations can, therefore, be written in the form

$$u_{c+1} = A[u_c, u_{c-1}, \ldots., c, q,] \qquad (4.14)$$

Included in this class of acceleration procedures are the following individual methods

(1) First order Chebyshev acceleration,

(2) Second order Chebyshev acceleration,

(3) Stiefel's acceleration, and

(4) Faddeev II acceleration.

Several closely related procedures are discussed in Section 3.4 among linear nonstationary iterations. Basically, both types of algorithms differ only in so far as nonstationary accelerations could be applied to any suitable iterative method, whereas nonstationary iterations are based on a particular linear stationary algorithm. Attempts to reformulate latter methods as acceleration procedures were not made since this generalization would have presented certain notational difficulties. The derivation of linear nonstationary acceleration procedures from the theory of orthogonal polynomials (Section 3.1.2) was investigated independently by Faddeev and Stiefel (Refs. 21, 74, 75). The computational details of the above algorithms as well as a number of explanations are given in the following paragraphs.

(1) First Order Chebyshev Acceleration (Ref. 21,84)

The first order Chebyshev acceleration, named after its connection, with Chebyshev polynomials, could be considered as a nonstationary counterpart of a certain linear stationary acceleration (Algorithm 1, Section 4.2). Its computational procedure can be written as

$$u_{c+1} = u_c+\gamma_c[I(u_c)-u_c] \tag{4.15a}$$

where $c = 0,1,\dots(q-1)$

$$\gamma_c = \frac{2}{2-(a_T+b_T)+(a_T-b_T)\cos\left(\frac{2c+1}{2q}\pi\right)}$$

$$-1 < a_T \leq \lambda_{min}(T) \qquad (4.15b)$$

$$+1 > b_T \geq \lambda_{max}(T)$$

The convergence of the acceleration procedure is only guaranteed if the eigenvalues of the iteration matrix, $\lambda_i(T)$, are real. The algorithm represents the only case in which the cycle interval q must be specified in advance since the parameter is directly used in the determination of the γ_c-factors. After executing a total of q iteration cycles, the algorithm may be restarted using u_q as the new initial approximation. However, it is necessary to carry out the solution process in such a way that the total number of iteration cycles is an integer multiple of q. Otherwise no guarantee can be given that the resulting vector u_c is a "good" approximation of the solution vector (Ref. 74). The sequence in which the quantities γ_c are used is immaterial since the vector u_q, except for roundoff errors, is not affected by this order. Under certain conditions the γ_c-factors may become very large, causing a breakdown of the solution process due to error accumulation.

The application of the first order Chebyshev acceleration to Version A of the Successive Approximation group (Section 3.2)

$$I(u_c) = u_c + r_c \tag{4.16a}$$

leads to the following accelerated iteration attributed to Richardson (Refs. 17, 84, 91)

$$u_{c+1} = u_c + \gamma_c r_c$$

where $c = 0,1,\ldots.(q-1)$

$$\gamma_c = \frac{2}{(a_K+b_K)+(a_K-b_K)\cos\left(\frac{2c+1}{2q}\pi\right)} \tag{4.16b}$$

$$0 < a_K \leq \lambda_{min}(K)$$

$$b_K \geq \lambda_{max}(K)$$

(2) Second Order Chebyshev Acceleration (Ref. 71,84)

An alternative to the previous algorithm is given by the second order Chebyshev acceleration which offers a number of computational advantages while retaining the basic character of the original algorithm. Its computational procedure can be written in the form

$$u_1 = \gamma_o \left[\frac{2}{(b_T-a_T)} I(u_o) - \frac{(b_T+a_T)}{(b_T-a_T)} u_o \right]$$

$$u_{c+1} = \gamma_c \left[\frac{4}{(b_T-a_T)} I(u_c) - \frac{2(b_T+a_T)}{(b_T-a_T)} u_c - \gamma_{c-1} u_{c-1} \right] \qquad (4.17a)$$

where $c = 1,2,\ldots.(q-1)$

$$\gamma_c = \frac{C_c(\alpha)}{C_{c+1}(\alpha)}$$

$C_c(\alpha)$ = Chebyshev polynomial of the first kind

$$\alpha = \frac{2-(b_T+a_T)}{(b_T-a_T)} \qquad (4.17b)$$

$$-1 < a_T \leq \lambda_{min}(T)$$

$$+1 > b_T \geq \lambda_{max}(T)$$

The second order Chebyshev acceleration has the advantage that numerical instabilities as described for Algorithm 1 will not arise. In addition, the solution process may be stopped for any value of c since all intermediate vectors u_c represent "good" approximations of the solution vector (Ref. 74). After executing a certain number of iteration cycles, q, the algorithm may be restarted by using u_q as the new initial approximation. The convergence of the second order Chebyshev acceleration is assured under the same conditions mentioned for the previous algorithm.

If the eigenvalues of the iteration matrix are contained in the range

$$-\beta \leq \lambda_i(T) \leq +\beta < 1 \qquad (4.18a)$$

the algorithm can be rewritten in the following simpler form (Refs. 21, 39, 82)

$$u_1 = I(u_o)$$

$$u_{c+1} = I(u_c)+\gamma_c[I(u_c)-u_{c-1}] \qquad (4.18b)$$

where $$\gamma_c = \frac{C_{c-1}(1/\beta)}{C_{c+1}(1/\beta)}$$

$$c = 1,2,\ldots.(q-1)$$

Several other modifications of the second order Chebyshev acceleration are described, for instance, in Ref. 17 and 75. If the γ_c-factors of Eq. 4.18b are assumed to be constant throughout the solution process, a degenerate form of Chebyshev acceleration (Refs. 39, 82) is obtained which corresponds to the Faddeev I algorithm of Section 4.2.

(3) Stiefel's Acceleration (Ref. 21,74)

A second nonstationary counterpart of the Faddeev I acceleration is given by the following acceleration procedure suggested by Stiefel

$$u_{c+1} = I(u_c)+\gamma_c[I(u_c)-u_{c-1}]$$

where $c = 0,1,\ldots.(q-1)$ (4.19)

$$\gamma_c = \frac{c}{c+2}$$

The above algorithm is guaranteed to converge if the eigenvalues of the iteration matrix are real and less than 1.0 in absolute value. As in the case of the second order Chebyshev acceleration, the solution process may be restarted after a certain number of iteration cycles, q, by using u_q as the new initial approximation.

(4) Faddeev II Acceleration (Ref. 21)

Based on the same principles that were used in the derivation of Algorithm 3, Faddeev developed the following acceleration procedure

$$u_{c+1} = \frac{(c+1)(2c+3)}{(c+2)^2} I(u_c) + \frac{(c+1)}{(2c+1)(c+2)^2} u_c - \frac{(2c+3)c^2}{(2c+1)(c+2)^2} u_{c-1} \qquad (4.20)$$

where $c = 0,1,....(q-1)$

Because of the similar nature of both algorithms, the remarks on the convergence and execution of Stiefel's acceleration also apply to the above method.

4.4.2 Numerical Tests

Among the four nonstationary acceleration procedures discussed in the previous section, the first two algorithms (first and second order Chebyshev acceleration) can be applied only if non-trivial upper and lower bounds for the eigenvalues of the iteration matrix are known. Since these quantities are not available in practice, the investigation was restricted to Algorithms 3 and 4, which do not require knowledge of such bounds. Both acceleration procedures were tested in combination with block overrelaxation as well as Successive Approximation Version D. The reasons for selecting these particular iterations are identical to those discussed in connection with nonlinear stationary accelerations (Section 4.3.2).

Numerical tests with Algorithms 3 and 4 applied to block overrelaxation indicate that the solution process diverges except for low ω-values in the vicinity of 1.0. The numerical instabilities can be attributed to the fact that the iteration matrix for block overrelaxation has complex eigenvalues for higher acceleration factors. Since the "a priori" determination of safe ω-values is not possible for the given type of linear equations, the application of nonstationary acceleration procedures to block overrelaxation has to be considered as unsuitable.

Instabilities of the solution process do not occur if the accelerations are used in connection with Successive Approximation Version D, whose iteration matrix has only real eigenvalues. The numerical tests with Algorithms 3 and 4 indicate that the convergence behavior of the accelerated iterations is nearly identical to that of Lanczos' method described in Section 3.4.2. In all cases the relative error ε_c exhibits cyclic oscillations whose cycle interval c_o is constant throughout the solution process. Moreover, the acceleration parameters ω and q affect the convergence in a similar way as b_K and q affect the course of Lanczos' iteration. The only major difference arises in the behavior of $(u)^c_{max}$, which in the case of the two acceleration procedures oscillates about its final value.

Therefore, a restarting procedure similar to that suggested for Lanczos' method cannot be applied to the accelerated iterations.

The numerical tests indicate that Stiefel's acceleration procedure (Table 17) converges somewhat faster than the Faddeev II acceleration, although a clear assessment of the performance can only be made if q is smaller than the oscillation interval c_o. For $\omega = 1/\lambda_{max}(K)$ the rates of convergence obtained by Algorithm 3 are nearly identical to those of Lanczos' method with $b_K = \lambda_{max}(K)$ (Table 13). However, in comparison with some of the more efficient linear stationary algorithms it has to be concluded that the accelerated iterations do not converge at a sufficiently high rate.

5. SEMI-ITERATIVE METHODS

5.1 Algorithms

Among the various methods for solving systems of linear equations, semi-iterative methods (or conjugate gradient methods as they are frequently called) play a unique role in the sense that they combine features of direct as well as iterative solution procedures (Refs. 5, 21, 38, 84). In the absence of roundoff errors, semi-iterative algorithms yield the solution of a system of equations within a finite number of numerical operations and, therefore, exhibit one of the most important characteristics of direct methods. The solution process is carried out in the form of procedural steps ("iteration cycles") whose maximum number corresponds to n, the size of the system of equations, or more precisely to n_λ, the total number of independent eigenvalues $\lambda_i(K)$ (Ref. 51). Theoretically, semi-iterative methods can also be used for finding the inverse of a matrix and for treating multiple load vectors in a similar efficient way as direct solution procedures.

However, fundamental differences between semi-iterative and direct methods exist in various other aspects of their application. The computational scheme of conjugate gradient methods strongly resembles that of nonlinear stationary iterations since it involves only matrix-vector products. The storage requirements of these methods are, therefore, essentially restricted to the non-zero elements of the coefficient matrix plus a certain number of additional vectors (p. 9). Since conjugate gradient methods have similar characteristics as total-step iterations (p. 34), the global stiffness matrix does not necessarily have to be available in assembled form. As other iterative algorithms, conjugate gradient methods require an initial approximation u_o which is continuously improved during the course of the solution process. The accuracy of the solution vector, therefore, depends on the amount of computational effort. In the presence of roundoff errors, the solution of a system of equations may not be obtained within n algorithm steps. In this case, the same computational process can be simply continued until a sufficient amount of error reduction is achieved. Depending on the load vector f and the initial approximation u_o, it may also occur that the solution is obtained within less than n_λ cycles.

Conjugate gradient methods can be derived from a minimization of error functions (Section 3.1.2) by using direction vectors which satisfy certain orthogonality conditions. These vectors are determined recursively during the course of the solution by means of a Gram-Schmidt orthogonalization process (Refs. 21, 38, 84). Since their derivation is based on identical principles, conjugate gradient algorithms belong to the larger group of conjugate directions methods, which also include most of the direct solution procedures. The specific nature of conjugate gradient methods is determined by the fact that the direction vectors are related to the residual vector r_c. Direct methods, such as Gauss elimination, are obtained if the unit vectors e_i are used as the basis of the orthogonalization procedure. The derivation of conjugate direction methods can be interpreted geometrically as a process of finding the center of an n-dimensional ellipsoid $\phi(u_c)$ = const. by a successive reduction of the number of its dimensions (Refs. 21, 38, 84).

As shown by Stiefel (Refs. 74, 75), conjugate gradient methods could also be derived from the theory of orthogonal polynomials by using a specific type of discontinuous weight function for the minimization process. From this point of view, the derivation of conjugate gradient methods resembles that of nonstationary iterations.

The most commonly used conjugate gradient algorithm, designated as Version A, can be expressed in the following form (Refs. 5, 21, 38, 84)

$$u_{c+1} = u_c+\alpha_c v_c$$

$$r_{c+1} = r_c-\alpha_c K v_c \qquad (5.1a)$$

$$v_{c+1} = r_{c+1}+\beta_c v_c$$

where

$$\alpha_c = \frac{r_c^T r_c}{v_c^T K v_c}$$

$$\beta_c = \frac{r_{c+1}^T r_{c+1}}{r_c r_c} \qquad (5.1b)$$

$$v_o = r_o = f-Ku_o$$

The algorithm could be derived by minimizing error function ϕ_2 using K-orthogonal direction vectors v_c. The method of derivation implies that error function $\phi_2(u_c)$ monotonically decreases during the solution process. Various properties of the above algorithm as well as numerous relationships between the vectors u_c, r_c, and v_c are described in detail by Hestenes and Stiefel (Ref. 38). Since a recursive evaluation of the residual vector r_c is used in the above expressions, the standard algorithm requires only one matrix-vector product per iteration cycle (Appendix 3). If the approximate solution vector u_c approaches its exact value, the denominators of the coefficients α_c and β_c become zero. In the application of the algorithm it is, therefore, necessary to test the possible occurence of this condition.

A second form of conjugate gradient method, designated as Version B, can be written in the form (Refs. 17, 53, 74, 75)

$$u_{c+1} = u_c+\alpha_c v_c$$

$$r_{c+1} = r_c-\alpha_c K v_c \tag{5.2a}$$

$$v_{c+1} = r_{c+1}+\beta_c v_c$$

where

$$\alpha_c = \frac{r_c^T K r_c}{v_c^T K K v_c}$$

$$\beta_c = \frac{r_{c+1}^T K r_{c+1}}{r_c^T K r_c} \tag{5.2b}$$

$$v_o = r_o = f-Ku_o$$

Essentially, both computational procedures differ only in the definition of their α_c- and β_c-coefficients. The algorithm of Version B can be derived by using KK-orthogonal direction vectors v_c for the minimization of error function ϕ_3. Consequently, the magnitude of

$$\phi_3(u_c) = r_c^T r_c \tag{5.3}$$

monotonically decreases during the course of the solution. Although the algorithm of Eq. 5.2 involves a recursive evaluation of the residual vector r_c, the solution process requires two matrix-vector products per iteration cycle. However, the computational effort can be reduced to one matrix-vector product if a slightly modified form of the algorithm is used, which requires the storage of two additional vectors (Ref. 53). As far as the termination of the solution process is concerned, similar remarks as those made for Version A apply to the above algorithm. In the original monograph on conjugate gradient methods (Ref. 38) it is implied, although not explicitly stated, that the following relationships exist between the approximate solution vectors u_c^A of Version A and u_c^B of Version B

$$u_{c+1}^B = \frac{1}{\gamma_{c+1}}(u_{c+1}^A+\beta_c^A \gamma_c u_c^B)$$

$$r_{c+1}^B = \frac{1}{\gamma_{c+1}} v_{c+1}^A \qquad (5.4)$$

where

$$\gamma_{c+1} = 1+\beta_c^A \gamma_c$$

$$\gamma_o = 1$$

The equations indicate that the residual vector r_c^B of Version B can be calculated from numerical quantities of Version A alone (cf. Chapter 8).

Because of various orthogonality conditions, the standard algorithms of Eqs. 5.1 and 5.2 can be expressed in a large number of different ways. Several of these alternative formulations are given in Ref. 38 (cf. Section 7.2), whereas other possible modifications, involving a recursive evaluation of α_c- and β_c-like coefficients, are described in Refs. 17, 74, and 75. The main characteristic of these algorithm versions is that the sequence of approximate solution vectors u_c, except for roundoff errors, is not affected by the modifications.

However, different sets of approximate vectors are obtained if Gauss transformations are applied to the conjugate gradient algorithms (Refs. 21, 37) or if the minimization process is carried out with other types of error functions (Ref. 37). A similar change in the nature of the solution process may result if different metrics H are used in the H-orthogonalization of the direction vector v_c (Ref. 37) and if the formulation of algorithms involves other orthogonality conditions (Refs. 8, 21, 51). The primary purpose of these modifications is to extend the applicability of semi-iterative solution methods to systems of equations with more general coefficient matrices. Among the various possibilities, Gauss transformations represent the most commonly used form of generalization. However, based on previous experience with these transformations (Sections 3.2 and 3.3), generalized conjugate gradient methods of this type were not included in the investigation.

In Section 3.3.1 it was shown that s-step gradient methods, which can be considered as block modifications of nonlinear stationary iterations, are closely related to semi-iterative solution procedures. In essence, conjugate gradient algorithms, restarted every s iteration cycles, represent a recursive form of the computational schemes described in Section 3.3. Specifically, the restarted form of Version A is identical to Kantorovich's s-step gradient method, whereas Version B corresponds to Khabaza's algorithm. If s assumes a value of 1, conjugate gradient Version A becomes identical to the steepest descent method, whereas Version B coincides with Krasnoselskii's iteration. In comparison with the computational procedures of Section 3.3.1, conjugate gradient methods have the advantage that their algorithms are considerably less complicated and that the choice of s-values is not restricted by storage limitations. As long as the restarting parameter s is selected in such a way that the solution is not obtained with the first interval of s cycles, conjugate gradient algorithms retain all the characteristics of iterative solution procedures.

5.2 Numerical Tests

The fact that semi-iterative methods have the character of n-step algorithms is an indication that their convergence behavior is affected not only by the condition but also by the size of the systems of equations. It is, therefore, not possible to judge the performance of conjugate gradient methods by the relatively small size test examples which are used in this comparative study (Section 2.1). In order to give a description of the general nature of their convergence, semi-iterative solution methods were, nevertheless, applied to the same systems of equations. It is realized, however, that general conclusions on the preferability of iterative or semi-iterative methods cannot be drawn from these test examples, since their small size is likely to favor conjugate gradient methods.

The numerical tests indicate that the condition of the systems of equations has a noticeable effect on the convergence behavior of conjugate gradient methods. For ill-conditioned problems (examples A3 and A4) it can be observed that after an initial period of relatively little error reduction, ε_c decreases drastically if the cycle counter c approaches the value of n (Fig. 10). Within a relatively few cycles the maximum element of the approximate solution vector $(u)^c_{max}$ reaches its final value without exceeding this particular quantity. The solution process, therefore, does not exhibit any signs of linear convergence. Under certain conditions an abrupt reduction of the relative error ε_c may also occur for c-values which are considerably smaller than n (example A5).

For well-conditioned systems of equations (examples A2 and A6), the relative error ε_c decreases in a less abrupt, although irregular form. For problems of this type a noticeable amount of error reduction can already be observed during the initial phase of the solution process. However, as in the previous case, the convergence behavior does not have the characteristics of linear convergence.

The $m_{0.1}$-values of Table 18 indicate that both conjugate gradient methods converge very rapidly for all test examples. With only one exception (example A3) the required number of cycles is less than or equal to the theoretical maximum value of n. In other words, roundoff errors have an effect on the convergence only for the relatively ill-conditioned example A3. For well-conditioned systems of equations the total number of iteration cycles which are necessary to obtain a sufficiently good approximation of the solution vector is considerably smaller than n. In general, Version B of the conjugate gradient methods converges consistently slower than Version A, although the differences in the corresponding $m_{0.1}$-values are relatively small (Table 18). In comparison with iterative solution methods, the test results clearly show that both conjugate gradient methods converge significantly faster than even the most efficient iterative algorithm (Table 16). However, the relatively small size of the test examples does not allow to draw general conclusions on the relative efficiency of both types of methods. Additional numerical tests with semi-iterative solution procedures are, therefore, included in part II of this dissertation.

A completely different type of convergence behavior can be observed if the conjugate gradient algorithms are used as s-step gradient methods, that is, if the solution process is restarted after a certain number of iteration cycles. As long as the restarting interval s is smaller than the number of iteration cycles, c_{con}, for which the unrestarted solution process converges abruptly, s-step gradient methods retain the characteristics of linear convergence (Fig. 10). This is illustrated by the fact that the logarithm of the relative errors ε_s, ε_{2s}, ε_{3s}, ... approaches a linear relationship with respect to c. Certain irregularities occur if the parameter s approaches the value of c_{con} (Fig. 10), whereas virtually no differences between s-step gradient and conjugate gradient methods exist for higher values of s.

The numerical tests with s-step gradient methods indicate that the rates of convergence of the solution process are comparatively low, unless large restarting intervals are chosen (Table 19). In particular, 2-step gradient methods converge only twice as fast as the corresponding nonlinear stationary iterations (Table 11). However, for increasing values of the parameter s, the rates of convergence rapidly increase until they become identical to those of conjugate gradient methods. The test results clearly illustrate that for the given systems of equations the restarting process does not have a beneficial effect on the performance of the solution procedures.

6. CONCLUSIONS

From the results of the comparative study of various types of iterative and semi-iterative solution procedures it is possible to draw the following conclusions:

(1) Among stationary iterations, the Block Gauss-Seidel method (Section 3.2) represents the most efficient unaccelerated algorithm, whereas its linearly accelerated form, block overrelaxation (Section 3.2), exhibits the best over-all performance among linearly accelerated iterations. Because of the relatively small block size, only minor differences exist between the block and point versions of both iterative methods. Rates of convergence which are of the same magnitude as those of block overrelaxation can also be obtained by applying a different linear stationary acceleration (Faddeev I, Section 4.2) to the Block Gauss-Seidel method. However, the combined algorithm is less suitable since it requires the storage of additional vectors.

(2) The most efficient iterative solution procedure is obtained by applying Wilson's acceleration to the

block overrelaxation method. In comparison with other iterations the algorithm has the advantage that the optimum range of acceleration parameters is relatively wide. For the test examples, near maximum rates of convergence were observed for ω-q-values in the neighborhood of $\omega = 1.90$ and $q = 10$, although somewhat higher values can be expected for more ill-conditioned problems. Unlike other nonlinear accelerations, Wilson's algorithm does not require the storage of previous approximations of the solution vector. The numerical tests also indicate that the combination of nonlinear stationary acceleration procedures with iterative methods, whose T-matrices have only real eigenvalues, results in less efficient algorithms.

(3) Among nonstationary solution procedures, only Lanczos' iteration (Section 3.4) is of certain practical value since its optimum restarting interval q_{opt} can be determined in a relatively simple way. Under optimum conditions, the performance of the iteration is comparable to that of block overrelaxation, although the algorithm is less efficient than block overrelaxation in combination with Wilson's acceleration.

(4) Numerical tests with linear and nonlinear stationary iterations indicate that the use of Gauss transforma-

tions (Section 3.1.2) for the derivation of new iterative algorithms has a strongly detrimental effect on their performance.

(5) For the given test examples, conjugate gradient methods (Chapter 5) converge significantly faster than any iterative solution procedure. However, due to the relatively small size of the systems of equations, a fair comparison of the performance of both types of methods is not possible.

From the results of this comparative study it can, therefore, be concluded that the total number of potentially useful algorithms is reduced to three, namely block overrelaxation in combination with Wilson's acceleration as well as two versions of the conjugate gradient method. In order to determine which of these algorithms exhibits the best overall performance, additional numerical tests are included in part II of this dissertation.

PART II

APPLICATION OF ITERATIVE SOLUTION METHODS

7. SELECTION OF SOLUTION METHOD

7.1 Additional Numerical Tests

As a result of the first part of the investigation, the search for the most efficient iterative or semi-iterative method for solving systems of linear equations can be restricted to the following three algorithms

(1) conjugate gradient Version A,

(2) conjugate gradient Version B,

(3) block overrelaxation in combination with Wilson's acceleration.

The objective of the additional numerical tests described in this section is to determine which one of the above three algorithms represents the most suitable solution procedure. The comparative study is complicated by the fact that the conjugate gradient methods differ from the third algorithm in certain basic aspects of their nature. As described in Chapter 5, the rate of convergence of conjugate gradient methods is primarily affected by roundoff errors as well as by the size of the system of equations. The amount of error accumulation is, in turn, influenced by the word length of the computer and by the condition of the coefficient matrix. Iterative methods, on the other hand, are characterized by the fact that their performance does not depend on the value of n, whereas roundoff errors generally have only an insignificant effect on their convergence (p. 145). The spectral radius of the iteration matrix and, therefore, the rate of convergence are, above all, affected by the condition of the system of equations. Although various other factors may also play a role, the situation remains essentially unchanged if nonlinear accelerations are applied to the iterative process. Therefore, the convergence rates of all three solution procedures are dominated by the P-condition number of the coefficient matrix, although fundamental differences exist in the way in which this parameter affects the performance of iterative and semi-iterative methods. In order to facilitate on equitable comparison of the performance of both types of solution methods, the additional numerical tests of this section are carried out with two larger and relatively ill-conditioned systems of equations (Appendix 2). Both test examples contain a parameter κ which allows giving the coefficient matrices any arbitrary degree of ill-conditioning.

The numerical tests with conjugate gradient Version A indicate that the convergence behavior is very similar to that described in Chapter 5 for ill-conditioned systems of equations. During an initial period whose length increases as more ill-conditioned the problem becomes, a relatively small amount of error reduction is achieved. The initial phase is followed by an abrupt decrease in the magnitude of the relative error ε_c within a small number of iteration cycles (Fig. 11). A more gradual, but irregular form of convergence can only be observed for very ill-conditioned problems ($\kappa \geq 10^3$, Fig. 11). Because of the abrupt convergence, higher accuracies of the approximate solution vector are obtained with comparatively little computational effort. The $m_{0.1}$-values of Table 20 illustrate that for a wide range of κ-values the number of iteration cycles, required to achieve convergence, exceeds the theoretical limit of n. In other words, roundoff errors have a detrimental effect on the convergence for comparatively small values of the P_1-condition number (Appendix 2). The correlation between the rate of convergence and the condition of the coefficient matrix is illustrated by the fact that both the $m_{0.1}$-values as well as the magnitude of $\log(P_1)$ vary as an approximately linear function of $\log(\kappa)$. Since nearly identical rates of convergence are obtained for $\kappa = 10^{+\alpha}$ and $\kappa = 10^{-\alpha}$ of example B1, it can be concluded that removable and non-removable types of ill-conditioning have essentially the same effect on the convergence (Appendix 2).

By comparing the test results of both versions of the conjugate gradient method it can be observed that the solution procedures exhibit an identical convergence behavior. Only minor differences arise in their performance, where Version B consistently shows somewhat higher $m_{0.1}$-values (Table 20). The numerical results thus concur with theoretical findings reported in Ref. 38. In view of the similar nature of both algorithms, these differences are, nevertheless, sufficient to show that Version B of the conjugate gradient method offers no advantages in comparison with Version A.

The numerical tests with block overrelaxation in combination with Wilson's acceleration indicate that the convergence behavior of the solution process is essentially identical to that described in Section 4.3. From the results of Table 20 it can be seen that the applicability of the accelerated iteration is restricted to problems with moderate degrees of ill-conditioning. For higher condition numbers a sufficient amount of error reduction cannot be obtained within a reasonable number of iteration cycles, even if optimum ω-q-values are chosen. The comparatively good performance of the iteration for systems of equations with removable ill-conditioning (example B1, $\kappa < 1.0$) indicates that its rate of convergence is affected by the P_1 condition number of the scaled rather than the unscaled coefficient matrix (Appendix 2). However, in contrast to conjugate gradient methods the required number of iteration cycles increases as an exponential rather than a linear function of $\log(P_1)$. By comparing the results of Table 20 it can be seen that both versions of the conjugate gradient method converge faster than the accelerated block overrelaxation method except for problems with removable ill-conditioning. The differences are particularly large for higher values of κ and, therefore, for more ill-conditioned systems of equations. A greater contrast could also be observed if m-values of higher accuracy were compared since the conjugate gradient methods converge rather abruptly whereas the accelerated iteration converges in an approximately linear fashion.

As a result of these numerical tests it can be concluded that Version A of the conjugate gradient method represents the most efficient solution procedure. The test results show that the algorithm can be successfully applied to the solution of rather ill-conditioned systems of equations, although in this case the rate of convergence is affected by roundoff errors. In comparison with the accelerated block overrelaxation method, the conjugate gradient algorithm has the advantage that it does not require the selection of acceleration parameters.

7.2 Improvement of Performance

From the numerical tests of the previous section it was found that the comparatively good performance of conjugate gradient methods is, nevertheless, impaired by the effect of roundoff errors. This deficiency of semi-iterative methods, which was already observed in some of the earliest numerical studies (Refs. 38, 51, 73), has led to the development of various procedures for improving their performance by reducing the amount of error accumulation. The purpose of this section is to investigate several of such modifications in order to allow an evaluation of their practical usefulness.

(1) Algorithm Modifications

The standard algorithm of conjugate gradient Version A is defined in Chapter 5 by the following expressions

$$u_{c+1} = u_c+\alpha_c v_c \qquad (7.1a)$$

$$r_{c+1} = r_c-\alpha_c K v_c \qquad (7.1b)$$

$$v_{c+1} = r_{c+1}+\beta_c v_c \qquad (7.1c)$$

where

$$\alpha_c = \frac{r_c^T r_c}{v_c^T K v_c}$$

$$\beta_c = \frac{r_{c+1}^T r_{c+1}}{r_c^T r_c} \tag{7.1d}$$

$$v_o = r_o = f-Ku_o$$

Due to various orthogonality conditions among the vectors u_c, v_c, and r_c, the above algorithm can be expressed in a large number of different ways. In Ref. 38 numerous such alternative presentations are suggested and their susceptibility to roundoff errors is discussed. Several of these algorithms, here designated as coefficient modifications, differ from the standard algorithm only in the definition of α_c- and β_c-like scalar quantities. In order to present these modifications in a uniform notation, the computational scheme of Eq. 7.1 is rewritten in a slightly different form

$$u_{c+1} = u_c+\alpha_c v_c$$

$$r_{c+1} = r_c-\alpha_c K v_c \tag{7.2a}$$

$$v_{c+1} = \frac{1}{\gamma_{c+1}}(r_{c+1}+\beta_c v_c)$$

where

$$v_o = \frac{1}{\gamma_o} r_o$$

$$r_o = f-Ku_o \tag{7.2b}$$

For a total of seven coefficient modifications the definitions of the parameters α_c, β_c, and γ_{c+1} are listed in Table 21.

A second group of algorithm versions differs from the previous modifications in so far as it involves the correction of vector rather than scalar quantities. The most commonly used vector correction (Modification 8) consists of replacing the recursively calculated residuals of the standard algorithm (Eq. 7.1b) by the so-called "true" residuals (Ref. 17)

$$r_{c+1} = f-Ku_{c+1} \tag{7.3}$$

Another algorithm version (Modification 9) is obtained if the following direction vector v_c is used instead of Eq. 7.1c (Ref. 38)

$$\begin{aligned} \bar{v}_{c+1} &= r_{c+1}+\beta_c v_c \\ v_{c+1} &= \bar{v}_{c+1}-\gamma_c v_c \end{aligned} \tag{7.4}$$

where $$\gamma_c = \frac{v_{c+1}^T K v_c}{v_c^T K v_c}$$

The simultaneous application of both vector corrections (Eqs. 7.3 and 7.4) finally leads to a tenth and last modification which is included in this study.

Numerical tests with these algorithm versions indicate that, except for Modification 2, the various alternative formulations show no significant differences in comparison to the standard algorithm (Table 22). In particular, the additional computational effort which, for example, is involved in the calculation of the true residuals (Modification 8) does not result in a better performance. In all numerical tests it was observed that Modification 2 diverges after an initial period during which its behavior is essentially identical to that of the other algorithms. The breakdown of the solution process apparently occurs when the approximate solution vector u_c comes in the vicinity of the correct solution. In view of the small differences in the performance of the remaining algorithms it can be concluded that no advantage is gained by using alternative formulations of the conjugate gradient method. Because of its simplicity and its comparatively small computational effort the standard algorithm of conjugate gradient Version A is, therefore, retained throughout the remainder of this investigation.

(2) Starting Procedures

A different approach to the problem of improving the convergence of conjugate gradient algorithms involves the use of iterative methods for generating suitable starting vectors of the solution process (Refs. 17, 51, 74, 75). By eliminating certain components of the initial error vector it is expected that the effect of roundoff errors on the convergence will be reduced. In order to investigate this possibility the following numerical tests are carried out: Starting with zero initial guesses, a total of c_s iteration cycles are performed with the following solution procedures

(a) block overrelaxation ($\omega = 1.95$),

(b) block overrelaxation in combination with Wilson's acceleration ($\omega = 1.95$, $q = 20$), and

(c) conjugate gradient Version A.

In a subsequent step the resulting approximation is used as a starting vector for Version A of the conjugate gradient method. The third starting procedure differs from s-step gradient methods described in Chapter 5 in so far as only a single restart is performed.

The results of the numerical tests, carried out with several values of c_s, indicate that none of the starting procedures has the desired effect on the convergence. In several cases the performance is detrimentally affected even if the computational effort for generating the starting vector is discounted. In other words, zero initial guesses represent better starting values in certain cases than those vectors generated by the above procedures. No essential differences can be observed if other values of the acceleration parameters ω and q are chosen. Summarizing the results of these tests it can be concluded that the starting procedures described above are not suitable for improving the performance of conjugate gradient Version A.

(3) Double Precision Arithmetic

Since the convergence of semi-iterative methods is primarily affected by roundoff errors, it can be expected that the use of higher precision arithmetic has a beneficial effect on their performance. In order to investigate this effect the following two types of double precision implementation are tested numerically

(a) double precision vectors, involving double precision arithmetic as well as double precision storage of vector quantities; and

(b) double precision inner products, involving double precision accumulation of inner products whereas vector quantities are stored in single precision.

For illustration purposes a third word length configuration is included as well which involves single precision accumulation of inner products whereas vector quantities are stored in a smaller word length. The details of the implementation of the above three word length modifications are given in Table 23.

The results of the numerical tests indicate that the computer word length has a considerable effect on the rate of convergence of conjugate gradient Version A (Table 24). The use of double precision vectors (Modification 1) causes a noticeable reduction in the number of required iteration cycles, particularly for ill-conditioned problems. Even if double precision is used only in the accumulation of inner products (Modification 2), the convergence is faster than in single precision, although the effect is smaller than in the previous case. However, for higher κ-values the use of double precision arithmetic in one form or another is not sufficient to achieve convergence within n iteration cycles. A reduction in the computer word length has, as expected, a detrimental effect which is particularly strong for ill-conditioned problems whereas comparatively small differences can be observed for well-conditioned systems of equations. Although the use of double precision storage and arithmetic allows a reduction in the required number of iteration cycles, the advantage is not large enough to offset the longer execution time of double precision arithmetic. Consequently, its use cannot be recommended for improving the performance of conjugate gradient methods. Exceptions may occur for computers which allow an accumulation of double precision inner products with relatively little increase in execution time. Comparative numerical tests with the accelerated block overrelaxation method indicate that for the same type of word length configurations virtually no effect on the $m_{0.1}$-values can be observed within the tested range of κ-values.

(4) Scaling Procedures

The primary purpose of using scaling procedures is to improve the condition of a system of linear equations. Basically it is possible to distinguish between scaling procedures involving a multiplication by matrix polynomials (Refs. 7, 17) and so-called diagonal scaling transformations (Refs. 27, 28, 69). The first group of methods has the disadvantage that its use in connection with conjugate gradient algorithms results in a substantial increase in the computational effort per iteration cycle. At the same time, the computational scheme of conjugate gradient methods becomes considerably more complicated (Ref. 17, cgT-method). The investigation is, therefore, restricted to the second group of scaling procedures which transforms the original system of equations

$$Ku = f \tag{7.5a}$$

into an equivalent system

$$K_s u_s = f_s$$

where

$$K_s = D_s^{-1} K D_s^{-1}$$

$$u_s = D_s u \tag{7.5b}$$

$$f_s = D_s^{-1} f$$

D_s = diagonal scaling matrix.

The above transformation does not affect the symmetric, positive definite character of the coefficient matrix and does not cause an increase in the storage requirements.

The numerical tests of this section include a total of four different scaling matrices D_s defined by the following expressions

$$\begin{array}{ll} \text{(A)} & (D_s)_{ii} = \left[(K)_{ii}\right]^{1/2} \\ \text{(B)} & (D_s)_{ii} = \left[n_{nz_i}(K)_{ii}\right]^{1/2} \\ \text{(C)} & (D_s)_{ii} = \left[\sum_{j=1}^{n} |(K)_{ij}|\right]^{1/2} \\ \text{(D)} & (D_s)_{ii} = \left[\sum_{j=1}^{n} (K)_{ij}^{2}\right]^{1/4} \end{array} \qquad (7.6)$$

The quantity n_{nz_i}, which appears in the definition of procedure B, represents the total number of non-zero elements in the i-th row or column of the coefficient matrix.

The test results of Table 25 indicate that, with few exceptions (example B1, procedure B), the scaling transformations cause a substantial improvement in the performance of the conjugate gradient method. The effect is particularly strong for example B1, although a noticeable improvement can also be observed in the $m_{0.1}$-values of example B2. In both cases the reduction in the number of required iteration cycles is considerably larger than it could be expected from the change in the P_1-condition numbers (Appendix 2). The test results clearly show that the largest decrease in the computational effort is obtained for systems of equations with removable ill-conditioning (example B1, $\kappa < 1.0$). In contrast to previous observations (p. 135), the $m_{0.1}$-values of the test examples vary as an approximately linear function of $\log(P_1)$ of the scaled rather than the original coefficient matrix. Except for procedure B, only minor differences arise in the performance of the remaining scaling transformations. Because of its simplicity, scaling procedure A is, therefore, selected as the most suitable transformation. If the same scaling procedures are applied to the accelerated block overrelaxation method, virtually no effect on its convergence can be observed.

As a result of the numerical tests of this chapter it can be concluded that the conjugate gradient algorithm defined by Eq. 7.1 represents the most efficient solution procedure. Its performance can be improved by applying the following scaling transformation to the original system of equations

$$(D_s^{-1} K D_s^{-1})(D_s u) = (D_s^{-1} f)$$

where

$$(D_s)_{ii} = \left[(K)_{ii}\right]^{1/2} \tag{7.7}$$

$$(D_s)_{ij} = 0 \quad \text{for } i \neq j$$

Various other means of improving the performance, such as algorithm modifications, starting procedures, and the use of double precision arithmetic, were found to be of little or no practical value.

8. ERROR PREDICTION

Since iterative and semi-iterative methods yield sequences of approximate solution vectors it is necessary to define certain criteria for terminating the solution process. In this respect iterative methods basically differ from direct methods where the problem of predicting the error of an approximate solution does not arise in this form. The definition of suitable termination and error prediction criteria is of great practical importance since an inadequate termination may either result in unnecessary computing time or, even worse, in unsatisfactory numerical solutions. The basic problem of error prediction could be described as estimating the magnitude of certain error measures based on numerical values, so-called error predictors, which can be easily computed during the course of the solution process. The properties of various quantities which can be used as a basis for either measuring or predicting the error of an approximate solution are described in the following paragraphs.

In general, reliable error measures are only obtained from numerical quantities which cannot be computed without prior knowledge of the solution. Among them, the error vector

$$e_c = u-u_c \qquad (8.1)$$

represents the most natural choice since it gives a direct indication of the deviation between the correct and the approximate solution vector. In order to express its magnitude in the form of a scalar quantity, it is common practice to use norms of the errors vector, $||e_c||$, rather than the vector itself. During the course of any converging iteration process, the magnitude of the error vector decreases and approaches zero as c approaches infinity. However, the particular manner in which this error reduction occurs depends on the nature of the solution process and, in the case of linear stationary iterations, also on the eigenvalues of the iteration matrix (Section 3.2). For iterations whose T-matrices have real eigenvalues, a continuous, monotonic decrease can be expected (Fig. 1), whereas cyclic oscillations may occur for iterations with complex $\lambda_i(T)$-values (Fig. 3).

For the numerous iterative methods which can be derived from the minimization of error functions (Section 3.1.2) an alternative form of error measure can be directly based on the particular ϕ-value which is minimized at each step of the solution process. The use of error functions has the advantage that even for iterations with oscillating convergence the corresponding ϕ-value decreases strictly monotonically and approaches zero as c approaches infinity. Among the three error functions defined in Section 3.1.2, only ϕ_3 can be computed directly, whereas ϕ_1 and the most commonly used error function ϕ_2 remain generally unknown. Since error functions represent quadratic quantities, the error measure has to be based on their square root so as to allow a comparison with vector norms.

In order to obtain problem-independent error indicators it is necessary to transform the above quantities into measures of the relative rather than the absolute error. This is most appropriately done by expressing their magnitude in per cent of those initial values which correspond to zero starting vectors. A different choice of initial values is less practical since the error measures would be affected by the initial guesses u_o.

In contrast to error indicators discussed so far, error predictors have to be based on numerical quantities which can be easily calculated during the course of the solution process. Ideally, such error predictors should provide relatively close upper bounds for the above error measures. In addition, error predictors should decrease monotonically since oscillations in their magnitude would impair the accuracy and the continuity of the prediction. In general, the following two vector quantities can be used as basis for the error prediction:

(1) residual vector $r_c = f-Ku_c = Ke_c$

(2) increment vector $w_c = u_{c+1}-u_c$

From the above expressions it can be seen that the residual vector r_c is closely related to the error vector e_c. However, the nature of the relationship does not imply that a small magnitude of r_c necessarily corresponds to small errors in the approximate solution vector. The relative magnitude of both vector quantities depends on the eigenvalues of the coefficient matrix and on the specific nature of the error vector. Due to the latter effect the ratio of the two vector norms $\|r_c\|_2$ and $\|e_c\|_2$ may vary within a range identical to that of the extreme eigenvalues of K (Eq. 8.6a).

The increment vector w_c is, except for its sign, identical to the difference between the corresponding error vectors

$$w_c = u_{c+1}-u_c = -(e_{c+1}-e_c) \qquad (8.2)$$

Large increments, therefore, indicate that the solution process has not reached a stage at which it could be terminated. However, small values of w_c may simply be a sign of slow convergence rather than of sufficient error reduction.

Both the residual vector r_c as well as the increment vector w_c approach zero during the course of the solution process. For iterations with monotonic linear convergence the decrease occurs in a smooth and regular form whereas oscillations in the magnitude of the prediction quantities can be observed in case of iterations with oscillating convergence. For single-step methods, such as overrelaxation, the residual vector is not computed as part of the algorithm (Section 3.2.1). However, a closely related vector quantity, which exhibits a similar behavior as that of r_c, is available for these iterations (Eq. 3.37b).

Aside from the above two vector quantities, it is also possible to base the error prediction on the specific error function which is minimized at each step of the solution process. Since the numerical values of ϕ_1 and ϕ_2 cannot be calculated without prior knowledge of the solution vector, the applicability of the approach is restricted to error function ϕ_3. However, for iterations involving a minimization of ϕ_2 it is possible to calculate the change in the magnitude of the error function by means of the following expression

$$\Delta\phi_2 = \phi_2(u_c)-\phi_2(u_{c+1}) = (u_{c+1}-u_c)^T(r_c+r_{c+1}) \qquad (8.3)$$

In the solution of elasticity problems, the change in error function ϕ_2 has an additional meaning since, except for a constant factor, it corresponds to the change in the total potential of the discretized structure (Section 3.1.2). As in the case of error measures, predictor quantities of this type have to be based on the square root of the ϕ-values in order to be of the same nature as those derived from vector norms.

Generally, it is difficult to establish a direct correlation between the percentagewise reduction of error measures and error predictors. Instead of estimating the relative error directly, simple numerical criteria for terminating the solution process are used in most practical applications. The basic aim of these termination procedures is to detect the stage of the solution process at which a significant amount of error reduction can no longer be expected. The most commonly used procedures consist of specifying limiting values for certain norms of the residual vector r_c or the increment vector w_c. In some cases, such criteria are also applied to the change in the total potential of the discretized structure (Refs. 27, 28). The above procedures have to be considered as inadequate for general application since the magnitude of appropriate limiting values strongly depends on the particular nature of the problem. The adequacy of the results can, therefore, only be judged from experience with previous solutions of similar problems. In addition, a premature termination may occur not only as a result of oscillations in the magnitude of r_c and w_c, but also due to small values of w_c and $\Delta\phi_2$ caused by slow convergence. Similar deficiencies are found for a termination procedure suggested by Ahamed (Ref. 1) which is based on specifying a limiting value for the ratio of two energy-like quantities. If applied to elasticity problems, these quantities correspond to the external and internal work of the structure whereas their ratio is identical to the γ_c-factor of Wilson's acceleration (Section 4.3.1).

The different convergence behavior of conjugate gradient methods has led to the development of specific termination techniques for this group of solution procedures (Refs. 5, 34, 90). One of these methods, suggested by Ginsburg (Ref. 34), is based on measuring the effect of roundoff errors by calculating the deviation between the "true" and the recursive residuals (Section 7.2). Although the procedure is suitable for terminating the solution process at a time when a significant amount of error reduction cannot be achieved anymore, an error prediction at intermediate stages is not possible with this method.

Aside from termination procedures, various methods have also been developed for predicting the absolute error of an approximate solution. Provided a close upper bound for the spectral radius of the iteration matrix is known, the magnitude of the absolute error can be estimated in a relatively simple way (Refs. 25, 63, 81, 84). The procedure has the disadvantage that the prediction itself, as well as the determination of $\|T\|_{sr}$, can be successfully carried out only for relatively inefficient iterations with monotonic linear convergence (Section 3.2). A different approach, suggested by Stiefel (Ref. 73) is free of such restrictions, although it results in (not necessarily close) lower bounds for the length of the error vector. Conservative error estimates, which would be of considerably greater practical interest, can only be obtained if lower bounds for the minimum eigenvalue of the coefficient matrix are known (Appendix 1). A third prediction procedure suggested by Albrecht (Ref. 2) allows the calculation of rigorous upper and lower bounds for the solution and for the error vector without prior knowledge of eigenvalues. Unfortunately, the application of the method is restricted to certain slowly converging linear stationary iterations. In addition, major computational problems arise in the determination of suitable starting vectors (Ref. 68) and in the separation of positive and negative elements of the iteration matrix. Consequently, all three procedures have to be considered as unsuitable for practical applications. In any case, the prediction of the absolute error has the disadvantage that the specification of suitable limit values depends on the particular nature of the problem. However, it would have been possible to modify the above procedures so as to allow an estimation of the relative error of the approximate solution.

One of the few methods for directly predicting the percentagewise error reduction (Ref. 61) is based on a ratio of energy-like quantities, similar to that described in context with Ahamed's termination procedure (p. 156). The method only provides less useful lower bounds of the relative error, which may even deteriorate into trivial estimates less than zero. A more pragmatic approach rests on the assumption that the error vector decreases at approximately the same rate as the prediction quantities, provided the error measures as well as the error predictors are nondimensionalized in a suitable way. Numerical tests of iterative methods with monotonic linear convergence indicate that the procedure allows a comparatively accurate prediction of the relative error as long as initial irregularities are properly taken into account. The success in using such error predictors results from the fact that practically all computable quantities decrease in a monotonic, regular form for this type of iterations. The situation is different for iterative methods with oscillating convergence, where a gross underestimation of the remaining error may occur due to oscillations in the magnitude of the prediction quantities. Because of their simple nature the same error predictiors can also be applied to solution methods which do not exhibit linear convergence.

In order to investigate various possibilities for estimating the relative error of approximate solutions obtained from conjugate gradient Version A, a number of numerical tests were carried out. The investigation included a total of three different error measures ε defined by the following expressions

$$\varepsilon_1 = 100 \frac{|(u)_k - (u)_k^c|}{|(u)_k|} \tag{8.4a}$$

$$\varepsilon_2 = 100 \frac{\|e_c\|_2}{\|u\|_2} = 100 \left(\frac{e_c^T e_c}{u^T u}\right)^{1/2} \tag{8.4b}$$

$$\varepsilon_3 = 100 \left(\frac{\phi_2}{u^T f}\right)^{1/2} = 100 \left(\frac{e_c^T r_c}{u^T f}\right)^{1/2} \tag{8.4c}$$

In the definition of ε_1, the quantity $(u)_k$ corresponds to the element of the solution vector whose absolute value has the largest magnitude. Therefore, ε_1 is identical to the relative error ε_c used throughout the preceding numerical tests (Section 2.2). The error measure ε_2 is based on the euclidean length of the error vector whereas ε_3 indicates the reduction of error function ϕ_2 which forms the basis for the derivation of conjugate gradient Version A (Section 5.1).

From theory it is known that ε_2 and ε_3 decrease continuously throughout the solution process (Ref. 38). However, a strictly monotonic type of convergence is not guaranteed in the case of error measure ε_1. For reasons discussed previously, the denominators of all three error measures correspond to those initial values obtained for zero starting vectors u_o (p. 152).

Based on various computable vector quantities of the conjugate gradient algorithm, a total of five different error predictors were defined for the numerical tests

$$\psi_1 = 100 \frac{\|r_c\|_2}{\|f\|_2} = 100 \left(\frac{r_c^T r_c}{f^T f}\right)^{1/2} \qquad (8.5a)$$

$$\psi_2 = 100 \frac{\|v_c\|_2}{\|f\|_2} = 100 \left(\frac{v_c^T v_c}{f^T f}\right)^{1/2} \qquad (8.5b)$$

$$\psi_3 = 100 \frac{\left[\dfrac{(r_c^T r_c)^{3/2}}{r_c^T K r_c}\right]}{\left[\dfrac{(f^T f)^{3/2}}{f^T K f}\right]} \qquad (8.5c)$$

$$\psi_4 = 100 \frac{1}{\gamma_c} \frac{\|v_c\|_2}{\|f\|_2} = 100 \frac{1}{\gamma_c} \left(\frac{v_c^T v_c}{f^T f}\right)^{1/2} \qquad (8.5d)$$

$$\psi_5 = 100 \frac{1}{\gamma_c} \frac{\|v_c\|_1}{\|f\|_1} = 100 \frac{1}{\gamma_c} \frac{\sum_{i=1}^{n} |(v)_i^c|}{\sum_{i=1}^{n} |(f)_i|} \qquad (8.5e)$$

The error predictor ψ_1 measures the reduction of the euclidean length of the residual vector r_c whereas ψ_2 represents a similar quantity applied to the direction vector v_c. Independent of the solution procedure, the following relationship between error measure ε_2 and error predictor ψ_1 can be shown to exist

$$\frac{1}{P}\,\psi_1 \leq \varepsilon_2 \leq P\,\psi_1 \tag{8.6a}$$

where P represents the ratio of the extreme eigenvalues of the coefficient matrix (Appendix 1). In addition, error predictors ψ_1 and ψ_2 are related by the following inequality

$$\begin{aligned} \|r_c\|_2 &< \|v_c\|_2 \\ \psi_1 &< \psi_2 \end{aligned} \tag{8.6b}$$

which can be derived from the basic properties of the conjugate gradient algorithm (Ref. 38). The error predictor ψ_3 is based on the same quantity used by Stiefel (Ref. 73) for the prediction of a lower bound of the absolute error (p. 157). Since the matrix-vector product Kr_c is not formed as part of the standard algorithm of conjugate gradient Version A, the use of error predictor ψ_3 requires additional computational effort or, if a recursive calculation of Kr_c is used, additional vector storage. For none of these three error predictors it can be guaranteed that their magnitude will decrease monotonically during the course of the solution process (Ref. 38).

The use of error predictors ψ_4 and ψ_5 is based on certain specific properties of conjugate gradient algorithms whose nature is briefly described in what follows. As mentioned in Section 5.1, the residual vector for conjugate gradient Version B can be calculated from vector quantities of Version A by using the relationship

$$r_c^B = \frac{1}{\gamma_c} v_c$$

where $$\gamma_c = 1+\beta_{c-1}\gamma_{c-1} \tag{8.7}$$

$$\gamma_o = 1$$

Consequently, error function ϕ_3^B, which is minimized at each step of the Version B algorithm, can be expressed as

$$\phi_3^B = r_c^{B^T} r_c^B = \frac{1}{\gamma_c^2} v_c^T v_c \tag{8.8}$$

Error predictor ψ_4, therefore, has the character of a monotonically decreasing error measure for Version B of the conjugate gradient method. From theory it is known that the euclidean length of the error vector e_c for Version A is smaller than that of the B-version (Ref. 38).

$$\|e_c\|_2 < \|e_c^B\|_2 \qquad (8.9)$$

Since both e_c^B as well as ϕ_3^B could be used as a basis for measuring the relative error in case of conjugate gradient Version B, it can be expected that error predictor ψ_4 results in conservative error estimates for Version A. However, it is not possible to conclude that ψ_4 represents a strictly conservative estimate. From the theoretical findings of Ref. 38 it can also be shown that

$$\psi_4 < \psi_1 < \psi_2 \qquad (8.10)$$

since $$\|r_c^B\|_2 < \|r_c\|_2 < \|v_c\|_2$$

Error predictor ψ_5 is based on similar considerations as those described for ψ_4. The only disparity arises in the use of a different norm of the vector v_c, for which a strictly monotonic decrease can no longer be guaranteed. In order to make the error prediction independent of the initial approximations, all predictors are expressed in per cent of those initial values obtained for zero starting vectors.

From the numerical results of Table 26 and Fig. 12 it can be seen that the error indicators ε_1 and ε_2 provide nearly identical measures of the error reduction, whereas ε_3 generally decreases at a somewhat lower rate. Therefore, the simple error indicator ε_1, used throughout the previous numerical tests, is found to represent an adequate measure of the relative error, at least for conjugate gradient Version A. As predicted by the theory, error measures ε_2 and ε_3 decrease monotonically during the course of the solution process, whereas the magnitude of ε_1 may undergo minor oscillations in certain cases (Ref. 38).

The behavior of the error predictors ψ_1, ψ_2, and ψ_3 is characterized by drastic variations in their numerical values (Fig. 12). The oscillations are particularly strong for ill-conditioned problems where the predicted relative error may exceed the actual value by several orders of magnitude. For well-conditioned systems of equations, however, the amplitude of the oscillations is usually smaller. In general, the lowest predictions are obtained on the basis ψ_3, whereas the numerical values of ψ_2 are consistently higher than those of the other estimates (Eq. 8.6b). Because of their strongly oscillatory behavior, all three error predictors have to be considered as unsuitable for practical applications. A meaningful error prediction on the basis of these quantities is only possible during the final stage of the solution process if the ψ-values assume a minimum in their oscillations. Since these minima occur in irregular cycle intervals, a continuous measure of the error reduction is not possible even at that stage. The numerical results also show that the additional computational effort (or additional vector storage) for predictor ψ_3 does not result in significantly better error estimates. Because of the irregular behavior and the limited applicability of these three error predictors, the corresponding $m_{0.1}$-values are not included in Table 26.

In contrast to the previous prediction quantities, error estimate ψ_4 decreases monotonically during the course of the solution and allows a rather accurate, continuous prediction of the relative error (Fig. 13). Minor problems arise only during the initial phase of the solution process where ψ_4 decreases more rapidly than the corresponding error measures, thus resulting in non-conservative error estimates. However, as soon as any noticeable amount of error reduction is achieved, ψ_4 assumes the character of a conservative error predictor. In a large number of numerical tests it was observed that the cross-over between the error measures ε and predictor ψ_4 occurs for relative errors greater than 10 per cent.

In comparison to ψ_4, error predictor ψ_5 provides slightly more accurate estimates during the initial phase of the solution process. However, the numerical values of ψ_5 no longer decrease in a strictly monotonic form, although the magnitude of their oscillations is comparatively small. As illustrated by Fig. 13, both quantities allow a rather accurate, conservative error prediction, at least within the practical range of ε-values. From the nearly identical results of Table 26 it can be concluded that both predictors are equally suitable. Nevertheless, the use of error estimate ψ_5 is preferred since non-conservative predictions are less likely to occur than in case of ψ_4.

9. EFFECTS ON THE SOLUTION PROCESS

The primary purpose of this final part of the dissertation is to investigate various factors influencing the convergence behavior and the efficiency of the conjugate gradient method. Particular attention is paid to those effects which have not been studied in the comparative numerical tests of Chapters 5 and 7. In addition, certain specific problems arising in the practical application of the conjugate gradient method are discussed.

(1) Nodal Point Enumeration

According to the criteria defined in Section 3.2.1, the conjugate gradient algorithm belongs to the group of total-step iterations. Theoretically, its convergence behavior is not affected by the nodal point enumeration or, more precisely, by the ordering of the equations (p. 34). However, the sequence in which the numerical operations are carried out may influence the amount of roundoff error accumulation (Ref. 64). Due to these roundoff errors it is possible that the convergence of the solution process may be indirectly affected by the numbering sequence.

In order to study this effect, numerical tests were carried out with example B1 using a total of four different enumeration schemes:

(a) Original numbering along horizontal rows of nodal points,

(b) "Best" numbering along vertical columns of nodal points, resulting in a minimum band width of the coefficient matrix,

(c) Random numbering, and

(d) "Worst" numbering in the sense that the sum of the (absolute) differences between the numbers of adjacent nodal points is a maximum or at least close to a maximum.

The numerical results indicate that the $m_{0.1}$-values of the solution process are practically unaffected by the enumeration sequence. This observation can even be made for ill-conditioned problems where roundoff errors prevent a convergence within n iteration cycles. The effect of the nodal point enumeration on the convergence of the conjugate gradient method can, therefore, be neglected for practical purposes. As a result, the generation of input data for a finite element analysis is considerably simplified since no attention has to be paid to any form of optimum enumeration. In this respect, the conjugate gradient method compares favorably with direct solution procedures where the band width of the coefficient matrix (and, therefore, the nodal point enumeration) has a dominant effect on the storage requirements and the execution time.

(2) Initial Guesses

As described in Chapter 2, all previous numerical tests were carried out with zero initial guesses in order to allow an unbiased comparison of the efficiency of various solution procedures. The purpose of this section is to investigate the specific effects of starting vectors on the convergence behavior and the performance of the conjugate gradient method. According to Ref. 38 the most suitable initial approximations are those for which the initial residual vector r_o is similar to the eigenvector corresponding to the smallest eigenvalue of the coefficient matrix. Since the determination of such initial approximations is rather difficult under practical conditions, various alternatives were investigated in numerical tests. In particular, the following initial guesses u_o, most of them expressed as scalar multiples of the solution vector u, were included

(0) $u_o = 0$
(1) $u_o = 0.10\ u$
(2) $u_o = 0.50\ u$
(3) $u_o = 0.90\ u$
(4) $u_o = 0.95\ u$

(5) $u_o = 10.0\ u$

(6) $u_o = -1.0\ u$

(7) u_o = linear variation

(8) u_o = random variation

For the cantilever problem of example B1, a seventh type of initial approximation was defined by assuming a linear variation of the vertical displacements between zero at the support and $(u)_{max}$ at the cantilever tip. In addition, an eighth and last starting vector was obtained by choosing random values of the vertical displacements, scaled to a range between zero and $(u)_{max}$. In both cases, the corresponding horizontal displacements were assumed to be zero.

The numerical tests with these initial approximations indicate that their effect is difficult to asses because its magnitude is influenced not only by the condition of the problem, but also by the required accuracy of the solution. However, for ill-conditioned problems, where the influence of the starting vectors is more pronounced, the following general trends can be observed. Initial approximations of type 1 through 4 apparently have no significant beneficial or detrimental effect in comparison to zero initial guesses (Table 27). Although the starting vector 4 differs from the correct solution only by 5 per cent of its magnitude, a noticeable improvement in the convergence cannot be observed except during the initial phase of the solution process. The reverse effect occurs for starting vectors 5 and 6, where u_o differs from the correct solution by a full order of magnitude or by its sign respectively. In both cases an initial delay in the convergence occurs, whereas the differences in comparison to zero initial guesses nearly vanish in the later phase of the solution process. Starting vector 7, which simulates initial guesses as they could be realistically estimated under practical conditions, leads to a comparatively rapid initial decrease of the relative error. However, in the final stage of the solution, the error reduction occurs at a lower rate than that for $u_o = 0$. As illustrated by the results of Table 27, randomly assigned initial approximations have a detrimental effect on the convergence and generally result in the highest $m_{0.1}$-values.

From the numerical tests it can be concluded that the choice of initial guesses has a comparatively small effect on the performance of the conjugate gradient method, especially for well-conditioned systems of equations. Under normal conditions, the difficulties in estimating and assigning suitable approximations are likely to outweigh potential savings in computational effort. Even if close approximations of the solution vector are available, as, for instance, in a step-wise analysis of nonlinear problems, a significant decrease in the required number of iteration cycles cannot be expected. The results of the numerical tests, therefore, concur with previous findings on the suitability of starting procedures described in Section 7.2.

(3) Load Vector

The convergence behavior of iterative and semi-iterative solution methods is primarily affected by the specific properties of the coefficient matrix K. Among various other factors, a dominant role is usually played by the extreme eigenvalues of K and their corresponding eigenvectors. Under certain conditions, however, the convergence behavior may also depend on the load vector f or, if non-zero initial guesses are chosen, on the initial residual vector r_o. In the numerical tests of the first part of the dissertation, this influence can be assessed by comparing the results for test examples A5 and A6 which employ identical structural discretizations but differ in their loading conditions (Appendix 2). Virtually all tested solution procedures show some form of disparity in the rates of convergence for these two examples. The consistently faster convergence in the case of example A6 can be explained by the orthogonality between the load vector f and the eigenvectors corresponding to several of the highest and lowest eigenvalues of the coefficient matrix. Therefore, the "effective" P-condition number of example A6 is smaller than the actual ratio of the extreme eigenvalues (Appendix 2). Due to specific properties of the test example, the above orthogonality conditions can be interpreted as a result of applying a symmetric load to a symmetric structural configuration.

In order to illustrate the effect of the load vector f in additional numerical tests, the structural discretization of example B1 is subjected to a uniformly distributed tip load applied under an angle α against the horizontal axis. The numerical results indicate that for $\alpha = 0°$ (symmetric loading) as well as for $\alpha = 90°$ (anti-symmetric loading) the solution process converges noticeably faster than under normal conditions, the improvement being the greatest for the symmetric loading case (Fig. 14). Except for initial irregularities, virtually no differences in the rates of convergence can be observed for intermediate values of the load angle α. The corresponding $m_{0.1}$-values are practically identical to those obtained for the original loading condition of example B1.

It can, therefore, be concluded that the load vector may cause a significant change in the convergence behavior of the conjugate gradient method if certain orthogonality conditions exist between the vector f and the eigenvalues of the coefficient matrix. For symmetric structures, deviations from the normal behavior will occur if the load vector corresponds to a strictly symmetric or anti-symmetric loading case.

(4) Finite Element Type

Throughout this investigation, numerical tests were carried out with systems of equations arising in the finite element analysis of plane stress problems using so-called CST-elements (Section 2.1). Supplementary tests indicate that the convergence behavior of the solution procedures remains essentially unchanged if the structural configurations of example A1 through A6 are used as discretizations of axi-symmetric problems. The similarity of the results can be explained by the fact that the general properties of finite element coefficient matrices are not affected by the element type (Appendix 1). However, certain numerical problems may arise with elements whose derivation involves approximations on the basis of St.-Venant's principle (beam, plate, and shell elements). Since the generalized load and displacement vectors of these elements contain several types of physical quantities, the numerical values of the stiffness components may differ considerably. The resulting systems of equations are frequently ill-conditioned because large deviations in the diagonal elements of the coefficient matrices correspond to high P-condition numbers. Numerical tests described in Ref. 27 indicate, however, that the existing form of ill-conditioning is similar to the removable type discussed in context with example B1 (Appendix 2). Therefore, simple diagonal scaling transformations described in Section 7.2 can be successfully used for improving the condition of the coefficient matrices.

(5) Singular Coefficient Matrices

In general, systems of linear equations arising from finite element analysis are positive definite provided the numerical problem is properly formulated (Appendix 1). Under certain conditions, however, it is possible that the coefficient matrices become singular or, more specifically, positive semi-definite. Situations of this type may arise, for instance, in the step-wise analysis of nonlinear elasticity problems or in cases where the essential boundary conditions of a structural problem are not specified. The investigation of singular systems of equations was restricted to cases where the structure as a whole or parts of it may undergo rigid body motions. A solution for this type of problems exists only if the load vector f is in self-equilibrium or if the external loads are transferred to the supports without causing unrestrained movements of the structure. As shown in Ref. 36 the physical interpretation of static equilibrium corresponds to the following orthogonality condition

$$f^T x = 0 \tag{9.1}$$

where $K x = 0$

$x =$ eigenvectors corresponding to zero eigenvalues of K

Due to the presence of rigid body motions, the elements of the solution vector may be of arbitary magnitude whereas the deflected shape of the structure as well as its stresses are unique.

Numerical tests with various structural problems of this type indicate that the conjugate gradient method can be successfully applied to the solution of singular systems of equations. It is necessary, however, to terminate the solution process as soon as a significant amount of error reduction can no longer be achieved. Under no conditions should the process be continued beyond a stage at which the denominator of the α_c-coefficients (Eq. 7.1d) becomes less than or equal to zero. Otherwise, the accuracy of the numerical results deteriorates due to a rapid growth of round-off errors. In cases where the load vector f does not satisfy Eq. 9.1, it is not possible to obtain meaningful results since the solution process diverges.

(6) Roundoff Errors

The effect of roundoff errors on the convergence behavior of the conjugate gradient method has been investigated in numerical tests described in Chapter 7. The purpose of the additional tests of this section is to illustrate their effect on the accuracy of the solution vector.

At any stage of an iterative or semi-iterative solution process, the total error of an approximate solution u_c is composed of "iteration" and roundoff errors. Iteration errors represent that part of the total deviation between u_c and u which is reduced at each step of the solution procedure. Roundoff errors, on the other hand, are caused by the fact that the numerical calculations are carried out with a finite number of binary digits. Their magnitude gradually increases during the course of the solution whereas errors of the former type show the opposite behavior.

In order to separate the effect of roundoff errors, the iteration process has to be continued until the solution procedure no longer allows a reduction of the iteration error. Generally it is difficult to determine this stage of the iteration process since changes in the last digits of the solution vector may continue for a large number of iteration cycles, especially for ill-conditioned problems. In actual numerical tests the criteria for terminating the solution process have to be chosen with great care since the magnitude of the remaining errors is strongly influenced by this choice.

In order to investigate the effect of roundoff errors on the accuracy of the solution, a number of numerical tests were carried out using the conjugate gradient method as well as a direct solution procedure (Cholesky decomposition). The basic aim of these tests is to provide a qualitative assessment of the effect rather than a detailed study of various influence factors. The numerical tests were performed with examples B1 and B2 which may assume any arbitrary degree of ill-conditioning if appropriate values of the parameter κ are specified (Appendix 2). For both solution procedures the diagonal scaling transformation defined by Eq. 7.7 is applied to the original system of equations.

The test results essentially show that the conjugate gradient algorithm is capable of providing solutions having the same accuracy as those obtained by Cholesky's method. In particular, the κ-values for which a meaningful solution of the test examples can no longer be obtained because of roundoff errors were found to be identical for both methods. The numerical results also indicate that the attainable accuracy of the solution procedures closely agrees with Rosanoff's relationship for estimating the error of the solution vector (Refs. 64,65,66)

$$d_r = d_i - \log(P) \tag{9.2}$$

In the above expression, d_i and d_r represent the initial and the remaining number of correct decimal digits whereas P corresponds to the condition number of the coefficient matrix (Appendix 1).

Supplementary tests of the conjugate gradient method show that the effect of roundoff errors can be reduced by comparatively simple means such as restarting the solution process or by using higher precision arithmetic for the accumulation of inner products. On the other hand, the magnitude of the errors is increased if the conjugate gradient algorithm is used without prior scaling transformation. A detailed study of these and various other effects was, however, not carried out within the scope of this dissertation.

(7) Large Systems of Equations

In order to keep the total amount of computations within reasonable limits, the numerical tests of this investigation were carried out with relatively small, although not necessarily well-conditioned test examples. It is the purpose of this section to investigate the use of the conjugate gradient method for the solution of larger systems of equations as they arise in the practical application of the finite element method. Particular attention is paid to the effect of increasingly finer discretizations of a structural problem on the convergence behavior and the efficiency of the conjugate gradient algorithm.

The total computational effort for the solution of a system of equations, Z, is proportional to the size of the system, n, multiplied by the required number of iteration cycles, m.

$$Z \sim n\ m \tag{9.3}$$

Since in the absence of roundoff errors the conjugate gradient method constitutes an n-step algorithm (Chapter 5), the theoretical limit for m coincides with the total number of equations. In order to describe the effect of successively finer discretizations of a basic structural problem, designated by the subscript 0, it is necessary to express the parameters Z, n, and m as functions of g, the ratio of the corresponding mesh spacings h

$$g = \frac{h_o}{h} \tag{9.4}$$

For two-dimensional elasticity problems which can be analyzed by means of CST-type elements, these relationships assume the following form

$$n \approx g^2 n_o \tag{9.5a}$$

$$m \leq n \approx g^2 n_o \tag{9.5b}$$

$$Z \approx g^4 Z_o \tag{9.5c}$$

The last expression, however, is only valid if the computer implementation of the solution procedure, in particular its storage manipulations, remain independent of the actual size of the system. Otherwise, discontinuities will occur in the proportionality factors of Eq. 9.5c. The same growth rate of the total computational effort, g^4, incidentally applies to direct solution methods (Ref. 22).

In order to investigate the discretization effect numerical tests were carried out with two examples whose basic structural configurations, C_o and D_o, are described in Appendix 2. The successively finer discretizations were chosen in such a way that the mesh spacing ratio g corresponds to integer powers of 2

$$g_k = \frac{h_o}{h_k} = 2^k \tag{9.6}$$

The size of the resulting systems of equations, the $m_{0.1}$-values of the conjugate gradient method as well as the total computational effort, measured by the corresponding central processor time (in seconds), is given in Table 28. The results clearly indicate that the required number of iteration cycles increases at a lower rate than its theoretical limit n (Fig. 15). This observation is surprising in so far as it is known that a more accurate discretization of a structural problem has a detrimental effect on the condition of the corresponding stiffness matrix (Refs. 29,30,49). Since the parameters m and n increase at different rates, it is not possible to express the value of m as a fixed percentage of the size of the system (cf. Ref. 90). However, it can certainly be concluded that the relationship

$$m = n \tag{9.7}$$

represents an unrealistically pessimistic estimate of the required number of iteration cycles.

In order to investigate the rate of increase of the parameters n, m, and Z more closely, the following growth ratios for two consecutive discretizations are defined

$$n_k = \alpha_k \, n_{k-1}$$

$$m_k = \beta_k \, m_{k-1}$$

$$Z_k = \gamma_k \, Z_{k-1} \approx \alpha_k \beta_k \, Z_{k-1} \tag{9.8}$$

Based on Eq. 9.5 the coefficients α_k and β_k should assume a value of 4, whereas the magnitude of γ_k should approach 16 for sufficiently high values of k. The numerical tests indicate, however, that β_k approaches 2 under the given conditions whereas the coefficient γ_k assumes values in the neighborhood of 8 (Table 29). By generalizing these findings to arbitrary mesh spacing ratios, the theoretical growth rates of Eq. 9.5 can be replaced by the following experimentally observed rates

$$\begin{aligned} n &\approx g^2 \, n_o \\ m &\approx g \, m_o \\ Z &\approx g^3 \, Z_o \end{aligned} \tag{9.9}$$

As indicated by the last expression, the computational effort of the conjugate gradient method increases at a lower rate than that of direct solution procedures (Eq. 9.5c). According to Refs. 22 and 28, even greater differences in the growth rates can be expected for three-dimensional elasticity problems although the behavior of the conjugate gradient method is affected not only by the number of dimensions but also by the order of the governing differential equation.

10. SUMMARY AND CONCLUSIONS

The application of iterative and semi-iterative methods for the solution of systems of linear equations with positive definite coefficient matrices is investigated in this dissertation. The study concentrates on systems of equations arising in the "assumed displacement" approach of the finite element method. However, the findings are equally applicable to other types of equations provided the coefficient matrices have essentially the same properties. The primary objective of the investigation is to determine the most suitable of those solution procedures in which the storage advantages of iterative methods can be exploited.

In the first part of the dissertation a survey and classification of iterative, semi-iterative, and acceleration algorithms is presented. The discussion includes the computational details of the individual methods, their specific properties as well as possible forms of their derivation. With most of the solution procedures numerical tests are carried out in order to investigate their convergence behavior and to allow a comparison of their performance. Various conclusions drawn from the results of the numerical study are summarized in Chapter 6.

Three of the most promising solution methods are subjected to additional tests in the second part of the investigation. The numerical tests with larger and more ill-conditioned systems of equations indicate that the following version of the conjugate gradient method represents the most efficient solution procedure

$$\begin{aligned} u_{c+1} &= u_c+\alpha_c v_c \\ r_{c+1} &= r_c-\alpha_c K v_c \\ v_{c+1} &= r_{c+1}+\beta_c v_c \end{aligned} \tag{10.1a}$$

where

$$\begin{aligned} \alpha_c &= \frac{r_c^T r_c}{v_c^T K v_c} \\ \beta_c &= \frac{r_{c+1}^T r_{c+1}}{r_c^T r_c} \end{aligned} \tag{10.1b}$$

$$\begin{aligned} \gamma_c &= 1+\beta_{c-1}\gamma_{c-1} \\ v_o &= r_o = f-Ku_o \\ \gamma_o &= 1 \end{aligned}$$

In comparison to various iterative methods, the conjugate gradient algorithm has the advantage of not requiring the specification of acceleration parameters. A numerical investigation of several alternative forms of the above solution procedure shows that algorithm modifications have no beneficial effect on the performance. Similar conclusions can be drawn regarding the use of so-called "starting procedures" for generating suitable initial approximations u_o. On the other hand, a decrease in the required number of iteration cycles is possible if the solution process is carried out in double precision. However, the reduction of the computational effort is likely to be off-set by the longer execution time of double precision arithmetic. The numerical investigation shows that diagonal scaling transformations represent the only effective means of improving the performance of the conjugate gradient algorithm. The simplest of these procedures transforms the original system of equations

$$K\ u = f \tag{10.2a}$$

into the following form

$$\begin{aligned} K_s \qquad u_s &= \quad f_s \\ (D_s^{-1}\ K\ D_s^{-1})(D_s u) &= (D_s^{-1}\ f) \end{aligned}$$

where
$$(D_s)_{ii} = \sqrt{(K)_{ii}} \tag{10.2b}$$

$$(D_s)_{ij} = 0 \quad \text{for } i \neq j$$

In all numerical tests it was found that the above scaling transformation has a beneficial effect on the performance, although the magnitude of the improvement depends on the nature of the coefficient matrix.

Various problems connected with the termination of iterative solution processes and the prediction of the relative error of approximate solutions are discussed in Chapter 8 of the dissertation. For the conjugate gradient method a new procedure is proposed which allows a comparatively accurate, conservative estimation of the relative error. Based on quantities contained in Eq. 10.1, the suggested error predictor can be expressed as

$$\psi = 100 \frac{1}{\gamma_c} \frac{\sum_{i=1}^{n} |(v)_i^c|}{\sum_{i=1}^{n} |(f)_i|} \qquad (10.3)$$

In Chapter 9, various effects on the convergence behavior and the efficiency of the conjugate gradient method are examined. Numerical tests indicate that the ordering of the equations and, therefore, the nodal point enumeration have practically no effect on the convergence. Similarly, the choice of initial guesses u_o influences the required number of iteration cycles only to a limited extent. Under certain conditions, however, the convergence behavior of the solution procedure may be affected by the loading condition of the structural system. Test results also show that the conjugate gradient algorithm can be applied to the solution of certain singular systems of equations which arise, for instance, if the essential boundary conditions of a structural problem are not specified.

A numerical study of the effect of roundoff errors on the solution of ill-conditioned equations indicates that the attainable accuracy of the conjugate gradient method is comparable to that of direct solution procedures. Relatively large systems arising from successively finer discretizations of structural problems are investigated in the final section. The results show that the required computational effort increases at a comparatively low rate and that the conjugate gradient method can be successfully used for solving large systems of linear equations.

11. TABLES

Group No.	Designation of Group	S	T	Basic Algorithm
I	Successive Approximation	I	$I-K$	$u_{c+1} = u_c+r_c$
II	Jacobi	D^{-1}	$I-D^{-1}K$	$u_{c+1} = u_c+D^{-1}r_c$
III	de la Garza	$D_1^{-1}K$	$I-D_1^{-1}KK$	$u_{c+1} = u_c+D_1^{-1}Kr_c$
IV	Kaczmarz	KD_1^{-1}	$I-KD_1^{-1}K$	$u_{c+1} = u_c+KD_1^{-1}r_c$
V	Block Jacobi	D_2^{-1}	$I-D_2^{-1}K$	$u_{c+1} = u_c+D_2^{-1}r_c$

Table 1

Linear Stationary Iterations - Basic Groups

Iteration Type		Unaccelerated Versions ($\omega = 1.0$)	Accelerated Versions ($\omega \neq 1.0$)
Total Step Iterations		A Basic Algorithm	D
Single Step Iterations	Seidel Process	B	E
	Aitken Process	C	F

Table 2

Designation of Algorithm Versions

Group No.	S-Matrices					
	Version A	Version B	Version C	Version D	Version E	Version F
I	I Successive Approximation	$(I+L)^{-1}$	$(I+L)^{-1^T}(2I-D)(I+L)^{-1}$	ωI	$\omega(I+\omega L)^{-1}$	$\omega(I+\omega L)^{-1^T}(2I-\omega D)(I+\omega L)^{-1}$
II	D^{-1} Jacobi	$(D+L)^{-1}$ Gauss-Seidel	$(D+L)^{-1^T}D(D+L)^{-1}$ Aitken	ωD^{-1} Extrapolated Jacobi	$\omega(D+\omega L)^{-1}$ (Point-) Over-relaxation	$\omega(2-\omega)(D+\omega L)^{-1^T}D(D+\omega L)^{-1}$ Extrapolated Aitken
III	$D_1^{-1}K$ de la Garza	$(D_1+L_1)^{-1}K$	$(D_1+L_1)^{-1^T}D_1(D_1+L_1)^{-1}K$	$\omega D_1^{-1}K$	$\omega(D_1+\omega L_1)^{-1}K$	$\omega(2-\omega)(D_1+\omega L_1)^{-1^T}D_1(D_1+\omega L_1)^{-1}K$
IV	KD_1^{-1}	$K(D_1+L_1)^{-1}$ Kaczmarz	$K(D_1+L_1)^{-1^T}D_1(D_1+L_1)^{-1}$	ωKD_1^{-1} Cimmino	$\omega K(D_1+\omega L_1)^{-1}$	$\omega(2-\omega)K(D_1+\omega L_1)^{-1^T}D_1(D_1+\omega L_1)^{-1}$
V	D_2^{-1} Block Jacobi	$(D_2+L_2)^{-1}$ Block Gauss-Seidel	$(D_2+L_2)^{-1^T}D_2(D_2+L_2)^{-1}$ Block Aitken	ωD_2^{-1} Extrapolated Block Jacobi	$\omega(D_2+\omega L_2)^{-1}$ Block Over-relaxation	$\omega(2-\omega)(D_2+\omega L_2)^{-1^T}D_2(D_2+\omega L_2)^{-1}$ Extrapolated Block Aitken

Table 3

Linear Stationary Iterations - S-Matrices

Group No.	Designation of Group	Iteration Matrices T		
		Version D	Version E	Version F
I	Successive Approximation	$I-\omega K$	$I-\omega(I+\omega L)^{-1}K$	$I-\omega(I+\omega L)^{-1^T}(2I-\omega D)(I+\omega L)^{-1}K$
II	Jacobi	$I-\omega D^{-1}K$	$I-\omega(D+\omega L)^{-1}K$	$I-\omega(2-\omega)(D+\omega L)^{-1^T}D(D+\omega L)^{-1}K$
III	de la Garza	$I-\omega D_1^{-1}KK$	$I-\omega(D_1+\omega L_1)^{-1}KK$	$I-\omega(2-\omega)(D_1+\omega L_1)^{-1^T}D_1(D_1+\omega L_1)^{-1}KK$
IV	Kaczmarz	$I-\omega KD_1^{-1}K$	$I-\omega K(D_1+\omega L_1)^{-1}K$	$I-\omega(2-\omega)K(D_1+\omega L_1)^{-1^T}D_1(D_1+\omega L_1)^{-1}K$
V	Block Jacobi	$I-\omega D_2^{-1}K$	$I-\omega(D_2+\omega L_2)^{-1}K$	$I-\omega(2-\omega)(D_2+\omega L_2)^{-1^T}D_2(D_2+\omega L_2)^{-1}K$

Table 4

Linear Stationary Iterations - Iteration Matrices

ω	Example A1	Example A4	Example A6
0.1	(17,000)	(140,000)	(3,750)
0.2	(12,000)	(82,000)	(2,400)
0.3	(9,800)	(59,500)	(1,800)
0.4	(8,350)	(47,500)	(1,400)
0.5	(7,300)	(39,500)	(1,150)
0.6	(6,400)	(34,500)	(990)
0.7	(5,700)	(30,500)	(855)
0.8	(5,100)	(27,500)	(750)
0.9	(4,600)	(25,000)	(670)
1.0	(4,150)	Div.	Div.

Table 5

$m_{0.1}$-Values for Block Jacobi Version D (Extrapolated Block Jacobi Iteration)

ω	Example A1		Example A4		Example A6	
	Jacobi Version F	Block Jacobi Version F	Jacobi Version F	Block Jacobi Version F	Jacobi Version F	Block Jacobi Version F
1.0	(1,200)	(1,200)	(9,700)	(9,700)	(185)	(185)
1.1	(1,050)	(1,050)	(9,600)	(9,550)	(170)	(165)
1.2	(905)	(905)	(9,800)	(9,700)	(155)	(150)
1.3	(820)	(820)	(10,500)	(10,000)	(150)	(140)
1.4	(765)	(765)	(11,500)	(11,000)	(150)	(140)
1.5	(755)	(755)	(13,000)	(13,000)	(165)	(150)
1.6	(805)	(805)	(16,000)	(15,500)	(190)	(170)
1.7	(955)	(955)	(20,500)	(19,500)	(240)	(215)
1.8	(1,300)	(1,300)	(28,500)	(27,500)	(355)	(310)
1.9	(2,450)	(2,450)	(51,000)	(48,500)	(675)	(595)
2.0	Div.	Div.	Div.	Div.	Div.	Div.

Table 6

$m_{0.1}$-Values for Jacobi and Block Jacobi Versions F (Extrapolated Aitken and Block Aitken Iteration)

Example A1			Example A4			Example A6		
ω	Jacobi Version E	Block Jacobi Version E	ω	Jacobi Version E	Block Jacobi Version E	ω	Jacobi Version E	Block Jacobi Version E
1.0	(2,100)	(2,100)	1.0	(11,000)	(11,500)	1.0	292	294
1.1	(1,700)	(1,700	1.1	(9,300)	(9,600	1.1	238	240
1.2	(1,400)	(1,400)	1.2	(7,750)	(8,000)	1.2	194	194
1.3	(1,150)	(1,150)	1.3	(6,350)	(6,600)	1.3	156	156
1.4	(900)	(900)	1.4	(5,100)	(5,300)	1.4	122	124
1.5	(700)	(700)	1.5	(4,000)	(4,150)	1.5	94	94
1.6	(520)	(520)	1.6	(3,000)	(3,100)	1.6	67	67
1.7	(365)	(365)	1.7	(2,100)	(2,200)	1.7	41	41
1.8	218	218	1.8	(1,250)	(1,350)	1.8	33	34
1.9	59	59	1.9	470	592	1.9	62	63
2.0	Div.	Div.	2.0	Div.	Div.	2.0	Div.	Div.
1.86	134	134	1.88	(635)	(735)	1.71	38	38
1.87	120	120	1.89	(550)	(660)	1.72	35	34
1.88	104	104	1.90	470	592	1.73	31	31
1.89	84	84	1.91	340	512	1.74	27	26
1.90	59	59	1.92	380	430	1.75	23	21
1.91	76	76	1.93	345	340	1.76	29	30
1.92	85	85	1.94	445	200	1.77	29	30
1.93	90	90	1.95	548	250	1.78	28	29
1.94	112	112	1.96	660	312	1.79	27	28

Table 7

$m_{o.1}$-Values for Jacobi and Block Jacobi Versions E (Overrelaxation and Block Overrelaxation)

ω'	Example A1	Example A4	Example A6
1.0	(9,200)	(57,500)	(2,050)
1.1	(8,500)	(53,000)	(1,850)
1.2	(7,850)	(49,500)	(1,700)
1.3	(7,300)	(46,000)	(1,600)
1.4	(6,850)	(43,500)	(1,500)
1.5	(6,400)	(41,000)	(1,400)
1.6	(6,000)	(39,000)	(1,300)
1.7	(5,700)	(37,000)	(1,200)
1.8	(5,350)	(35,000)	(1,150)
1.9	(5,100)	(33,500)	(1,100)
2.0	(4,850)	(32,000)	(1,050)

Table 8

$m_{0.1}$-Values for Successive Approximation Version D

ω	Example A1		Example A4		Example A6	
	De la Garza Version D	Kaczmarz Version D	De la Garza Version D	Kaczmarz Version D	De la Garza Version D	Kaczmarz Version D
0.1	(105,000)	(105,000)	(2,750,000)	(2,700,000)	(39,000)	(36,500)
0.2	(88,000)	(88,500)	(2,200,000)	(2,150,000)	(32,500)	(30,500)
0.3	(79,500)	(80,000)	(1,800,000)	(1,750,000)	(29,500)	(27,500)
0.4	(73,500)	(74,000)	(1,550,000)	(1,500,000)	(27,500)	(26,000)
0.5	(69,500)	(69,500)	(1,400,000)	(1,350,000)	(25,500)	(24,500)
0.6	(66,000)	(66,500)	Div.	Div.	Div.	Div.
0.7	Div.	Div.	Div.	Div.	Div.	Div.

Table 9

$m_{o.1}$-Values for de la Garza and Kaczmarz Versions D

No.	Designation	Computational Form		Derivation	
				Error Function	Direction Vector
1	Steepest Descent	$u_{c+1} = u_c+\gamma_c r_c$	$\gamma_c = \dfrac{r_c^T r_c}{r_c^T K r_c}$	ϕ_2	r_c
2	Krasnoselskii	$u_{c+1} = u_c+\gamma_c r_c$	$\gamma_c = \dfrac{r_c^T K r_c}{r_c^T K K r_c}$	ϕ_3	r_c
3	Householder	$u_{c+1} = u_c+\gamma_c K r_c$	$\gamma_c = \dfrac{r_c^T r_c}{r_c^T K K r_c}$	ϕ_1	Kr_c
4	Cauchy	$u_{c+1} = u_c+\gamma_c K r_c$	$\gamma_c = \dfrac{r_c^T K K r_c}{r_c^T K K K K r_c}$	ϕ_3	Kr_c
5	Gastinel-Householder	$u_{c+1} = u_c+\gamma_c t_c$	$\gamma_c = \dfrac{r_c^T t_c}{t_c^T K t_c}$	ϕ_2	t_c
6	Gastinel	$u_{c+1} = u_c+\gamma_c K t_c$	$\gamma_c = \dfrac{r_c^T t_c}{t_c^T K K t_c}$	ϕ_1	Kt_c

Table 10 Computational Forms of Basic Nonlinear Stationary Iterations

Method	Example A1	Example A4	Example A6
Steepest Descent	(4,850)	(34,500)	1,000
Krasnoselskii	(4,850)	(33,000)	990
Householder	(255,000)	(6,850,000)	(120,000)
Cauchy	(255,000)	(6,600,000)	(120,000)
Gastinel-Householder	(2,900)	(97,000)	(1,900)

Table 11

$m_{0.1}$-Values for Basic Nonlinear Stationary Iterations

ω	Example A1	Example A4	Example A6
1.00	(4,850)	(34,500)	1,000
0.95	268	1,320	132
0.90	236	995	104
0.85	332	870	152
0.80	436	800	116
0.70	400	---	148
0.60	532	---	184
0.50	504	---	180

Table 12

$m_{o.1}$-Values for Almost Optimum Steepest Descent Method

q	Example A1		Example A4		Example A6	
	$b_K = \lambda_{max}(K)$	$b_K = \|\|K\|\|_1$	$b_K = \lambda_{max}(K)$	$b_K = \|\|K\|\|_1$	$b_K = \lambda_{max}(K)$	$b_K = \|\|K\|\|_1$
5	(4,150)	(4,650)	(28,000)	(29,000)	(890)	(1,300)
10	(2,400)	(2,700)	(17,000)	(18,000)	(515)	(760)
20	(1,300)	(1,450)	(9,500)	(9,900)	(270)	(405)
40	(665)	(745)	(4,950)	(5,200)	118	192
100	196	256	(2,000)	(2,050)	136	--
∞	--	--	--	--	--	--
$q_{opt} = c_o$	114	122	308	312	54	64

Table 13

$m_{0.1}$-Values for Lanczos' Method

Example A1		Example A4		Example A6	
ω	$m_{0.1}$	ω	$m_{0.1}$	ω	$m_{0.1}$
0.1	(1,750)	0.1	(10,000)	0.1	240
0.2	(1,400)	0.2	(8,300)	0.2	194
0.3	(1,150)	0.3	(6,700)	0.3	154
0.4	(900)	0.4	(5,350)	0.4	120
0.5	(695)	0.5	(4,150)	0.5	90
0.6	(515)	0.6	(3,100)	0.6	60
0.7	(355)	0.7	(2,200)	0.7	40
0.8	200	0.8	(1,350)	0.8	59
0.9	128	0.9	(585)	0.9	125
1.0	Div.	1.0	Div.	1.0	Div.
0.83	154	0.91	(505)	0.65	43
0.84	136	0.92	432	0.66	38
0.85	114	0.93	342	0.67	33
0.86	86	0.94	208	0.68	38
0.87	108	0.95	248	0.69	40
0.88	104	0.96	308	0.70	40
0.89	122	0.97	456	0.71	39

Table 14

$m_{0.1}$-Values for Faddeev I Acceleration Applied to Block Gauss-Seidel

No.	Designation	Computational Form	Parameters
1	Wilson	$u_q = \gamma_L u_L$ $\gamma_L = \frac{f^T u_L}{u_L^T K u_L}$	q
2	Forsythe	$u_q = u_L + \gamma_L (u_L - u_{L-p})$ $\gamma_L = \frac{r_L^T (u_L - u_{L-p})}{(u_L - u_{L-p})^T K (u_L - u_{L-p})}$	p,q
3	Aitken	$(u)_i^q = (u)_i^L - \gamma i_L \left[(u)_i^L - (u)_i^{L-p}\right]$ $\gamma i_L = \frac{(u)_i^L - (u)_i^{L-p}}{(u)_i^L - 2(u)_i^{L-p} + (u)_i^{L-2p}}$ $i = 1 \ldots n$	p,q
4	Ishibashi	$u_q = u_L - \gamma_L (u_L - u_{L-p})$ $\gamma_L = \frac{(u)_k^L - (u)_k^{L-p}}{(u)_k^L - 2(u)_k^{L-p} + (u)_k^{L-2p}}$ $(u)_k$ = arbitarily selected element of u	p,q (k)
5	Dyer I	$u_q = u_L - \gamma_L (u_L - u_{L-p})$ $\gamma_L = \frac{(u_L - u_{L-p})^T (u_L - u_{L-p})}{(u_L - 2u_{L-p} + u_{L-2p})^T (u_L - u_{L-p})}$	p,q
6	Milne I	$u_q = u_L - \gamma_L (u_L - u_{L-p})$ $\gamma_L = \frac{(u_L - u_{L-p})^T (u_{L-p} - u_{L-2p})}{(u_L - 2u_{L-p} + u_{L-2p})^T (u_{L-p} - u_{L-2p})}$	p,q
7	Modified Aitken	$u_q = u_L - \gamma_L (u_L - u_{L-p})$ $\gamma_L = \frac{(u_L - u_{L-p})^T (u_L - 2u_{L-p} + u_{L-2p})}{(u_L - 2u_{L-p} + u_{L-2p})^T (u_L - 2u_{L-p} + u_{L-2p})}$	p,q
8	Irons-Tuck	$u_q = u_L - \gamma_L (u_L - u_{L-1})$ $\gamma_L = \gamma_{L-q} - (\gamma_{L-q} - 1) \frac{(u_L - u_{L-1})^T (u_L - u_{L-1} - u_{L-2} + u_{L-3})}{(u_L - u_{L-1} - u_{L-2} + u_{L-3})^T (u_L - u_{L-1} - u_{L-2} + u_{L-3})}$ γ_{L-q} = γ-value of previous acceleration interval	(q=2)
9	Rashid	$(u)_i^q = (u)_i^L + \gamma i_L \left[(u)_i^L - (u)_i^{L-1}\right]$ $\gamma i_L = \sum_{j=1}^{k} \left[\frac{(u)_i^L - (u)_i^{L-1}}{(u)_i^{L-1} - (u)_i^{L-2}}\right]^j$ $i = 1 \ldots n$	k,q
10	Dyer II	$u_q = u_L - \gamma_L (u_L - u_{L-p})$ $\gamma_L = \frac{u_L^T KK (u_L - u_{L-p})}{(u_L - u_{L-p})^T KK (u_L - u_{L-p})}$	p,q
11	Milne II	$u_q = u_L - \gamma 1_L (u_L - u_{L-p}) - \gamma 2_L (u_L - u_{L-2p})$ $\gamma 1_L = \frac{\alpha_1}{1 + \alpha_1 + \alpha_2}$ $\gamma 2_L = \frac{\alpha_2}{1 + \alpha_1 + \alpha_2}$ where α_1 and α_2 are obtained from $\begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} -\beta_{10} \\ -\beta_{20} \end{bmatrix}$ $\beta_{ij} = \beta_{ji} = (u_{L-ip} - u_{L-ip-p})^T (u_{L-jp} - u_{L-jp-p})$ $i = 0,1,2$ $j = 0,1,2$	p,q

Table 15

Computational Forms of Nonlinear Stationary Accelerations

Example	ω	q-Values					
		2	5	10	20	40	∞ (No Acceleration)
A1	1.20	106	96	100	140	200	(1,400)
	1.40	64	66	80	100	160	(900)
	1.60	34	40	50	80	150	(520)
	1.80	52	34	40	60	80	218
	1.95	106	84	100	110	104	132
A4	1.20	528	348	320	340	400	(8,000)
	1.40	356	236	220	240	280	(5,300)
	1.60	230	156	150	160	200	(3,100)
	1.80	152	100	100	120	160	(1,350)
	1.95	285	196	180	180	160	250

Table 16

$m_{0.1}$-Values for Wilson's Acceleration Applied to Block Overrelaxation

Example	ω	q-Values					
		10	20	40	100	∞	1 (No Acceleration)
A1	$1/\lambda_{max}(K)$	(2,400)	(1,300)	(635)	(380)	-	(9,200)
	$2/\|K\|_1$	(1,350)	(710)	(325)	(405)	-	(5,400)
	$2/\lambda_{max}(K)$	(1,200)	(635)	280	(355)	-	(4,850)
A4	$1/\lambda_{max}(K)$	(17,000)	(9,450)	(4,900)	(1,900)	-	(57,500)
	$2/\|K\|_1$	(9,100)	(4,950)	(2,550)	(915)	-	(33,500)
	$2/\lambda_{max}(K)$	(8,700)	(4,700)	(2,450)	(865)	-	(32,000)
A6	$1/\lambda_{max}(K)$	(505)	(255)	108	--	-	(2,050)
	$2/\|K\|_1$	(370)	178	146	--	-	(1,500)
	$2/\lambda_{max}(K)$	(245)	112	174	--	-	(1,050)

Table 17

$m_{0.1}$-Values for Stiefel's Acceleration Applied to Successive Approximation Version D

Example	n	Version A	Version B
A1	22	15	18
A2	77	21	26
A3	42	43	49
A4	24	23	24
A5	48	23	24
A6	48	14	15

Table 18

$m_{0.1}$-Values for Conjugate Gradient Methods

Version	Example	s-Values					
		2	5	10	15	20	≥ 25
A	A1	(2,400)	(850)	388	15	15	15
	A4	(16,500)	(6,600)	(2,550)	(1,300)	690	23
	A6	488	192	84	14	14	14
B	A1	(2,450)	(865)	200	60	18	18
	A4	(16,000)	(6,600)	(2,800)	(1,800)	312	24
	A6	490	198	78	15	15	15

Table 19

$m_{0.1}$-Values for s-Step Gradient Methods

Example	κ	Conjugate Gradient Version A	Conjugate Gradient Version B	Block Over-relaxation/ Wilson's Acceleration*
B1	10^{-2}	255	300	100
	10^{-1}	138	159	140
	1	78	93	160
	10^{+1}	141	165	220
	10^{+2}	264	330	720
	10^{+3}	483	591	>1000
	10^{+4}	699	900	>1000
	10^{+5}	963	>1000	>1000
B2	10^{-2}	195	207	>1000
	10^{-1}	102	114	128
	1	57	63	136
	10^{+1}	216	228	>1000
	10^{+2}	366	423	>1000
	10^{+3}	525	549	>1000

Table 20

Comparison of $m_{0.1}$-Values

*Note: $\omega = 1.95$, $q = 20$

Modification	α_c	β_c	γ_{c+1}	δ_c	Remarks
0	$\frac{r_c^T r_c}{v_c^T K v_c}$	$\frac{r_{c+1}^T r_{c+1}}{r_c^T r_c}$	1	1	Standard Version
1	$\frac{r_c^T v_c}{v_c^T K v_c}$	$-\frac{r_{c+1}^T K v_c}{v_c^T K v_c}$	1	1	
2	$\frac{r_o^T v_c}{v_c^T K v_c}$	$-\frac{r_{c+1}^T K v_c}{v_c^T K v_c}$	1	1	
3	$\frac{r_c^T r_c}{v_c^T K v_c}$	$\frac{r_{c+1}^T r_{c+1} - r_{c+1}^T r_c}{r_c^T r_c}$	1	1	
4	$\frac{r_c^T r_c}{v_c^T K v_c} \cdot \frac{1}{\delta_c}$	$\frac{r_{c+1}^T r_{c+1}}{r_c^T r_c} \cdot \delta_c$	1	$1-\beta_{c-1}\frac{v_{c-1}^T K v_c}{v_c^T K v_c}$	$\delta_o = 1$
5	$\frac{1}{v_c^T K v_c}$	$r_{c+1}^T r_{c+1}$	$r_{c+1}^T r_{c+1}$	1	$\gamma_o = r_o^T r_o$
6	1	$\frac{r_{c+1}^T r_{c+1}}{r_c^T r_c} \gamma_c$	$\frac{r_{c+1}^T K r_{c+1}}{r_{c+1}^T r_{c+1}} - \beta_c$	1	$\gamma_o = \frac{r_o^T K r_o}{r_o^T r_o}$
7	$\frac{r_c^T r_c}{r_c^T K r_c}$	$\frac{r_{c+1}^T r_{c+1}}{r_c^T r_c} \gamma_c$	$1 - \frac{\beta_c \alpha_{c+1}}{\alpha_c}$	1	$\gamma_o = 1$

Table 21

Coefficients of Algorithm Modifications

Modification	κ-Values (Example B1)			
	1	10^2	10^3	10^4
0	78	264	483	699
1	78	264	486	723
2	Div.	Div.	Div.	Div.
3	78	264	483	726
4	78	261	474	714
5	78	261	474	684
6	78	264	474	684
7	78	267	468	696
8	78	270	492	744
9	78	264	483	726
10	78	267	492	753

Table 22

$m_{0.1}$-Values for Algorithm Modifications

Word Length Modification	Designation	Mantissa Size (in Bits)		
		Storage of K and f	Storage of Vector Quantities	Arithmetic
0	Single Precision	48	48	48
1	Double Precision Vectors	48	96	96
2	Double Precision Inner Products	48	48	96
3	Reduced Precision Vectors	48	30	48

Table 23

Mantissa Size of Word Length Modifications

Modification	κ-Values (Example B1)			
	1	10^2	10^3	10^4
0	78	264	483	699
1	75	198	327	438
2	78	249	432	600
3	81	294	552	861

Table 24

$m_{0.1}$-Values for Word Length Modifications

Example	κ	No Scaling	Scaling Procedure			
			A	B	C	D
B1	10^{-2}	255	69	75	75	72
	1	78	75	81	75	75
	10^{+2}	264	96	105	96	96
	10^{+4}	699	120	129	120	120
B2	10^{-2}	195	123	168	126	126
	1	57	54	57	51	54
	10^{+2}	366	240	366	249	246
	10^{+3}	525	291	546	318	312

Table 25

$m_{0.1}$-Values for Scaling Procedures

Example	κ	Error Measures			Error Predictors	
		ε_1	ε_2	ε_3	ψ_4	ψ_5
B1	1	78	78	93	99	102
	10^2	264	267	315	351	354
	10^4	699	723	834	904	908
B2	1	57	60	66	69	72
	10^1	216	216	222	228	228
	10^2	366	366	390	426	426

Table 26

$m_{0.1}$-Values of Error Measures and Error Predictors

Initial Guesses	κ-Values (Example B1)		
	1	10^2	10^4
0	78	264	699
1	78	264	696
2	78	258	708
3	78	255	693
4	75	252	678
5	87	288	762
6	81	273	735
7	93	318	771
8	99	342	849

Table 27

$m_{0.1}$-Values for Initial Guesses

Example	k	$\frac{h_o}{h}$	n	$m_{0.1}$	$z_{0.1}$
C	0	1	19	9	0.072
	1	2	71	21	0.485
	2	4	271	46	3.44
	3	8	1055	98	27.22
D	0	1	10	9	0.045
	1	2	33	21	0.240
	2	4	115	51	1.73
	3	8	423	110	12.37
	4	16	1615	223	97.07

Table 28

Convergence of Large Systems of Equations

Example	k	α_k	β_k	γ_k
C	1	3.74	2.33	6.74
	2	3.82	2.19	7.09
	3	3.89	2.13	7.91
D	1	3.30	2.33	5.33
	2	3.48	2.43	7.21
	3	3.68	2.16	7.15
	4	3.82	2.03	7.85

Table 29

Effect of Successively Finer Discretizations

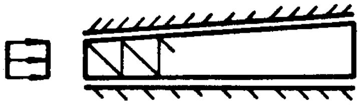
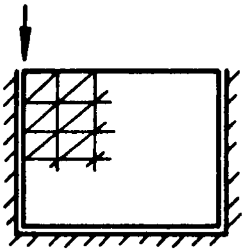
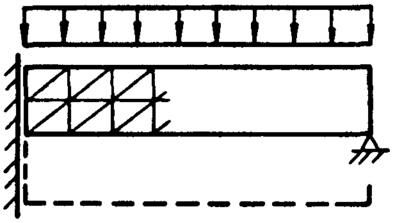
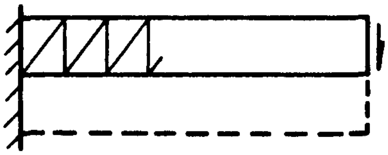
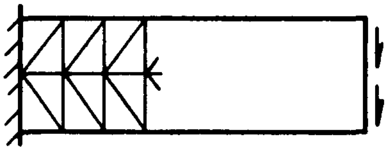
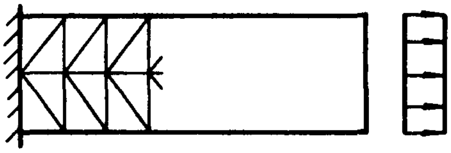
Example	Description	Structural System
A1	Thick-walled cylinder under internal pressure	
A2	Concentrated load on half-space	
A3	Simply supported beam with uniform lateral load	
A4	Cantilever with lateral tip load	
A5	Cantilever with lateral tip load	
A6	Cantilever with axial tip load	

Table A.1

Test Examples A1 through A6

Example	n	n_e	n_p
A1	22	20	22
A2	77	72	49
A3	42	32	27
A4	24	16	18
A5	48	32	27
A6	48	32	27
B1	146	120	77
B2	152	128	81
C_o	19	16	15
C_1	71	64	45
C_2	271	256	153
C_3	1055	1024	561
D_o	10	6	8
D_1	33	24	21
D_2	115	96	65
D_3	423	384	225
D_4	1615	1536	833

Table A.2

Numerical Characteristics of Test Examples

Example	$\lambda_{max}(K)$	$\lambda_{min}(K)$	$P(K)$	$P_1(K)$
A1	$3.16\cdot10^4$	$2.25\cdot10^1$	$1.41\cdot10^3$	$1.74\cdot10^3$
A2	$8.09\cdot10^4$	$5.40\cdot10^2$	$1.50\cdot10^2$	$2.62\cdot10^2$
A3	$9.40\cdot10^4$	$1.13\cdot10^0$	$8.30\cdot10^4$	$1.61\cdot10^5$
A4	$3.90\cdot10^4$	$3.86\cdot10^0$	$1.01\cdot10^4$	$1.65\cdot10^4$
A5	$5.55\cdot10^4$	$5.12\cdot10^0$	$1.09\cdot10^4$	$2.56\cdot10^4$
A6	$5.55\cdot10^4$	$5.12\cdot10^0$	$1.09\cdot10^4$	$2.56\cdot10^4$

Table A.3

Extreme Eigenvalues and Condition Numbers
of Examples A1 through A6

κ	Example B1		Example B2	
	$P_1(K)$	$P_1(K_s)$	$P_1(K)$	$P_1(K_s)$
10^{-4}	$1.07\cdot10^8$	$8.78\cdot10^3$	--	--
10^{-3}	$1.08\cdot10^7$	$8.90\cdot10^3$	--	--
10^{-2}	$1.15\cdot10^6$	$9.65\cdot10^3$	$5.01\cdot10^5$	$1.48\cdot10^5$
10^{-1}	$1.74\cdot10^5$	$1.55\cdot10^4$	$7.85\cdot10^3$	$2.66\cdot10^3$
10^0	$6.26\cdot10^4$	$5.49\cdot10^4$	$3.54\cdot10^3$	$3.10\cdot10^3$
10^{+1}	$4.44\cdot10^5$	$3.85\cdot10^5$	$2.28\cdot10^6$	$1.85\cdot10^6$
10^{+2}	$4.11\cdot10^6$	$3.53\cdot10^6$	$3.14\cdot10^8$	$2.48\cdot10^8$
10^{+3}	$4.07\cdot10^7$	$3.48\cdot10^7$	$3.12\cdot10^{10}$	$2.46\cdot10^{10}$
10^{+4}	$4.06\cdot10^8$	$3.47\cdot10^8$	$3.11\cdot10^{12}$	$2.45\cdot10^{12}$
10^{+5}	$4.06\cdot10^9$	$3.47\cdot10^9$	--	--

Table A.4

Condition Numbers of Examples B1 and B2

Designation of Group	Version A/D	Version B/E	Version C/F
Successive Approximation	1	1	2
Jacobi	1	1	2
de la Garza	2	2*	4*
Kaczmarz	2	2	4
Block Jacobi	1	1	2

*Use of recursive relationships necessary

Table A.5

Computational Effort of Linear Stationary Iterations (Matrix-Vector Products per Iteration Cycle)

12. FIGURES

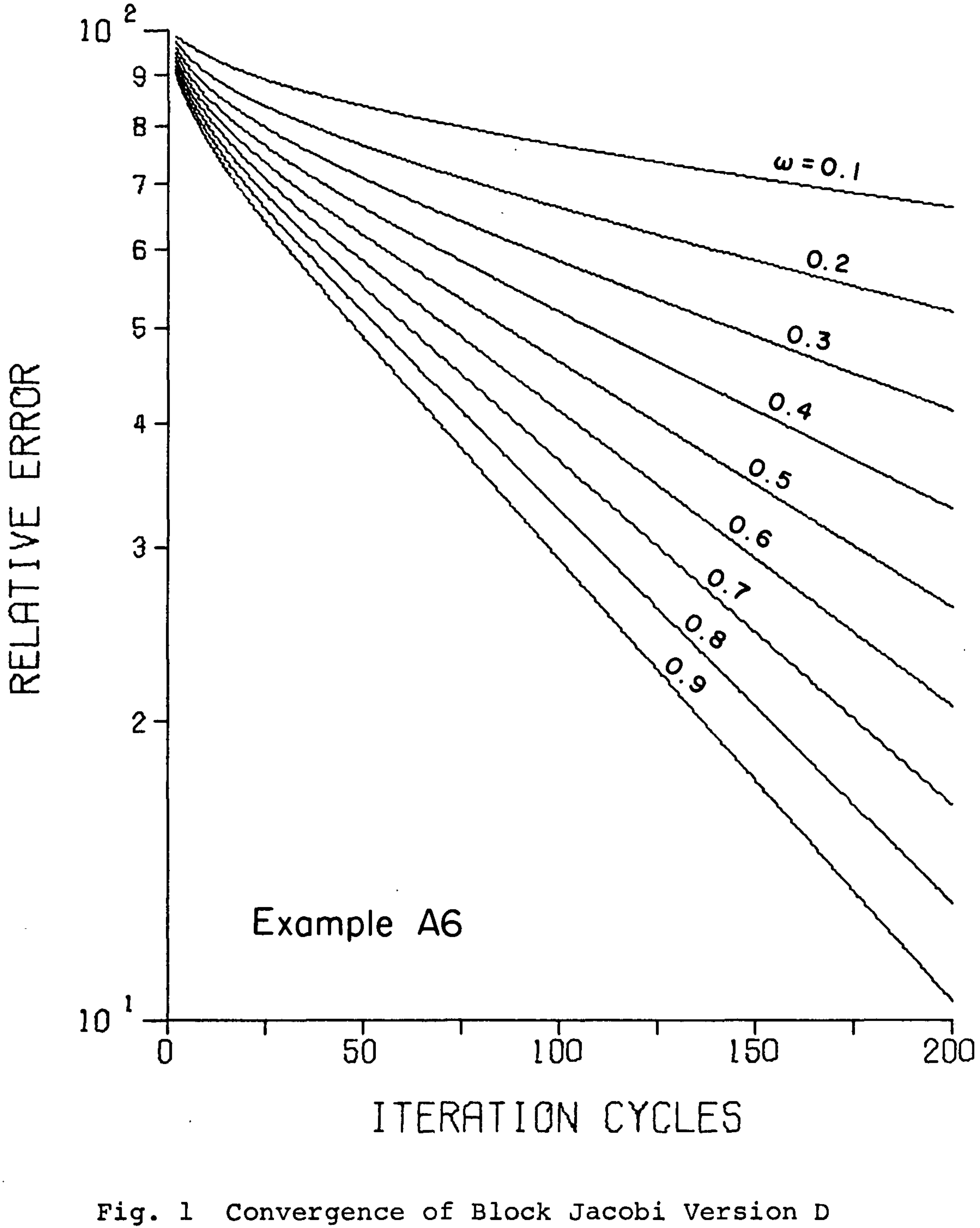


Fig. 1 Convergence of Block Jacobi Version D (Extrapolated Block Jacobi Iteration)

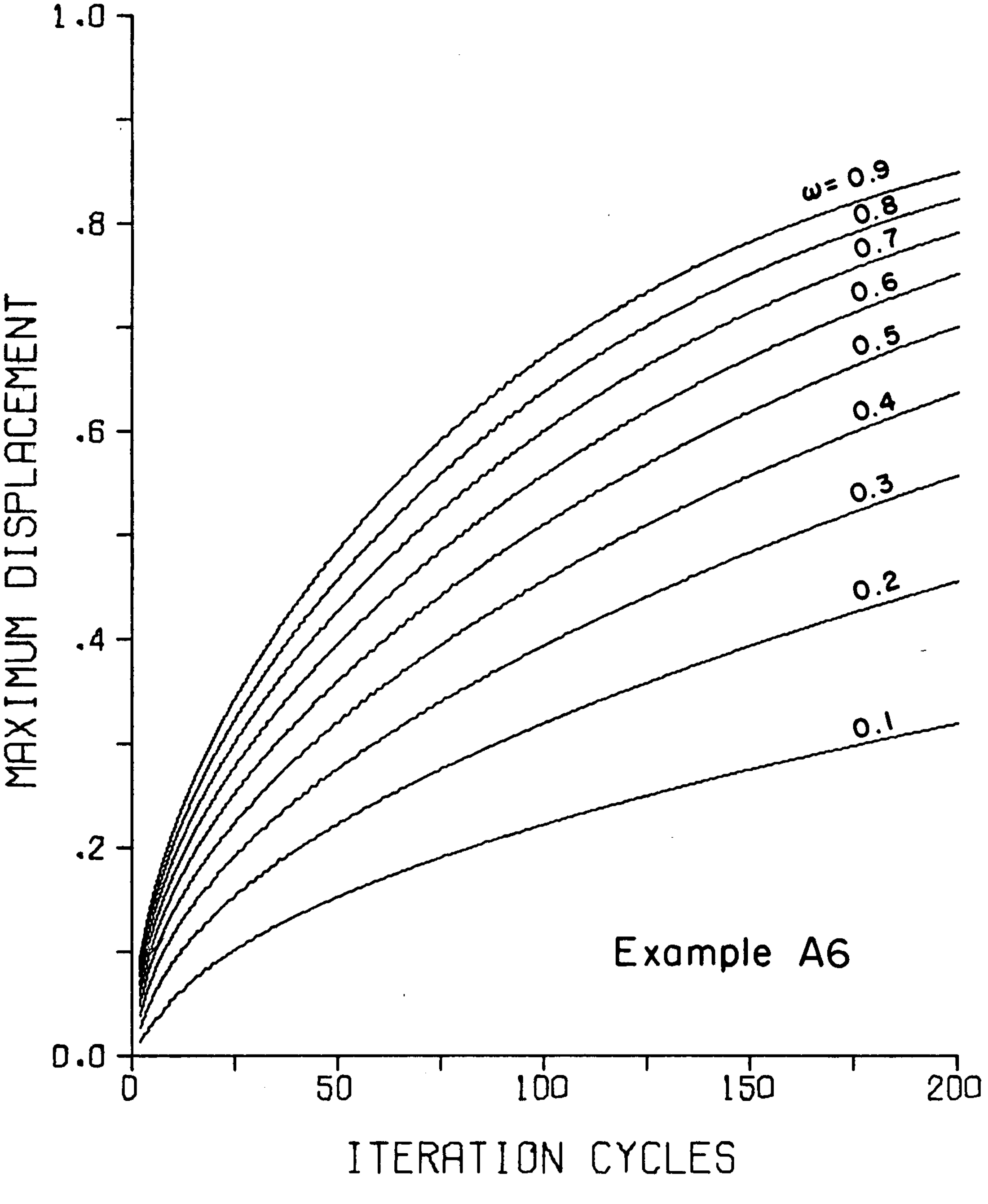


Fig. 2 Convergence of Block Jacobi Version D (Extrapolated Block Jacobi Iteration)

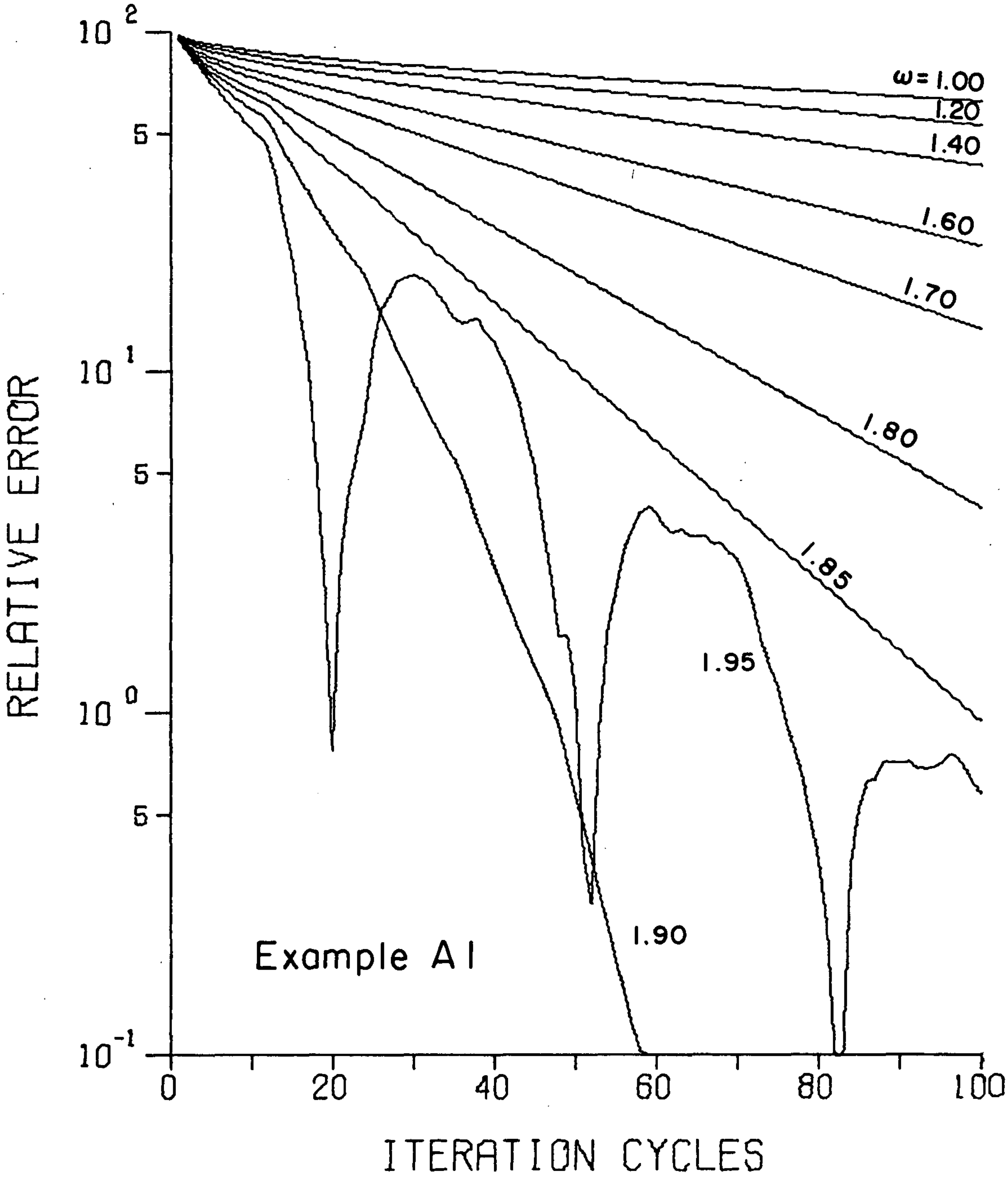


Fig. 3 Convergence of Block Jacobi Version E
(Block Overrelaxation)

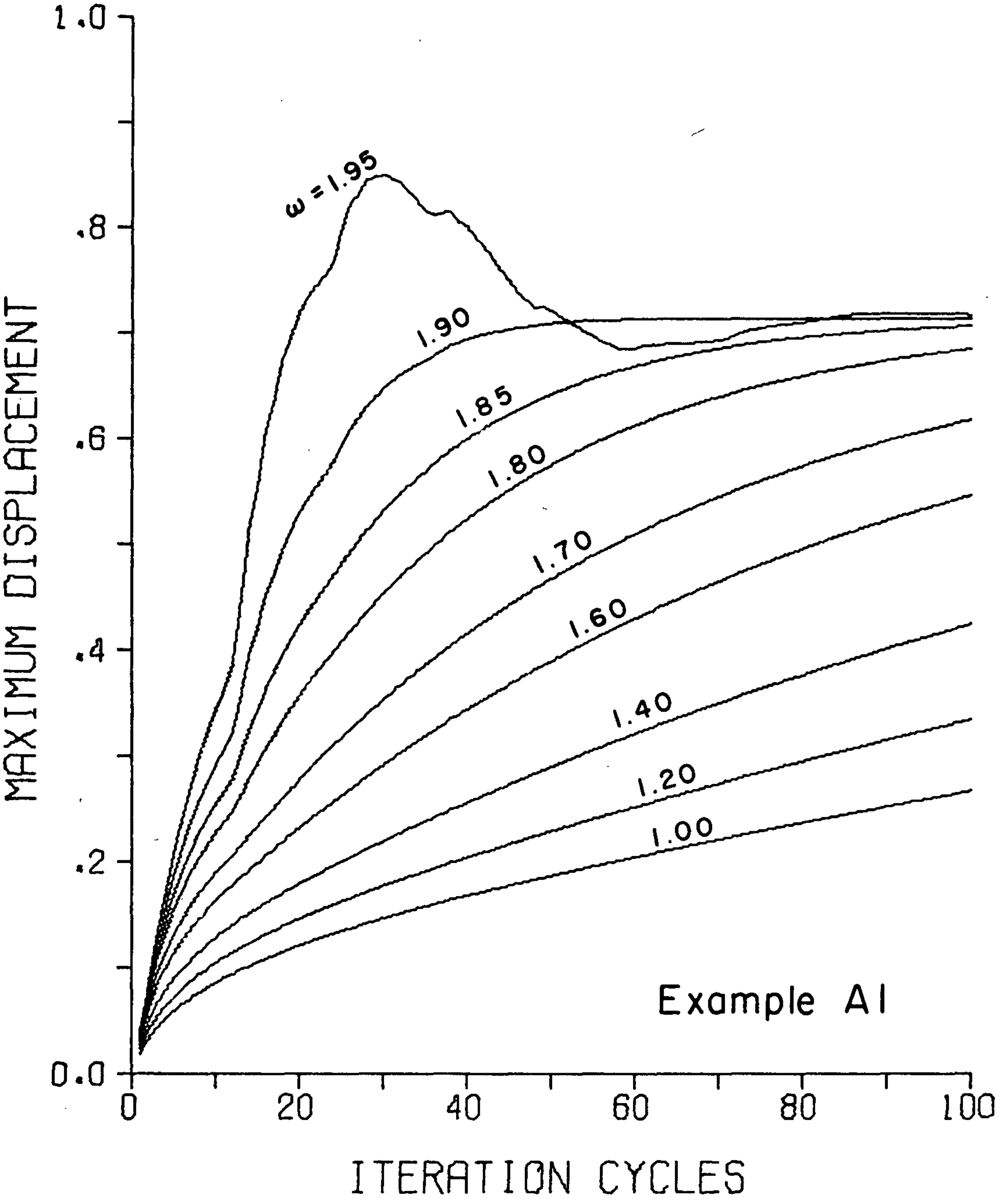


Fig. 4 Convergence of Block Jacobi Version E (Block Overrelaxation)

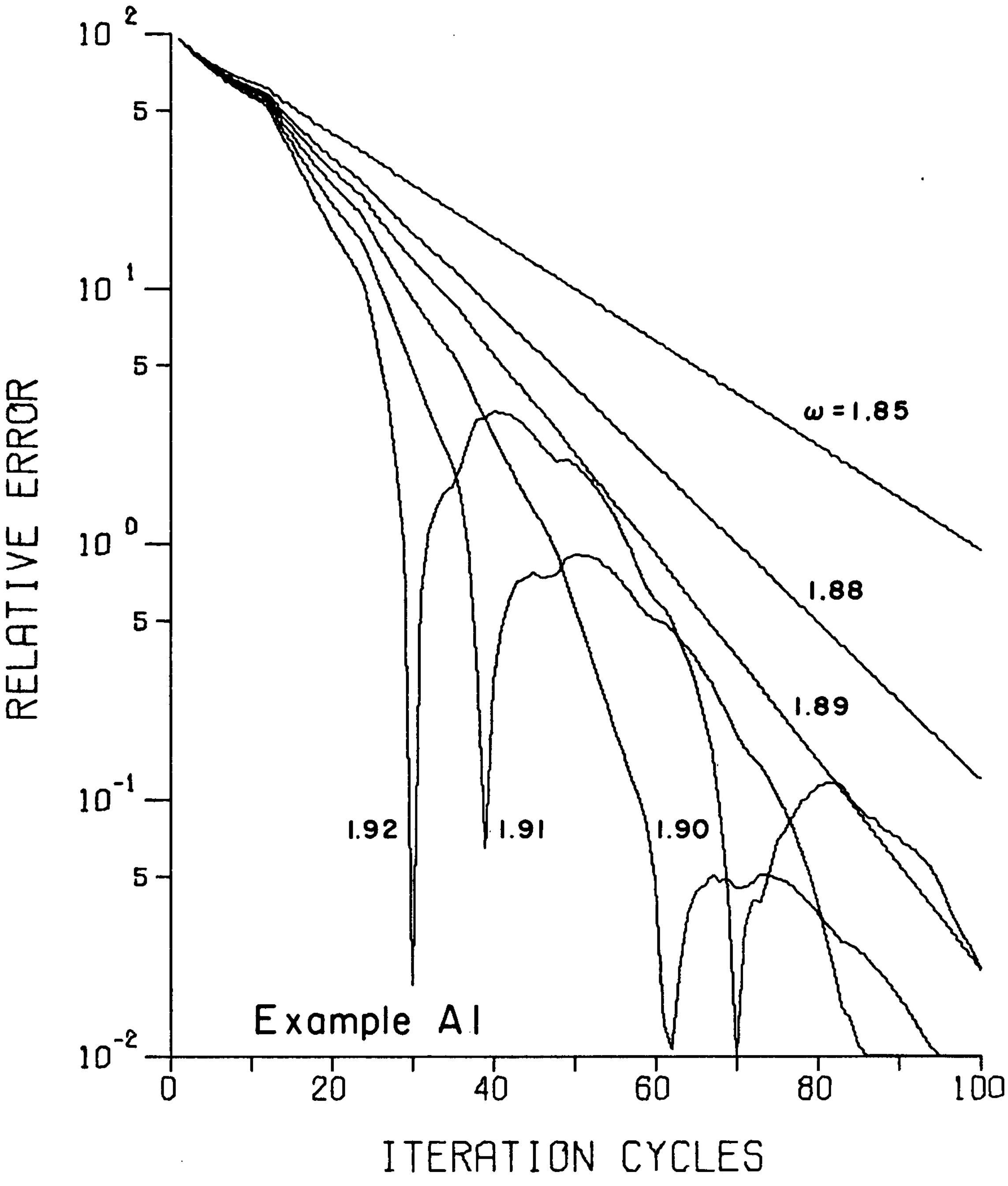


Fig. 5 Convergence of Block Jacobi Version E (Block Overrelaxation)

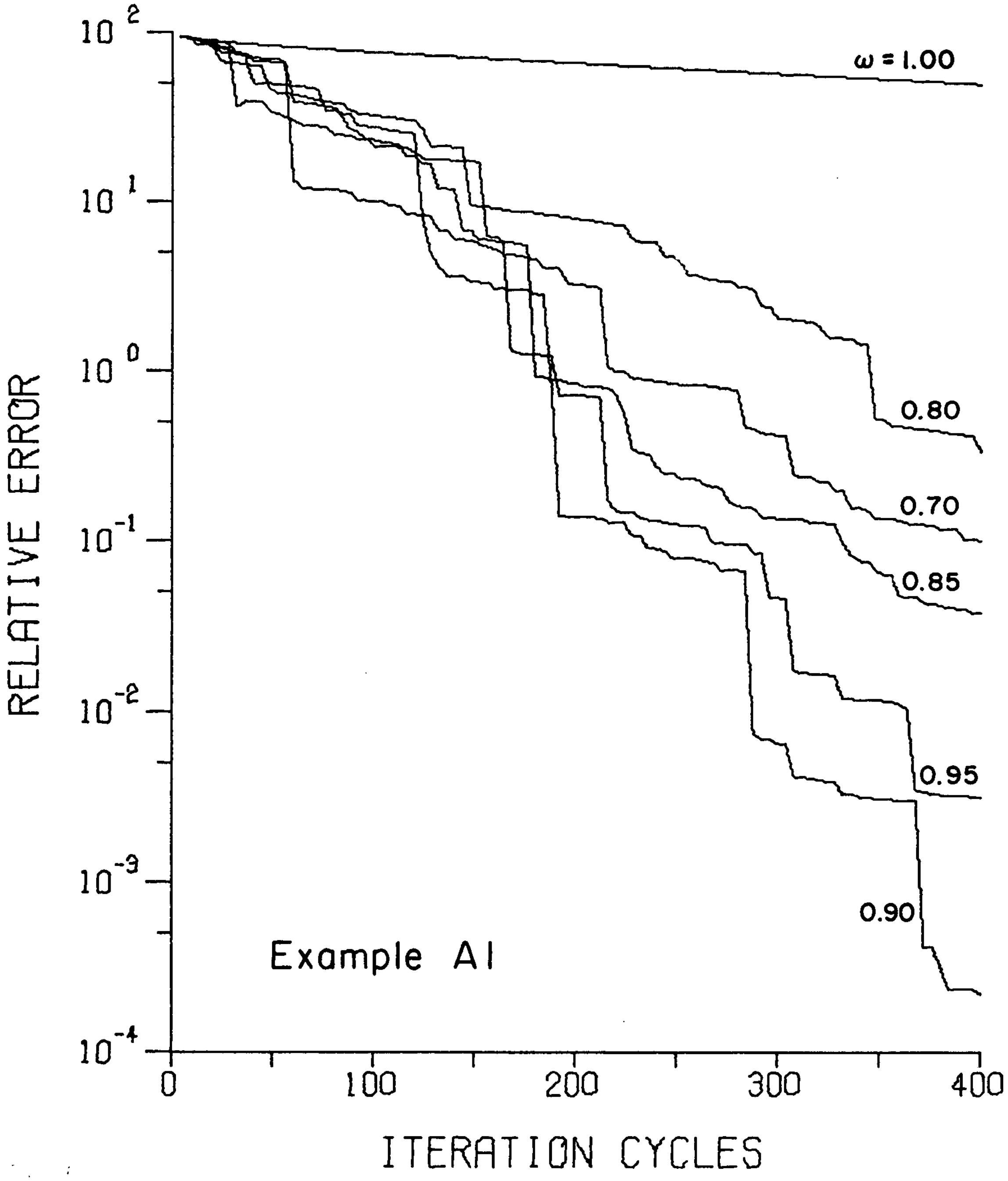


Fig. 6 Convergence of Almost Optimum Steepest Descent Method

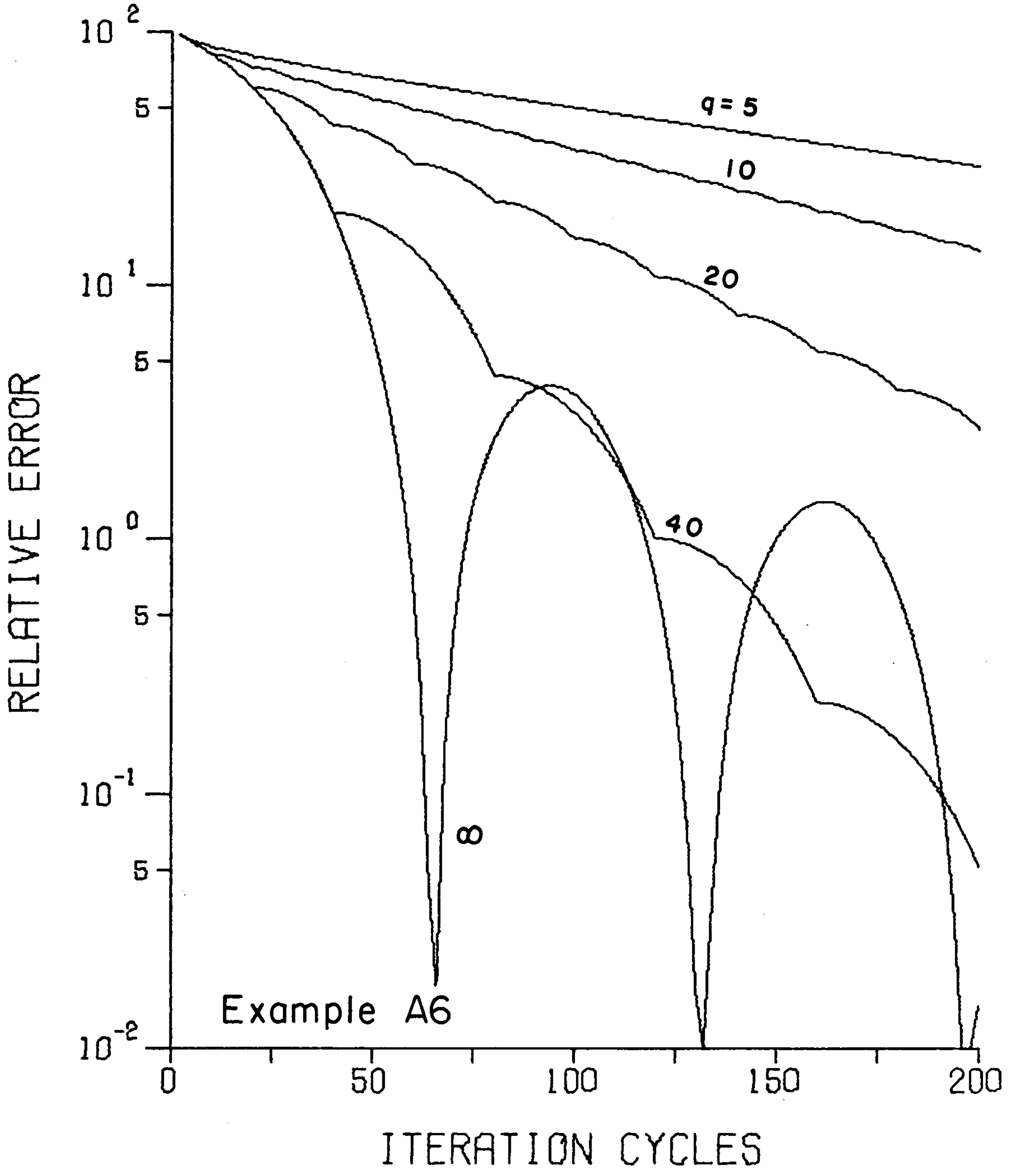


Fig. 7 Convergence of Lanczos' Method

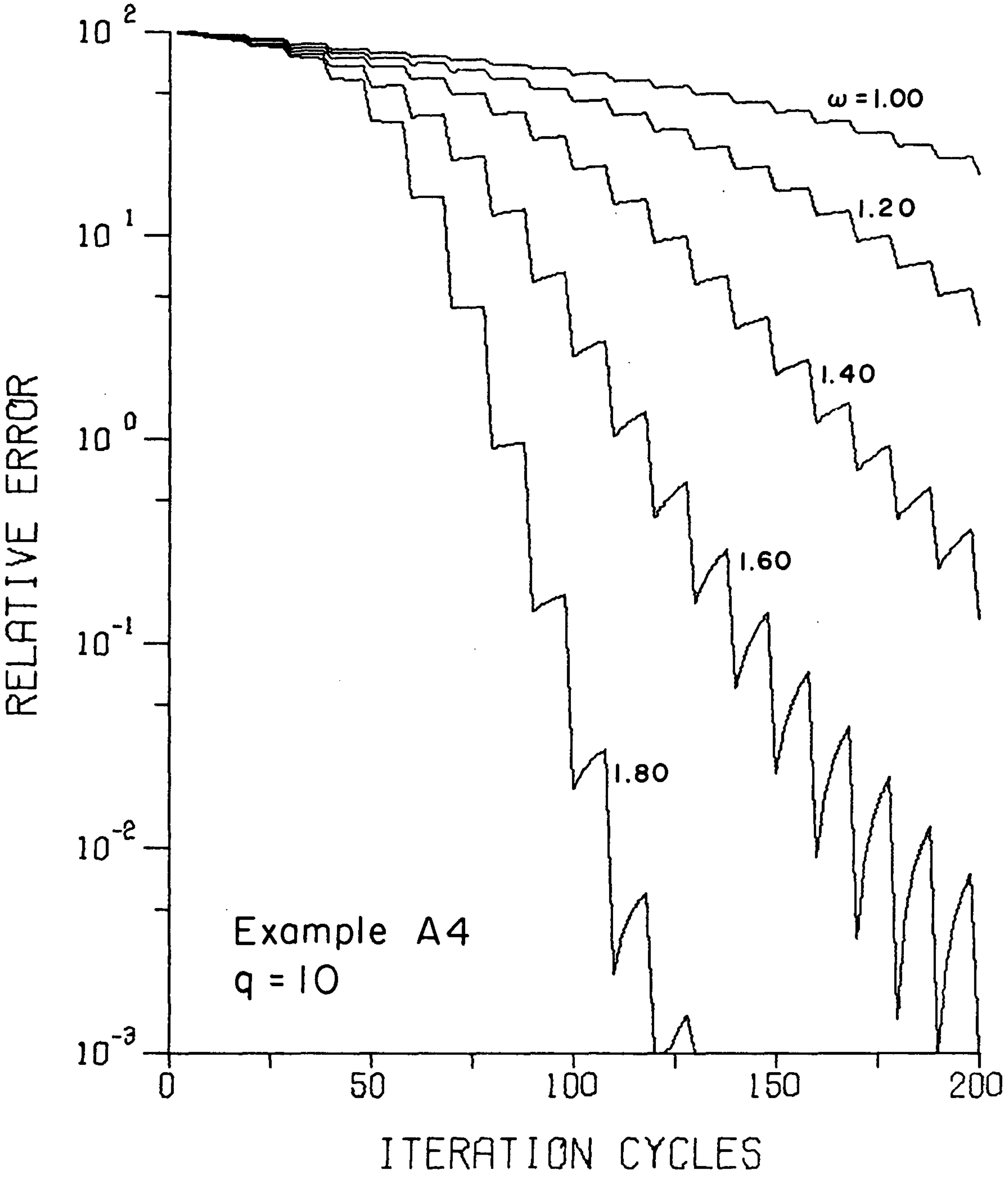


Fig. 8 Convergence of Wilson's Acceleration Applied to Block Overrelaxation

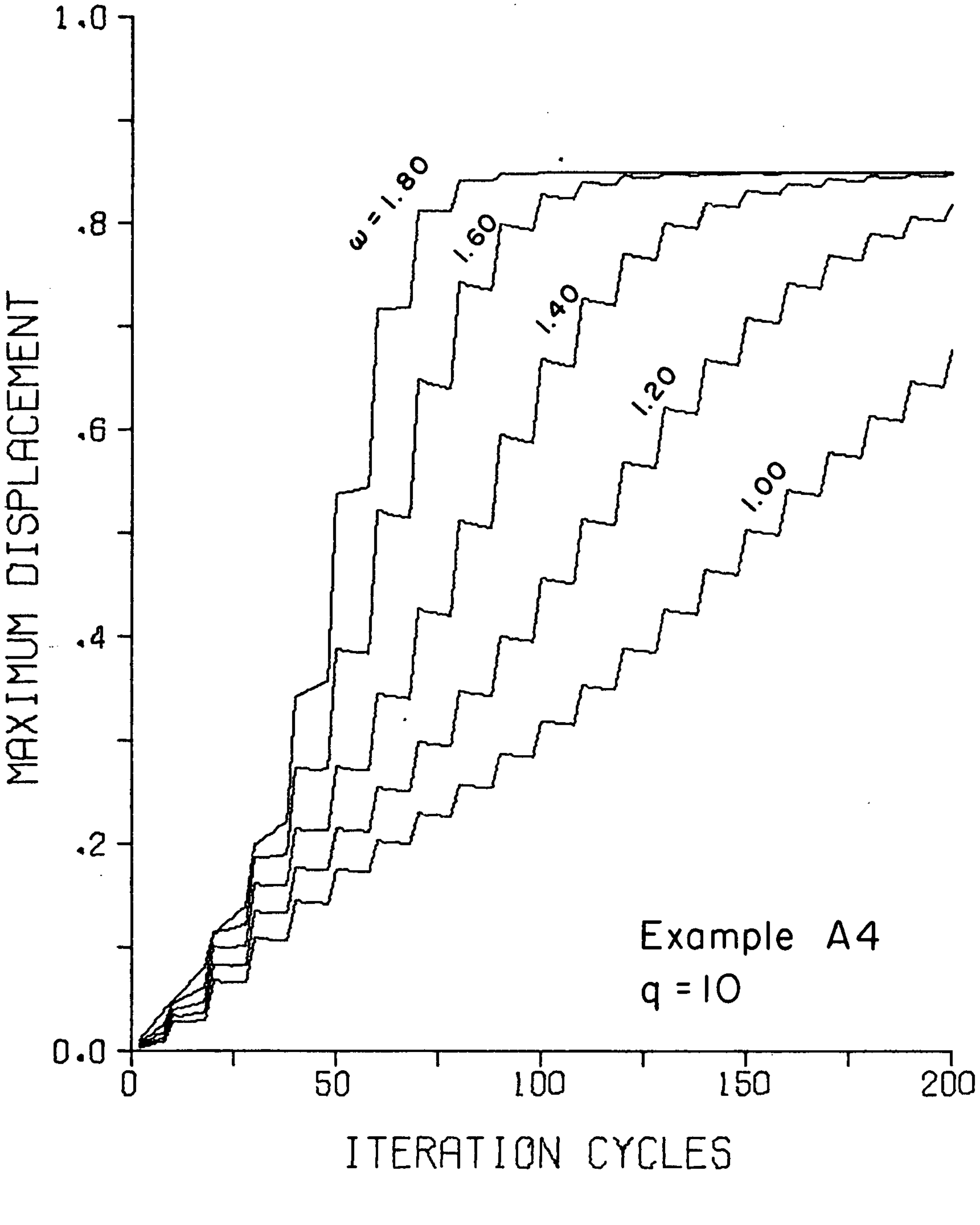


Fig. 9 Convergence of Wilson's Acceleration Applied to Block Overrelaxation

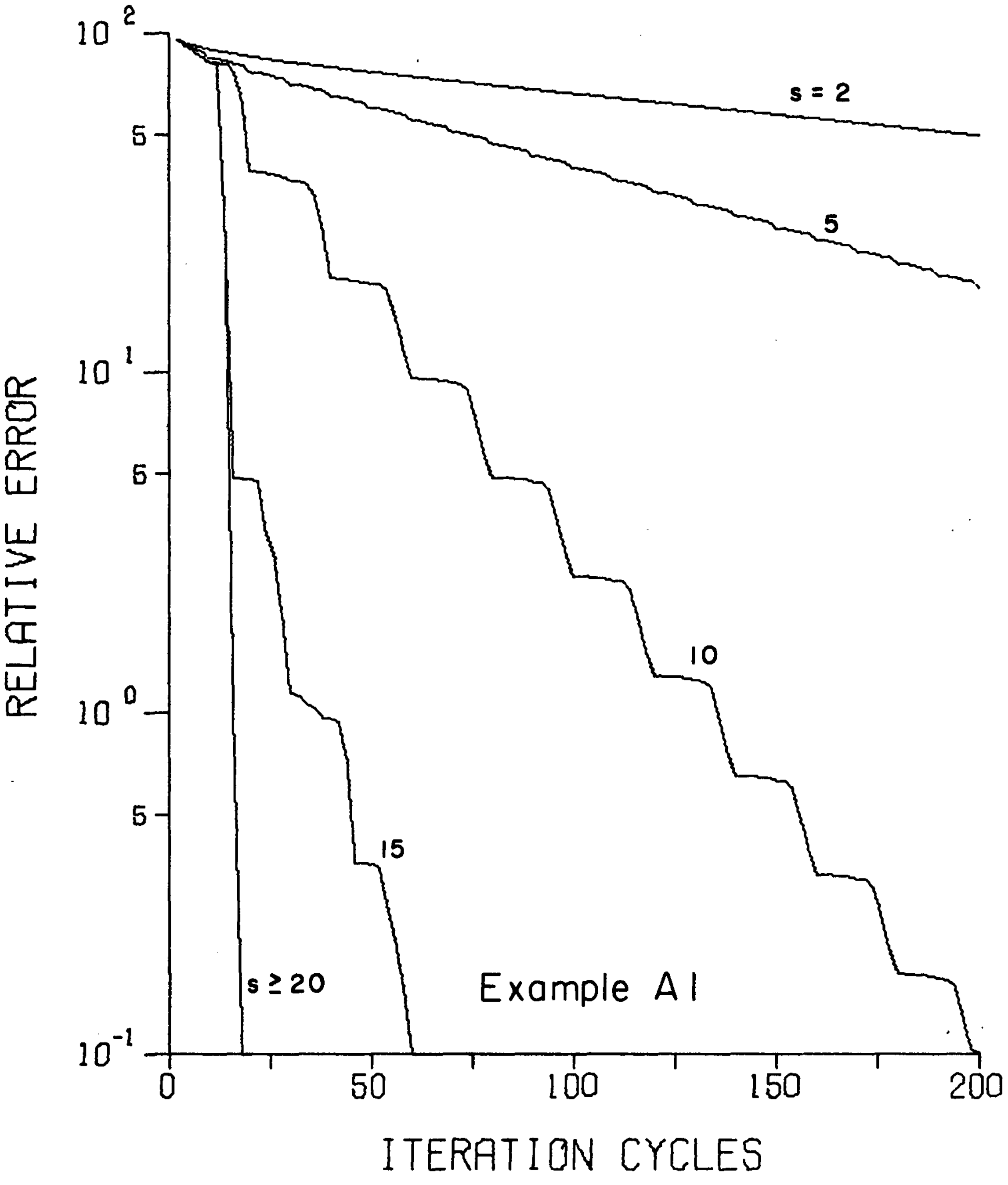


Fig. 10 Convergence of Conjugate Gradient Version B

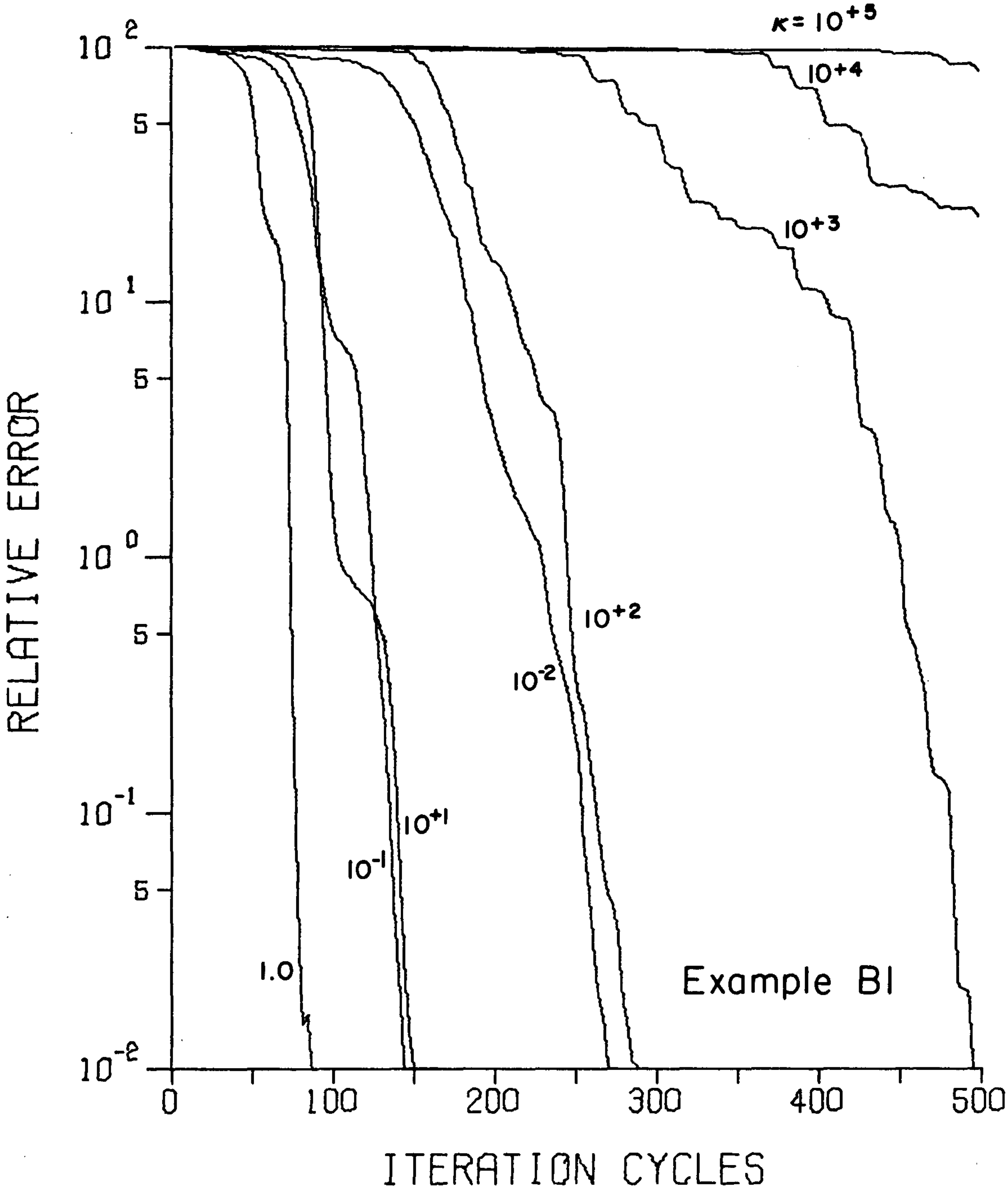


Fig. 11 Convergence of Conjugate Gradient Version A

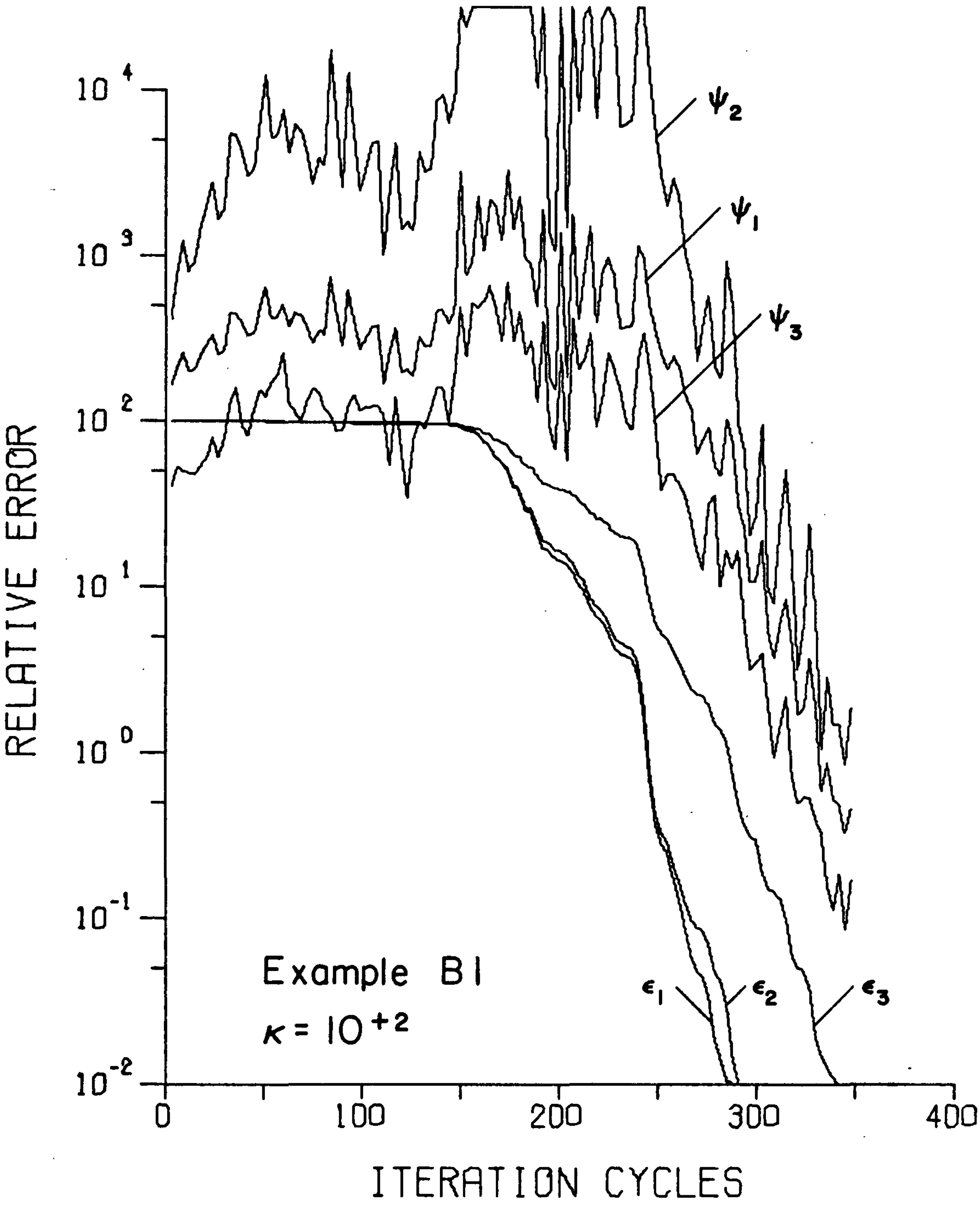


Fig. 12 Behavior of Error Predictors ψ_1, ψ_2, and ψ_3

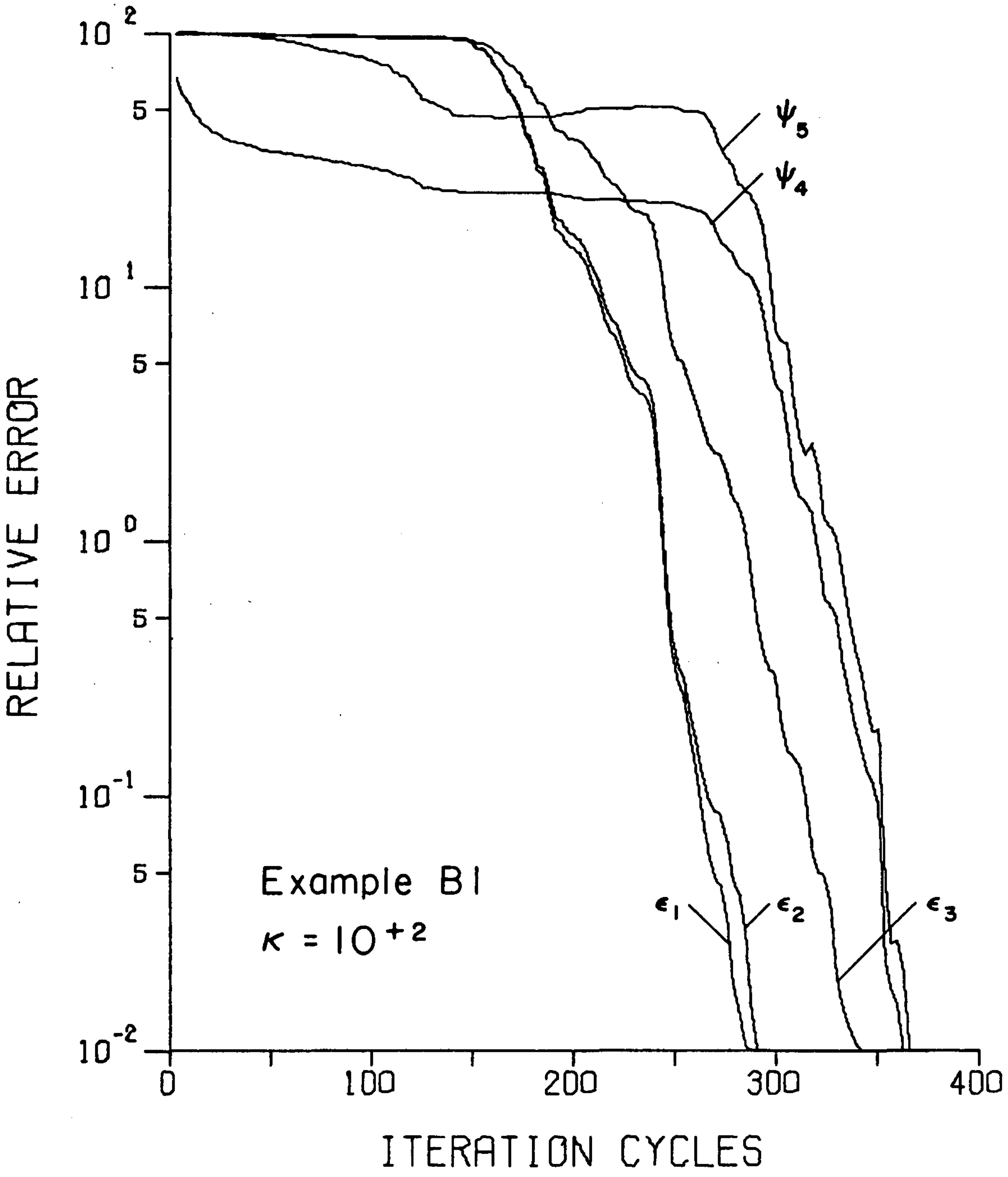


Fig. 13 Behavior of Error Predictors ψ_4 and ψ_5

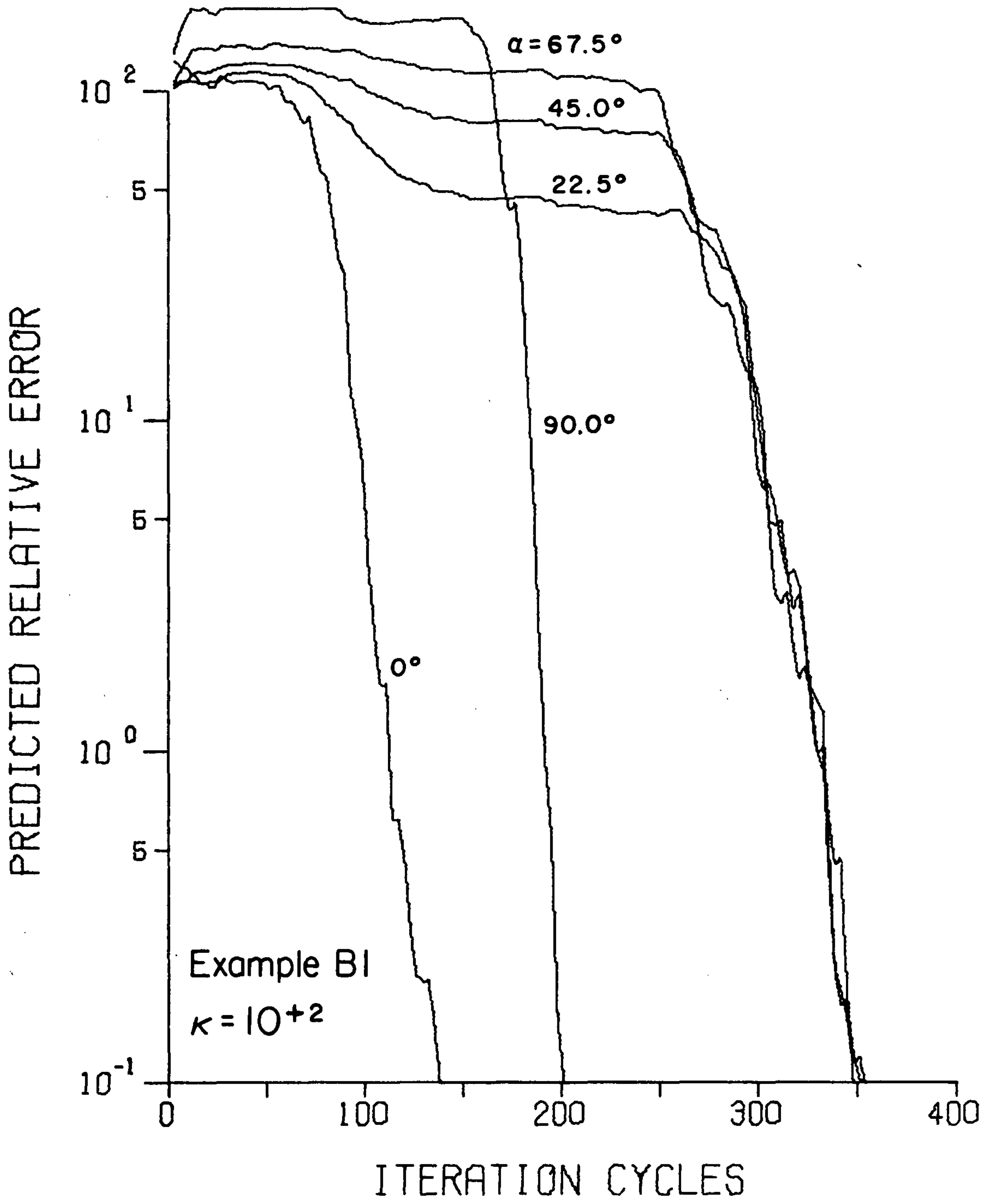


Fig. 14 Effect of Load Vector

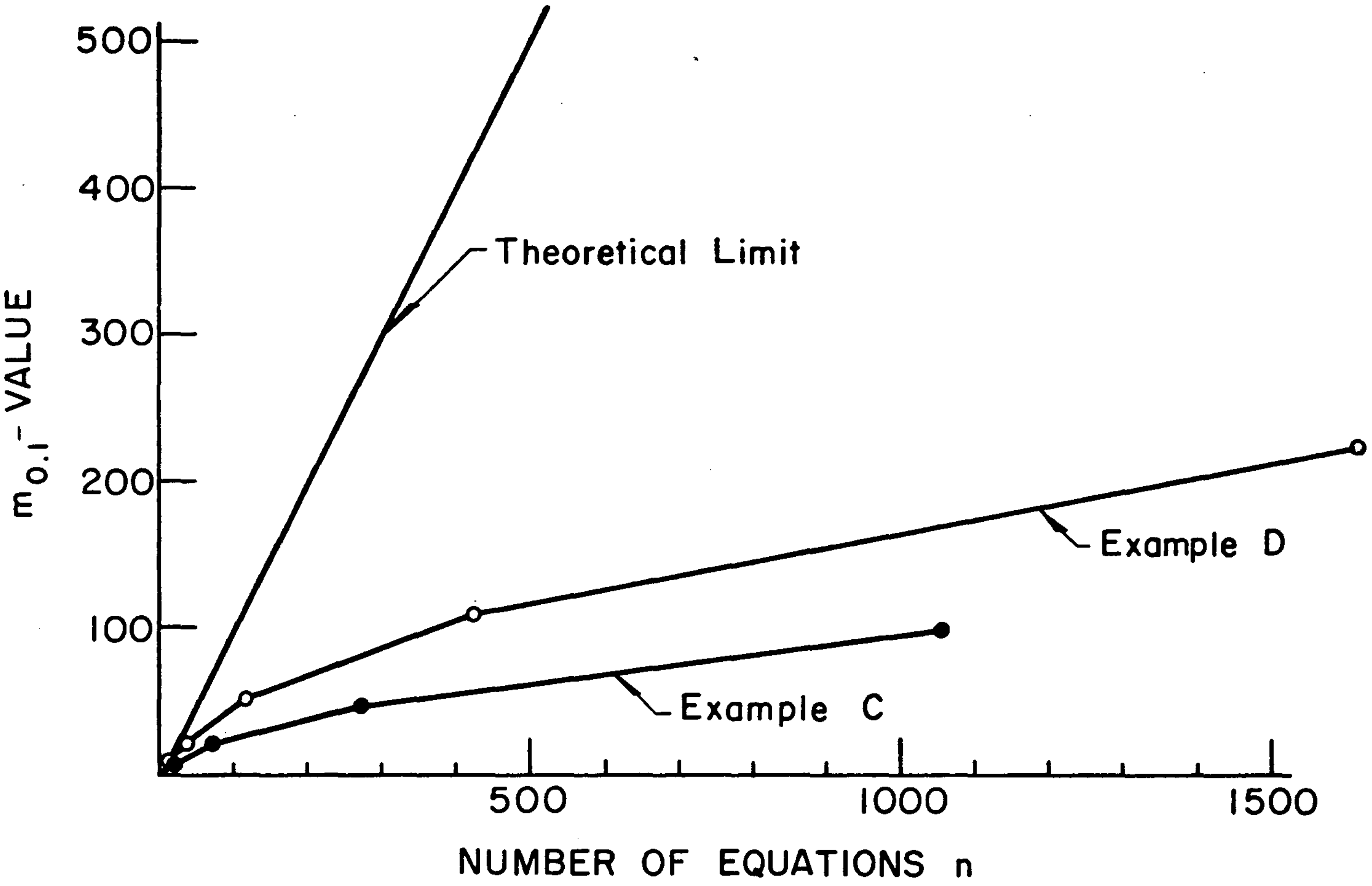


Fig. 15 Convergence of Large Systems of Equations

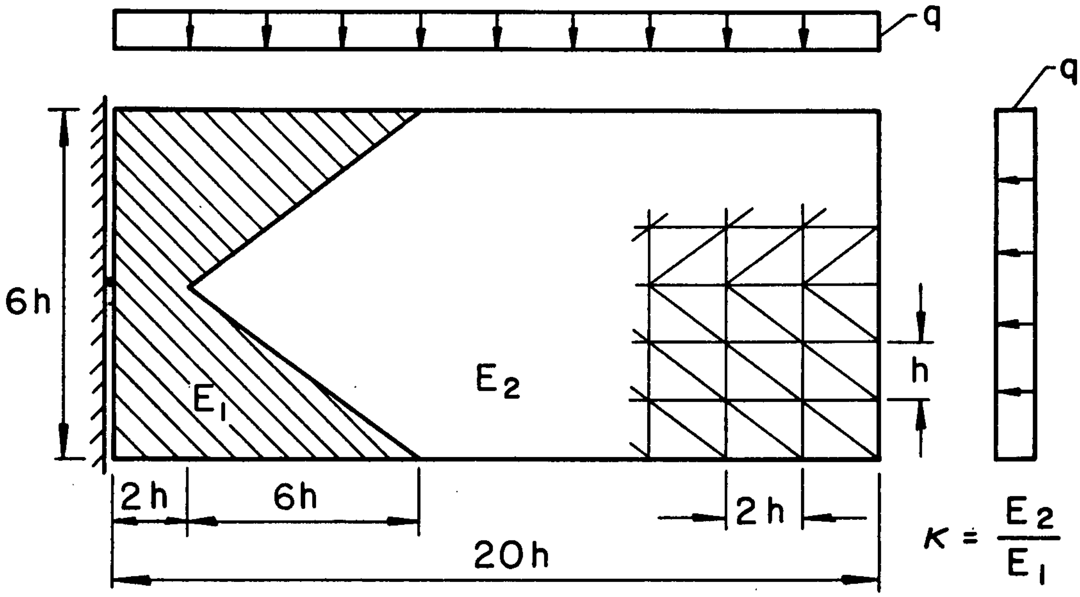


Fig. A.1 Test Example B1

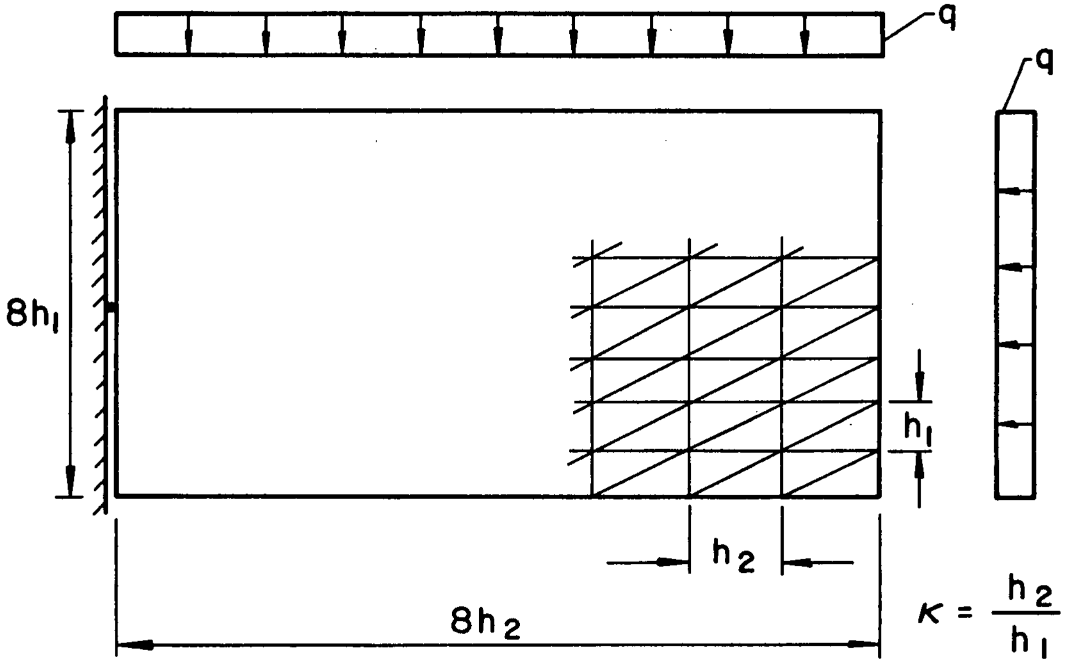


Fig. A.2 Test Example B2

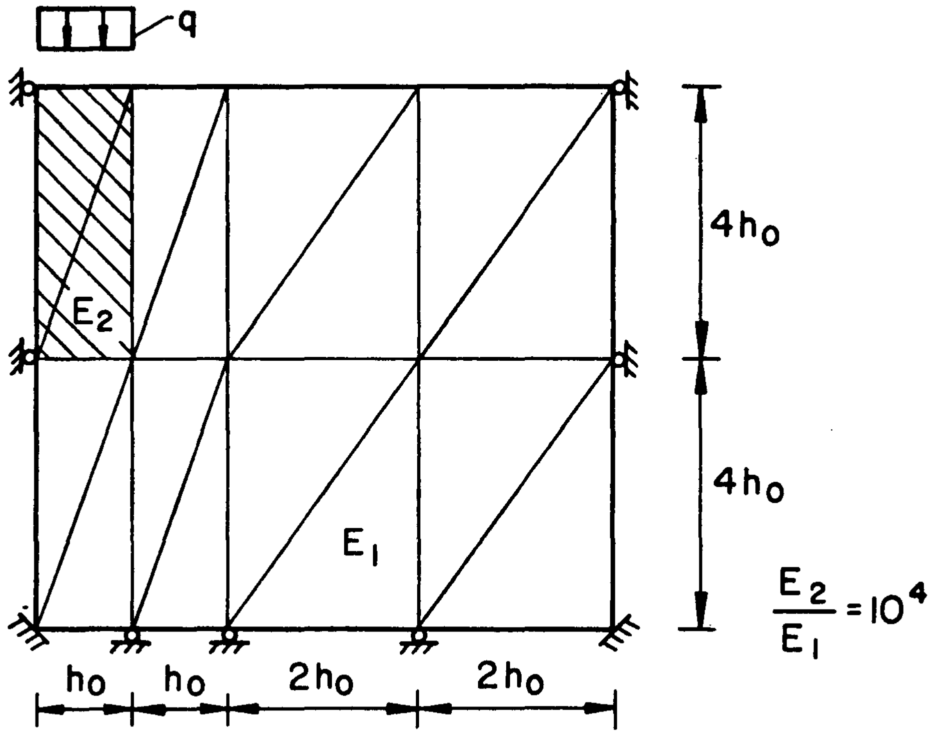


Fig. A.3 Test Example C_o

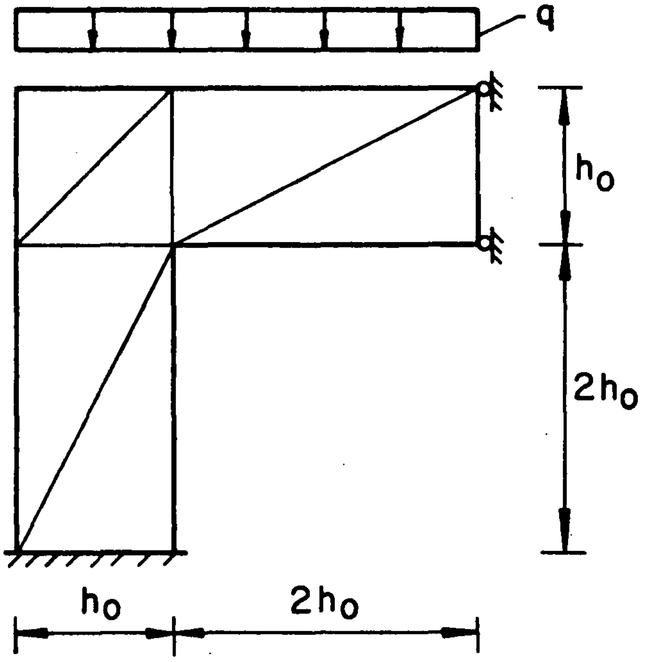


Fig. A.4 Test Example D_o

13. REFERENCES

1. Ahamed, S. V.
ACCELERATED CONVERGENCE OF NUMERICAL SOLUTION OF LINEAR AND NON-LINEAR VECTOR FIELD PROBLEMS, Computer Journal, Vol. 8, No. 1, pp. 73-76, 1965.

2. Albrecht, J.
FEHLERSCHRANKEN AND KONVERGENZBESCHLEUNIGUNG BEI EINER MONOTONEN ODER ALTERNIERENDEN ITERATIONSFOLGE, Numerische Mathematik, Vol. 4, No. 3, pp. 196-208, 1962.

3. ASCE SECOND CONFERENCE ON ELECTRONIC COMPUTATION, Pittsburgh, September 1960.

4. Beaubien, M. J. and Wexler, A.
ITERATIVE, FINITE DIFFERENCE SOLUTION OF INTERIOR EIGENVALUES AND EIGENFUNCTIONS OF LAPLACE'S OPERATOR, Computer Journal, Vol. 14, No. 3, pp. 263-269, 1971.

5. Beckman, F. S.
THE SOLUTION OF LINEAR EQUATIONS BY THE CONJUGATE GRADIENT METHOD, in Ref. 59, pp. 62-72.

6. Bellar, F. J.
AN ITERATIVE SOLUTION OF LARGE-SCALE SYSTEMS OF SIMULTANEOUS LINEAR EQUATIONS, Journal of the Society for Industrial and Applied Mathematics, Vol. 9, No. 2, pp. 189-193, 1961.

7. Bodewig, E.
MATRIX CALCULUS, 2nd ed., North-Holland Publishing Company, Amsterdam, 1955.

8. Booth, A. D.
NUMERICAL METHODS, Butterworths Scientific Publications, London, 1955.

9. Brew, J. S. and Brotton, D. M.
NON-LINEAR STRUCTURAL ANALYSIS BY DYNAMIC RELAXATION, International Journal for Numerical Methods in Engineering, Vol. 3, No. 4, pp. 463-483, 1971.

10. Brezinski, C.
ETUDES SUR LES ε- ET ρ-ALGORITHMES, Numerische Mathematik, Vol. 17, No. 2, pp. 153-162, 1971.

11. Cassell, A. C. and Hobbs, R. E.
DYNAMIC RELAXATION, in Ref. 83, pp. 787-808.

12. Clough, R. W., Wilson, E. L., and King, I. P.
LARGE CAPACITY MULTISTORY FRAME ANALYSIS PROGRAMS, Journal ASCE, Vol. 89, No. ST4, pp. 179-204, 1963.

13. Curtiss, J. H., (ed.)
NUMERICAL ANALYSIS, Proceedings of Symposia in Applied Mathematics, Vol. VI, 1956.

14. de la Vallee Poussin, F.
AN ACCELERATED RELAXATION ALGORITHM FOR ITERATIVE SOLUTION OF ELLIPTIC EQUATIONS, SIAM Journal on Numerical Analysis, Vol. 5, No. 2, pp. 340-351, 1968.

15. D'Sylva, E. and Miles, G. A.
THE S.S.O.R. ITERATION SCHEME FOR EQUATIONS WITH σ_1 ORDERING, Computer Journal, Vol. 6, No. 4, pp. 366-367, 1964.

16. Dupont, T.
A FACTORIZATION PROCEDURE FOR THE SOLUTION OF ELLIPTIC DIFFERENCE EQUATIONS, SIAM Journal on Numerical Analysis, Vol. 5, No. 4, pp. 753-782, 1968.

17. Engeli, M., Ginsburg, T., Rutishauser, H., and Stiefel, E.
REFINED ITERATIVE METHODS FOR COMPUTATION OF THE SOLUTION AND THE EIGENVALUES OF SELF-ADJOINT BOUNDARY VALUE PROBLEMS, Mitteilungen aus dem Institut für Angewandte Mathematik, ETH Zürich, No. 8, Birkhäuser Verlag, Basel, 1959.

18. Evans, D. J.
NOTE ON THE LINE OVER-RELAXATION FACTOR FOR SMALL MESH SIZE, Computer Journal, Vol. 5, No. 1, pp. 48-50, 1962.

19. Evans, D. J.
ESTIMATION OF THE LINE OVER-RELAXATION FACTOR AND CONVERGENCE RATES OF AN ALTERNATING DIRECTION LINE OVER-RELAXATION TECHNIQUE, Computer Journal, Vol. 7, No. 4, pp. 318-321, 1965.

20. Evans, D. J. and Forrington, C. V. D.
AN ITERATIVE PROCESS FOR OPTIMIZING SYMMETRIC SUCCESSIVE OVER-RELAXATION, Computer Journal, Vol. 6, No. 3, pp. 271-273, 1963.

21. Faddeev, D. K. and Faddeeva, V. N.
COMPUTATIONAL METHODS OF LINEAR ALGEBRA, W. H. Freeman, San Francisco, 1963.

22. Fix, G. J. and Larsen, K.
ON THE CONVERGENCE OF SOR ITERATIONS FOR FINITE ELEMENT APPROXIMATIONS TO ELLIPTIC BOUNDARY VALUE PROBLEMS, SIAM Journal on Numerical Analysis, Vol. 8, No. 3, pp. 536-547, 1971.

23. Forsythe, G. E.
SOLVING LINEAR EQUATIONS CAN BE INTERESTING, Bulletin of the American Mathematical Society, Vol. 59, No. 4, pp. 299-329, 1953.

24. Forsythe, A. I. and Forsythe, G. E.
PUNCHED-CARD EXPERIMENTS WITH ACCELERATED GRADIENT METHODS FOR LINEAR EQUATIONS, in Ref. 77, pp. 55-69.

25. Forsythe, G. E. and Wasow, W. R.
FINITE-DIFFERENCE METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS, J. Wiley, New York, 1960.

26. Fox, L.
AN INTRODUCTION TO NUMERICAL LINEAR ALGEGRA, Oxford University Press, New York, 1965.

27. Fox, R. L. and Stanton, E. L.
DEVELOPMENTS IN STRUCTURAL ANALYSIS BY DIRECT ENERGY MINIMIZATION, AIAA Journal, Vol. 6, No. 6, pp. 1036-1042, 1968.

28. Fried, I.
A GRADIENT COMPUTATIONAL PROCEDURE FOR THE SOLUTION OF LARGE PROBLEMS ARISING FROM THE FINITE ELEMENT DISCRETIZATION METHOD, International Journal for Numerical Methods in Engineering, Vol. 2, No. 4, pp. 477-494, 1970.

29. Fried, I.
DISCRETIZATION AND COMPUTATIONAL ERRORS IN HIGH-ORDER FINITE ELEMENTS, AIAA Journal, Vol. 9, No. 10, pp. 2071-2073, 1971.

30. Fried, I.
CONDITION OF FINITE ELEMENT MATRICES GENERATED FROM NONUNIFORM MESHES, AIAA Journal, Vol. 10, No. 2, pp. 219-221, 1972.

31. FURTHER CONTRIBUTIONS TO THE SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS AND THE DETERMINATION OF EIGENVALUES, National Bureau of Standards, Applied Mathematics Series, No. 49, 1958.

32. Garabedian, P. R.
ESTIMATION OF THE RELAXATION FACTOR FOR SMALL MESH SIZE, Mathematical Tables and Other Aides to Computation, Vol. 10, No. 56, pp. 183-185, 1956.

33. Gastinel, M. N.
PROCÉDÉ ITÉRATIF POUR LA RÉSOLUTION NUMÉRIQUE D'UN SYSTÈME D'ÉQUATIONS LINÉAIRES, Comptes Rendus, Vol. 246, pp. 2571-2574, 1958.

34. Ginsburg, T.
THE CONJUGATE GRADIENT METHOD, Numerische Mathematik, Vol. 5, No. 2, pp. 191-200, 1963.

35. Gladwell, G. M. L., (ed.)
COMPUTER AIDED ENGINEERING, Proceedings of Symposium, University of Waterloo, May 1971.

36. Griffin, D. S. and Kellogg, R. B.
A NUMERICAL SOLUTION FOR AXIALLY SYMMETRICAL AND PLANE ELASTICITY PROBLEMS, International Journal of Solids and Structures, Vol. 3, No. 5, pp. 781-794, 1967.

37. Hestenes, M. R.
THE CONJUGATE-GRADIENT METHOD FOR SOVLING LINEAR SYSTEMS, in Ref. 13, pp. 83-102.

38. 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, pp. 409-436, 1952.

39. Hodgkins, W. R.
ON THE RELATION BETWEEN DYNAMIC RELAXATION AND SEMI-ITERATIVE MATRIX METHODS, Numerische Mathematik, Vol. 9, No. 5, pp. 446-451, 1967.

40. Householder, A. S.
PRINCIPLES OF NUMERICAL ANALYSIS, McGraw-Hill, New York, 1953.

41. Householder, A. S.
THE GEOMETRY OF SOME ITERATIVE METHODS OF SOLVING LINEAR SYSTEMS, in Ref. 58, pp. 35-37.

42. Householder, A. S. and Bauer, F. L.
ON CERTAIN ITERATIVE METHODS FOR SOLVING LINEAR SYSTEMS, Numerische Mathematik, Vol. 2, No. 2, pp. 55-59, 1960.

43. Irons, B. M. and Tuck, R. C.
A VERSION OF THE AITKEN ACCELERATOR FOR COMPUTER ITERATION, International Journal of Numerical Methods in Engineering, Vol. 1, No. 3, pp. 275-277, 1969.

44. Isaacson, E. and Keller, H. B.
ANALYSIS OF NUMERICAL METHODS, J. Wiley, New York, 1966.

45. Ishibashi, S.
A METHOD OF ACCELERATING CONVERGENCE OF ITERATION, in Ref. 76, pp. 63-72.

46. Jensen, H. G. and Parks, G. A.
EFFICIENT SOLUTIONS FOR LINEAR MATRIX EQUATIONS, Journal ASCE, Vol. 96, No. ST1, pp. 49-64, 1970.

47. John, F.
LECTURES ON ADVANCED NUMERICAL ANALYSIS, Gordon and Breach, New York, 1967.

48. Kamel, H. A., Liu, D., and White, E. I.
THE COMPUTER IN SHIP STRUCTURE DESIGN, in Ref. 57.

49. Kelsey, S., Lee, K. N., and Mak, C. K. K.
THE CONDITION OF SOME FINITE ELEMENT COEFFICIENT MATRICES, in Ref. 35, pp. 267-283.

50. Khabaza, I. M.
AN ITERATIVE LEAST-SQUARE METHOD SUITABLE FOR SOLVING LARGE SPARSE MATRICES, Computer Journal, Vol. 6, No. 2, pp. 202-206, 1963.

51. Lanczos, C.
SOLUTION OF SYSTEMS OF LINEAR EQUATIONS BY MINIMIZED ITERATIONS, Journal of Research of the National Bureau of Standards, Vol. 49, No. 1, pp. 33-53, 1952.

52. Langer, R. E., (ed.)
BOUNDARY PROBLEMS IN DIFFERENTIAL EQUATIONS, University of Wisconsin Press, Madison, 1960.

53. Luenberger, D. G.
THE CONJUGATE RESIDUAL METHOD FOR CONSTRAINED MINIMIZATION PROBLEMS, SIAM Journal on Numerical Analysis, Vol. 7, No. 3, pp. 390-398, 1970.

54. Lynn, M. S.
ON THE EQUIVALENCE OF SOR, SSOR AND USSOR AS APPLIED TO σ_1-ORDERED SYSTEMS OF LINEAR EQUATIONS, Computer Journal, Vol. 7, No. 1, pp. 72-75, 1964.

55. Melosh, R. J. and Bamford, R. M.
EFFICIENT SOLUTION OF LOAD-DEFLECTION EQUATIONS, Journal ASCE, Vol. 95, No. ST4, pp. 661-676, 1969.

56. Milne, W. E.
NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS, Dover Publications, New York, 1970.

57. Office of Naval Research
INTERNATIONAL SYMPOSIUM ON NUMERICAL AND COMPUTER METHODS IN STRUCTURAL MECHANICS, Urbana, September 1971.

58. Paige, L. J. and Taussky, O., (ed.)
SIMULTANEOUS LINEAR EQUATIONS AND THE DETERMINATION OF EIGENVALUES, National Bureau of Standards, Applied Mathematics Series, No. 29, 1953.

59. Ralston, A. and Wilf, H. S., (ed.)
MATHEMATICAL METHODS FOR DIGITAL COMPUTERS, J. Wiley, New York, Vol. I, 1962, Vol. 2, 1967.

60. Rashid, Y. R.
SOLUTION OF ELASTO-STATIC BOUNDARY VALUE PROBLEMS BY THE FINITE ELEMENT METHOD, Ph.D. Dissertation, University of California, Berkeley, 1964.

61. Rashid, Y. R.
THREE-DIMENSIONAL ANALYSIS OF ELASTIC SOLIDS, International Journal of Solids and Structures,
Part I: Vol. 5, No. 12, pp. 1311-1331, 1969;
Part II: Vol. 6, No. 1, pp. 195-207, 1970.

62. Rashid, Y. R., Smith, P. D., and Prince, N.
ON FURTHER APPLICATION OF THE FINITE ELEMENT METHOD TO THREE-DIMENSIONAL ELASTIC ANALYSIS, in Ref. 83, pp. 433-453.

63. Reid, J. K.
A METHOD FOR FINDING THE OPTIMUM SUCCESSIVE OVER-RELAXATION PARAMETER, Computer Journal, Vol. 9, No. 2, pp. 200-204, 1966.

64. Rosanoff, R. A. and Ginsburg, T. A.
MATRIX ERROR ANALYSIS FOR ENGINEERS, Proceedings of Conference on Matrix Methods in Structural Mechanics, AFFDL-TR-66-80, pp. 887-910, Wright-Patterson Air Force Base, Dayton, October 1965.

65. Rosanoff, R. A., Gloudemann, J. F., and Levy, S.
NUMERICAL CONDITIONING OF STIFFNESS MATRIX FORMULATIONS FOR FRAME STRUCTURES, Proceedings of the Second Conference on Matrix Methods in Structural Mechanics, AFFDL-TR-68-150, pp. 1029-1060, Wright-Patterson Air Force Base, Dayton, October 1968.

66. Roy, J. R.
NUMERICAL ERROR IN STRUCTURAL SOLUTIONS, Journal ASCE, Vol. 97, No. ST4, pp. 1039-1054, 1971.

67. Rubinstein, M. F. and Wikholm, D. E.
ANALYSIS BY GROUP ITERATION USING SUBSTRUCTURES, Journal ASCE, Vol. 94, No. ST2, pp. 363-375, 1968.

68. Schmidt, J. W.
AUSGANGSVEKTOREN FÜR MONOTONE ITERATIONEN BEI LINEAREN GLEICHUNGSSYSTEMEN, Numerische Mathematik, Vol. 6, No. 2, pp. 78-88, 1964.

69. Schrem, E.
COMPUTER IMPLEMENTATION OF THE FINITE ELEMENT PROCEDURE, in Ref. 57.

70. Shanks, D.
NON-LINEAR TRANSFORMATIONS OF DIVERGENT AND SLOWLY CONVERGENT SEQUENCES, Journal of Mathematics and Physics, Vol. 34, No. 1, pp. 1-42, 1955.

71. Sheldon, J. W.
ITERATIVE METHODS FOR THE SOLUTION OF ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS, in Ref. 71, pp. 144-156.

72. Stein, M. L.
GRADIENT METHODS IN THE SOLUTION OF SYSTEMS OF LINEAR EQUATIONS, Journal of Research of the National Bureau of Standards, Vol. 48, No. 6, pp. 407-413, 1952.

73. Stiefel, E. L.
ÜBER EINIGE METHODEN DER RELAXATIONSRECHNUNG, Zeitschrift für Angewandte Mathemati und Physik, Vol. 3, No. 1, pp. 1-33, 1952.

74. Stiefel, E. L.
RELAXATIONSMETHODEN BESTER STRATEGIE ZUR LÖSUNG LINEARER GLEICHUNGSSYSTEME, Commentarii Mathematici Helvetici, Vol. 29, pp. 157-179, 1955.

75. Stiefel, E. L.
KERNEL POLYNOMIALS IN LINEAR ALGEBRA AND THEIR NUMERICAL APPLICATIONS, in Ref. 31, pp. 1-22.

76. Tanaka, H. and Kanamata, S., (ed.)
RECENT RESEARCHES OF STRUCTURAL MECHANICS, Uno Shoten, Tokyo, 1968.

77. Taussky, O., (ed.)
CONTRIBUTIONS TO THE SOLUTION OF SYSTEMS OF LINEAR EQUATIONS AND THE DETERMINATION OF EIGENVALUES, National Bureau of Standards, Applied Mathematics Series, No. 39, 1954.

78. Taylor, P. J.
A GENERALIZATION OF SYSTEMATIC RELAXATION METHODS FOR CONSISTENTLY ORDERED MATRICES, Numerische Mathematik, Vol. 13, No. 5, pp. 377-395, 1969.

79. Tewarson, R. P.
PROJECTION METHODS FOR SOLVING SPARSE LINEAR SYSTEMS, Computer Journal, Vol. 12, No. 1, pp. 77-80, 1969.

80. Tewarson, R. P.
COMPUTATION WITH SPARSE MATRICES, SIAM Review, Vol. 12, No. 4, pp. 527-543, 1970.

81. Varga, R. S.
FACTORIZATION AND NORMALIZED ITERATIVE METHODS, in Ref. 52, pp. 121-142.

82. Varga, R. S.
MATRIX ITERATIVE ANALYSIS, Prentice-Hall, Englewood Cliffs, 1962.

83. Veubeke, B. F. de, (ed.)
HIGH SPEED COMPUTING OF ELASTIC STRUCTURES, Proceedings of the Symposium of International Union of Theoretical and Applied Mechanics, Liege, August 1970.

84. Westlake, J. R.
A HANDBOOK OF NUMERICAL MATRIX INVERSION AND SOLUTION OF LINEAR EQUATIONS, J. Wiley, New York, 1968.

85. Willoughby, R. A., (ed.)
PROCEEDINGS OF THE SYMPOSIUM ON SPARSE MATRICES AND THEIR APPLICATIONS, IBM Watson Research Center, Yorktown Heights, September 1968.

86. Wilson, E. L.
FINITE ELEMENT ANALYSIS OF TWO-DIMENSIONAL STRUCTURES, Ph.D. Dissertation, University of California, Berkeley, 1963.

87. Wright, J. P. and Baron, M. L.
A SURVEY OF FINITE DIFFERENCE METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS, in Ref. 57.

88. Wuytack, L.
A NEW TECHNIQUE FOR RATIONAL EXTRAPOLATION TO THE LIMIT, Numerische Mathematik, Vol. 17, No. 3, pp. 215-221, 1971.

89. Wynn, P.
GENERAL PURPOSE VECTOR EPSILON ALGORITHM ALGOL PROCEDURES, Numerische Mathematik, Vol. 6, No. 1, pp. 22-36, 1964.

90. Yettram, A. L. and Hirst, M. J. S.
THE SOLUTION OF STRUCTURAL EQUILIBRIUM EQUATIONS BY THE CONJUGATE GRADIENT METHOD WITH PARTICULAR REFERENCE TO PLANE STRESS ANALYSIS, International Journal for Numerical Methods in Engineering, Vol. 3, No. 3, pp. 349-360, 1971.

91. Young, D.
ON THE SOLUTION OF LINEAR SYSTEMS BY ITERATIONS, in Ref. 13, pp. 283-298.

92. Young, E. and Ehrlich, L.
SOME NUMERICAL STUDIES OF ITERATIVE METHODS FOR SOLVING ELLIPTIC DIFFERENCE EQUATIONS, in Ref. 52, pp. 143-162.

93. Zienkiewicz, O. C.
THE FINITE ELEMENT METHOD IN ENGINEERING SCIENCE,
2nd ed., McGraw-Hill, London, 1971.

14. NOMENCLATURE

Matrices

D diagonal matrix containing the diagonal elements of K

D_1 diagonal matrix containing the diagonal elements of (KK)

D_2 quasi-diagonal matrix containing principal submatrices of K

D_s diagonal scaling transformation matrix

I identity matrix

K global stiffness matrix of the discretized structure, coefficient matrix of the system of linear equations

K_s coefficient matrix after scaling transformation

L lower triangular matrix containing the corresponding elements of K

L_1 lower triangular matrix containing the corresponding elements of (KK)

L_2 lower triangular matrix containing the corresponding elements of $(K-D_2)$

M_c error matrix-polynomial defining the total reduction of the initial error vector $e_c = M_c e_o$

S matrix defining the computational characteristics of linear stationary iterations

T iteration matrix for linear stationary iterations

T_c general iteration matrix relating two consecutive error vectors $e_c = T_c e_{c-1}$

Vectors

e_c error vector defined as the difference between the correct and the approximate solution vector $e_c = u-u_c$

e_i vector corresponding to the i-th column of the identity matrix

f nodal point load vector

f_s load vector after scaling transformation

r_c residual vector containing the unbalanced nodal point forces $r_c = f-Ku_c$

t_c direction vector of Gastinel-type nonlinear stationary iterations, defined by $(t)_i^c = \text{sign}[(r)_i^c]$

u nodal point displacement vector, solution vector of the system of linear equations

u_c approximate solution vector after c iteration cycles

u_s solution vector after scaling transformation

v_c K- or KK-orthogonal direction vector of conjugate gradient algorithms

w_c increment vector defined by $w_c = u_{c+1}-u_c$

Scalars

a lower bound for the minimum eigenvalue of a matrix

b upper bound for the maximum eigenvalue of a matrix

c iteration cycle counter

c_o cycle interval of oscillations

C_c c-th order Chebyshev polynomial of the first kind

E modulus of elasticity

g ratio of mesh spacing parameters

h mesh spacing parameter

$(K)_{ij}$ element in the i-th row and j-th column of K

m general measure for the required number of iteration cycles

$m_{0.1}$ number of iteration cycles necessary to reduce the relative error of the maximum nodal point displacement to a value below 0.1%

n total number of degrees of freedom of the discretized structure, size of the system of linear equations

n_e total number of elements

n_p total number of nodal points

n_s block size of linear stationary block iterations

p cycle interval for identifying previous approximations of the solution vector

P	condition number of a matrix defined by $P(K) = \lambda_{max}(K)/\lambda_{min}(K)$
P_1	upper bound for the P-condition number
q	cycle interval for restarting nonstationary solution procedures, length of the acceleration interval of nonlinear stationary accelerations
q_{opt}	optimum value of the parameter q
s	parameter of s-step gradient methods
$(u)_i^c$	i-th element of the vector u_c
$(u)_{max}^c$	approximate value of the maximum nodal point displacement after c iteration cycles
Z	measure of the total computational effort for solving a system of linear equations
α,β,γ	scalar quantities
$\varepsilon,\varepsilon_1 \ldots \varepsilon_3$	measures of the relative error of an approximate solution
ε_c	relative error of the maximum nodal point displacement after c iteration cycles
κ	parameter governing the condition of test examples B1 and B2
λ_i	i-th eigenvalue of a matrix
λ_{min}	minimum eigenvalue of a matrix
λ_{max}	maximum eigenvalue of a matrix
Π	total potential energy of the discretized structure
ρ_c	average rate of convergence

ρ_∞ asymptotic rate of convergence

σ_1, σ_2 parameters of the hypergeometric relaxation method

$\phi, \phi_1 \ldots \phi_3$ error functions

$\psi, \psi_1 \ldots \psi_5$ quantities for predicting the relative error of an approximate solution

ω (constant) acceleration factor of linear stationary accelerations

ω_{opt} optimum acceleration factor

ω' modified acceleration factor for Successive Approximation Version D, defined by $\omega' = \omega \cdot b_K$

Miscellaneous Symbols

A [ ] general operator for acceleration algorithms

c subscript for identifying the value of a matrix, vector, or scalar quantity after c iteration cycles

I [ ] general operator for iteration algorithms

L subscript for identifying the last iterate of an acceleration interval

$\| \;\; \|$ matrix or vector norms (Hölder norms) defined by

$$\|K\|_1 = \max_{j=1 \ldots n} \sum_{i=1}^{n} |(K)_{ij}|$$

$$\|K\|_2 = \max_{i=1 \ldots n} \sqrt{|\lambda_i (K^T K)|} \quad \text{spectral norm}$$

$$\|K\|_\infty = \max_{i=1 \ldots n} \sum_{j=1}^{n} |(K)_{ij}|$$

$$\|u\|_1 = \sum_{i=1}^{n} |(u)_i|$$

$$\|u\|_2 = \sqrt{\sum_{i=1}^{n} [(u)_i]^2}$$ euclidean length

$$\|u\|_\infty = \max_{i=1...n} |(u)_i|$$

for symmetric matrices: $\|K\|_1 = \|K\|_\infty$

$\|K\|_2 = \|K\|_{sr}$

$\| \; \|_{sr}$ spectral radius of a matrix, equivalent to the eigenvalue of largest (absolute) magnitude

$$\|K\|_{sr} = \max_{i=1...n} |\lambda_i(K)|$$

15. APPENDICES

Appendix I

Properties of Finite Element Stiffness Matrices

Global stiffness matrices arising in the stiffness formulation (or more precisely in the "assumed displacement" approach) of the finite element method exhibit certain general properties which are described in the following paragraphs. The primary purpose of this compilation of properties is to define the precise nature of the systems of equations whose iterative solution is investigated in this dissertation. At the same time, various general characteristics of the stiffness matrix K are described which may affect the applicability of a particular solution procedure.

(1) K is real, square, and symmetric. The symmetry of the stiffness matrix can be directly established on the basis of Betti-Maxwell's reciprocal theorem (Ref. 93), assuming that the load and displacement vectors associated with K are ordered in an identical manner.

(2) K is positive definite. By definition, a real, symmetric matrix is said to be positive definite if

the quadratic form u^TKu assumes values greater than zero for any arbitrary non-zero vector u (Refs. 21, 84). As the strain energy of the discretized structure, being a positive quantity for any non-zero displacement, differs from the above quadratic form only by a constant factor (Section 3.1.2, Ref. 93), the global stiffness matrix necessarily is positive definite. This particular property of K implies that the eigenvalues $\lambda_i(K)$ are real and positive, that all diagonal elements $(K)_{ii}$ are greater than zero, and that the stiffness matrix is nonsingular. In establishing the positive definite character of K it is assumed that the "essential" displacement boundary conditions, preventing rigid body motions of the structure, are specified. Otherwise, the global stiffness matrix remains semi-definite and, therefore, singular (cf. Chapter 9).

(3) <u>K is generally a large, sparse matrix</u>. The total size of the stiffness matrix depends on the number of nodal points as well as on the number of degrees of freedom per individual node. To a certain extent, the size of K is also affected by the displacement boundary conditions. The average number of non-zero elements in a row or column of K is essentially defined by the finite element type, whereas displace-

ment boundary conditions, the element mesh configuration, and the size of K have only a secondary effect on this number.

The population pattern of the stiffness matrix, i.e. the pattern in which the non-zero elements of K appear, is influenced by the nodal point enumeration, by the geometry of the structure, and by the way in which the components of the associated load and displacement vectors are arranged. For structural problems with regular, "chain"-like geometry it is possible to order the equations in such a way that the non-zero elements appear within a comparatively narrow band along the main diagonal. The band-structure of the coefficient matrix is extensively used for the purpose of reducing the storage requirements as well as the computational effort of direct solution procedures (Chapter 1). Storage requirements of iterative methods, however, are largely independent of the population pattern. For structures with irregular geometry and for less systematic enumeration schemes the band-character of the global stiffness matrix will be less pronounced. A similar situation occurs if the input data for various parts of the structure are generated independently or if subsequent modifications of the finite element mesh are performed without complete renumeration of all nodal points.

(4) K is generally irreducible. Under normal conditions it is not possible to transform K into a block triangular or (since K is symmetric) quasi-diagonal matrix by simultaneous row and column permutations (Ref. 82). Transformations of this type can only be carried out if the global stiffness matrix comprises two or more completely independent structural systems whose solution can be obtained by solving an equivalent number of lower order subsystems.

In addition to the above main characteristics of finite element stiffness matrices, the discussion also contains several properties normally found only in systems of equations arising from finite difference approximations of certain elliptic partial differential equations. Their special characteristics can be utilized in the determination of optimum acceleration factors, in the derivation of convergence conditions as well as in the efficient organization of iterative solution processes (Section 3.2). The additional properties are included in this compilation in order to indicate that they cannot be exploited in connection with finite element coefficient matrices.

(5) K is generally not diagonally dominant since the elements of the stiffness matrix normally do not satisfy the relationship (Ref. 82)

$$(K)_{ii} \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |(K)_{ij}| \quad i = 1 \ldots n \qquad \text{(I.1)}$$

(6) K is generally not a Stieltjes matrix, that is, its off-diagonal elements are not necessarily less than or equal to zero (Ref. 82).

(7) K generally does not satisfy "property A" and, in a more general sense, is not a consistently ordered p-cyclic matrix. These properties, whose precise nature is defined, for instance, in Refs. 17, 25, 82, and 84, form the basis for determining optimum acceleration factors for various overrelaxation methods (Section 3.2). As shown in Ref. 17, it is possible to transform any system of equations in such a way that the coefficient matrix assumes "block property A." For the given type of matrices this possibility is of little practical value since the transformation may cause a considerable increase in the storage requirements.

The numerical characteristics of a system of equations are largely defined by the eigenvalues of the coefficient matrix, in particular by the values of $\lambda_{min}(K)$

and $\lambda_{max}(K)$. Both quantities play an important role not only in the application of various iterative methods, but also in describing the condition of the coefficient matrix. The iterative determination of the extreme eigenvalues cannot be considered as practical since the required computational effort is likely to be of the same magnitude as that for actually solving the system of equations. It is, therefore, common practice to use simple "a priori" bounds of these quantities whose determination should require a minimum of numerical computations. In the remainder of this section various methods for establishing such bounds are reviewed as far as they apply to general finite element stiffness matrices.

A relatively close upper bound for the maximum eigenvalue of the stiffness matrix can be established on the basis of Gershgorin's theorem (Refs. 7,21,82,84)

$$\lambda_{max}(K) \leq \|K\|_1 = \|K\|_\infty = \max_{i=1...n} \sum_{j=1}^{n} |(K)_{ij}| \qquad (I.2)$$

Provided K is sufficiently large, the numerical value predicted by Eq. I.2 is smaller and, therefore, more accurate than Schur's estimate (Refs. 7,21,84)

$$\lambda_{max}(K) \leq \left[\sum_{i=1}^{n} \sum_{j=1}^{n} (K)_{ij}^{2} \right]^{1/2} \qquad (I.3)$$

as well as the trace of the stiffness matrix

$$\lambda_{max}(K) \leq \sum_{i=1}^{n} \lambda_i(K) = \sum_{i=1}^{n} (K)_{ii} \qquad (I.4)$$

A simple lower bound for $\lambda_{max}(K)$ is, on the other hand, given by

$$\lambda_{max}(K) \geq \max_{i=1...n} (K)_{ii} \qquad (I.5)$$

Computational experience indicates that the estimates of Eqs. I.2 and I.5 are relatively close since their numerical values usually differ by a factor less than 10 (Ref. 49).

Utilizing the same idea as in Eq. I.5, a simple upper bound for the minimum eigenvalue of the stiffness matrix is obtained from

$$\lambda_{min}(K) \leq \min_{i=1...n} (K)_{ii} \qquad (I.6)$$

The relationship provides reasonably close estimates only under exceptional conditions, for instance, if large differences in the stiffness properties of the structure and, therefore, in the magnitude of the diagonal elements $(K)_{ii}$ exist (Appendix 2, examples B1 and C). In general, however, the bound will be of little practical value.

In order to define the limits of the entire range of $\lambda_i(K)$-values, it is of considerably greater importance to establish "a priori" lower bounds of the minimum eigenvalue. Gershgorin's theorem (Refs. 7,21,81,84) cannot be used for this purpose since the relationship

$$\lambda_{\min}(K) \geq \min_{i=1...n} \left[(K)_{ii} - \sum_{\substack{j=1 \\ j \neq i}}^{n} |(K)_{ij}| \right] \tag{I.7}$$

results in trivial bounds less than zero as K is generally not a diagonally dominant matrix (p.253). Attempts to base a lower bound for $\lambda_{\min}(K)$ on the smallest eigenvalue of the undiscretized structure (Refs. 29,30,49) have to be considered unsuitable as well since the particular quantity is unavailable for all but the simplest structural systems. The fact that the eigenvalues of the coefficient matrix and those of its inverse are related by (Refs. 21,84)

$$\lambda_i(K) = \frac{1}{\lambda_i(K^{-1})} \tag{I.8}$$

allows the establishment of the following eigenvalue bound

$$\lambda_{\min}(K) \geq \frac{1}{\|K^{-1}\|_1} = \frac{1}{\|K^{-1}\|_\infty} = \frac{1}{\max\limits_{i=1...n} \sum\limits_{j=1}^{n} |(K^{-1})_{ij}|} \tag{I.9}$$

Since the inverse of the coefficient matrix remains generally unknown, Eq. I.9 is, however, only of theoretical interest (Appendix 2).

Summarizing the various possibilities, it has to be concluded that useful "a priori" bounds for the minimum eigenvalue of the stiffness matrix are not available in practice. However, a relatively close "a posteriori" bound of this quantity can be obtained provided the solution vector u is known

$$\lambda_{min}(K) \leq \frac{f^T u}{u^T u} = \frac{u^T K u}{u^T u} \qquad (I.10)$$

The numerical value predicted by Eq. I.10 represents a comparatively accurate estimate of $\lambda_{min}(K)$ if the deflected shape of the structure resembles its lowest vibrational eigenmode. A similarity between the solution vector u and the eigenvector corresponding to $\lambda_{min}(K)$ is usually found for structural systems primarily subjected to bending (Appendix 2, examples A3, A4, A5, B1, B2). In cases where both vectors are orthogonal, however, Eq. I.10 may result in relatively coarse approximations (cf. Chapter 9).

The convergence behavior of iterative solution procedures as well as the accumulation of roundoff errors is primarily affected by the condition of the given system of equations. Generally it is difficult to define appropriate criteria by which the condition of a matrix could be accurately described in the form of a single numerical quantity. Various investigations indicate, however, that the P-condition number defined by

$$P(K) = \frac{\lambda_{max}(K)}{\lambda_{min}(K)} = \|K\|_2 \; \|K^{-1}\|_2 \qquad (I.11)$$

represents a suitable measure of the relevant numerical characteristics (Refs. 7,21,64,84). Since the determination of the extreme eigenvalues requires a considerable amount of computational effort, Eq. I.11 is frequently replaced by simpler, equally suitable expressions such as (Ref. 64)

$$P_1(K) = \|K\|_1 \; \|K^{-1}\|_1 = \|K\|_\infty \; \|K^{-1}\|_\infty \geq P(K) \qquad (I.12)$$

The absence of practically useful lower bounds for $\lambda_{min}(K)$ makes it impossible to establish rigorous "a priori" bounds for the P-condition number of finite element stiffness matrices. "A posteriori" estimates, however, can be obtained by combining the eigenvalue bounds of Eqs. I.2 and I.10

$$P(K) \approx \left[\max_{i=1\ldots n} \sum_{j=1}^{n} |(K)_{ij}| \right] \frac{u^T u}{f^T u} \qquad (I.13)$$

The accuracy of the predicted P-condition number is affected by the same uncertainties discussed in connection with Eq. I.10.

Appendix II

Test Examples

The specific properties of various test examples used in the numerical investigation of iterative and semi-iterative solution procedures are described in this section. Table A.1 contains a schematical representation of six examples, A1 through A6, employed in the comparative tests of the first part of the dissertation. Three main parameters of the corresponding finite element discretizations, namely the total number of degrees of freedom, n, the number of elements, n_e, as well as the number of nodal points, n_p, are listed in Table A.2. By comparing the structural configurations A4 and A5 it can be seen that both examples represent discretizations of the same structural system. In the case of example A4, however, the symmetric and anti-symmetric properties of the problem are utilized for the purpose of reducing the size of the corresponding stiffness matrix. It can also be observed that examples A5 and A6 involve identical structural discretizations but differ in their loading conditions (cf. Chapter 9). The systems of equations arising from these test examples exhibit all properties of general finite element stiffness matrices described in Appendix 1. The only exception occurs in example A1 whose K-matrix satisfies the condition of diagonal dominance (cf. Section 3.2.3).

The extreme eigenvalues of the coefficient matrices as well as their P-condition numbers (Eq. I.11) are listed in Table A.3. In order to allow a comparison between the P-values and their corresponding upper bounds defined by Eq. I.12, Table A.3 also contains the P_1-condition numbers of the test examples. In the process of describing the convergence behavior and the performance of various iterative methods, a distinction is sometimes made between well- and ill-conditioned problems. In general, such a classification cannot be based on absolute standards, for instance, in the form of limiting P-values. However, it is comparatively simple to define the condition of a problem in relation to that of other structural systems. For the test examples of the first part of the dissertation it is understood that examples A2 and A6 represent well-conditioned problems, whereas the "bending-type" structural systems A3, A4, and A5 are, relatively speaking, ill-conditioned. This behavior is reflected in the magnitude of the condition numbers except in the case of example A6 where certain orthogonality conditions of the load vector have an effect (cf. Chapter 9).

The majority of the numerical tests in the second part of the dissertation are carried out with two examples whose structural discretizations are illustrated in Figs. A.1 and A.2. The numerical characteristics of the corresponding finite element discretizations are listed in Table A.2. Both test examples contain a parameter κ which allows giving the resulting systems of equations any arbitrary degree of ill-conditioning. In the case of example B1 the ill-conditioning is caused by differences in the material properties of two structural subregions. The condition of example B2, however, is governed by the geometry of the over-all structure as well as by the side length ratio of the individual elements.

In Table A.4 the P_1-condition numbers of the global stiffness matrices are listed for various values of κ. In the case of example B1, the logarithm of $P_1(K)$ varies at more or less the same rate as the absolute value of $\log(\kappa)$, whereas the rate is approximately twice as high for test example B2. Table A.4 also contains the condition numbers of the K_s-matrices obtained by applying the scaling transformation of Eq. 7.7 to the original systems of equations. In most cases the transformation has no significant effect on the magnitude of the P_1-values. However, for κ less than 1.0, major differences in the condition numbers can be observed in the case of example B1. For decreasing values of the parameter, the magnitude of $P_1(K)$ varies as an inverse function of κ, whereas $P_1(K_s)$ rapidly approaches an asymptotic value. For this behavior, designated as removable or artificial ill-conditioning, the following physical interpretation can be given. As κ approaches zero, the cantilever problem of example B1 becomes similar to a structural system in which the borderline between the two subregions assumes the character of a fixed support (Fig. A.1). Consequently, the structural configuration approaches that of a less ill-conditioned cantilever problem with smaller span length. The condition of the unscaled coefficient matrix, however, remains directly affected by the ratio of the E-moduli. Various forms of removable ill-conditioning arising in other types of structural systems are discussed, for instance, in Refs. 27, 30, 49, 65, and 66.

In order to investigate the effect of successively finer discretizations on the convergence of the conjugate gradient method, numerical tests were carried out with two additional examples whose basic structural configurations, C_o and D_o, are illustrated in Figs. A.3 and A.4. Table A.2 contains the numerical characteristics of the basic as well as the increasingly finer discretizations obtained by successively halving the mesh spacing h. Convergence problems encountered in the analysis of a structural system similar to that of example C are discussed by Rashid in Ref. 60.

APPENDIX III

Computational Effort

Throughout this investigation the performance of various iterative and semi-iterative solution procedures was measured in the form of $m_{0.1}$-values representing the number of iteration cycles necessary to obtain a certain specified accuracy of the solution vector (Section 2.2). In order to allow a direct comparison of the efficiency, it is necessary to define a suitable index of the required computational effort per iteration cycle. Among various possible forms of such indicators, the total number of matrix-vector products per cycle was selected in this study. It is realized that the quantity represents only an approximate measure since the computational effort for vector-vector and scalar-vector operations is not taken into account. As these operations constitute only a relatively small percentage of the total effort, the measure can, nevertheless, be considered sufficiently accurate for comparison purposes.

In the following paragraphs the number of matrix-vector products is given for various types of solution procedures described in Chapters 3, 4, and 5. The individual values define the minimum computational effort which is required if the algorithms are used in their most efficient formulations. In order to arrive at these minimum values it is frequently necessary to rely on recursive relationships for calculating certain intermediate vector quantities. In practice this approach may not be suitable, however, since the use of recursive relationships increases the effect of roundoff errors on the attainable accuracy of the solution procedures.

The computational effort of linear stationary iterations, expressed in terms of matrix-vector products per iteration cycle, is summarized in Table A.5. It can be observed that Versions A, B, D, and E of a particular algorithm group require an identical number of multiplications, whereas the value is twice as high in the case of Versions C and F. For the single-step iterations of the de la Garza group it is necessary, however, to use recursive relationships in order to arrive at these values (Section 3.2.2).

Within the group of nonlinear stationary iterations, the computational effort per iteration cycle amounts to the following number of matrix-vector multiplications

Steepest Descent, Almost Optimum
Steepest Descent, Krasnoselskii, 1
Gastinel-Householder

Householder, Cauchy, Gastinel 2

Except for Householder's and Gastinel's iterations, the values are based on the assumption that certain vector quantities are computed recursively (Section 3.3.1). The three algorithms included in the linear nonstationary group of iterations (Section 3.4) require only one matrix-vector product per cycle and do not involve recursive calculations.

Since most acceleration procedures are based on vector-vector and scalar-vector operations, their computational effort is generally smaller than that of iterative algorithms. The only exceptions occur among nonlinear stationary accelerations where the following matrix-vector multiplications arise

Wilson	1
Forsythe, Dyer II	2
Other	< 1

By using various recursive relationships, the computational effort for the two conjugate gradient versions can be reduced to a single matrix-vector product per iteration cycle (Chapter 5). The same value applies to s-step gradient methods provided they are carried out in the form of restarted conjugate gradient algorithms.

The computational effort for a matrix-vector product itself depends not only on the size of the system of equations, but also on a large variety of other factors. Among the most important of them are the characteristics of the computing equipment, in particular the size and access time of the storage devices as well as the execution time of arithmetic operations. A similar important role is played by the finite element type which influences the average number of non-zero elements per row or column of the coefficient matrix and determines the number of degrees of freedom per nodal point. The computational effort is also affected by the specific form of algorithm implementation. Major differences may arise depending on whether emphasis is placed on program flexibility or maximum efficiency in a specific case. Except for single-step versions of linear stationary iterations (Section 3.2.1), additional implementation options arise from the fact that the coefficient matrix may be used in assembled or unassembled form. Although it is possible to save a considerable amount of storage by performing the matrix-vector multiplications on the basis of element stiffness matrices alone (Refs. 28, 90), the corresponding computational effort necessarily increases. A similar trade-off between required storage space and required computational effort exists in using recursive relationships for calculating certain intermediate vector quantities. The large variety of influence factors is an indication that a more accurate, yet generally applicable definition of the computational effort per iteration cycle is not feasible.

16. VITA

The author was born on February 13, 1942 in Cottbus, Germany, the son of the late Mr. and Mrs. G. Schultchen.

After completing his high school education, he attended the Colleges of Civil Engineering in Cottbus and Leipzig. From 1966 to 1968 he studied at the Technical University of Stuttgart where he received the Diplom-Ingenieur degree in March 1968. Since July of the same year he has been a research assistant at Fritz Engineering Laboratory, Lehigh University.