Journal of Applied Mathematics and Physics (ZAMP)
Vol. 30, 1979 Birkhäuser Verlag Basel

The Method of Conjugate Gradients in Finite Element Applications

By H. R. Schwarz, Seminar für angewandte Mathematik der Universität Zürich

Dedicated to Professor E. Stiefel

1. Introduction

In 1952 Hestenes and Stiefel [6] developed the method of conjugate gradients for solving symmetric linear systems of high order n with sparse and positive definite coefficient matrices $\mathbf{A}$. The basically iterative procedure was devised in order to take into account the sparsity of the matrix $\mathbf{A}$ to full extent and to operate with the originally given matrix allowing representation of the system of linear equations by means of the operator principle. The operator principle suggested itself, since the large systems of linear equations envisaged had their origin in difference approximations of elliptic and biharmonic boundary value problems. Hence, the systems to be solved could be represented in the computer with a fairly small storage requirement without needing to store the complete coefficient matrix. At that time this was an essential aspect and represented one of the great advantages of the method of conjugate gradients.

Moreover, in [6] the fundamental property of the algorithm was shown, in that the basically iterative procedure yields the solution after at most n steps and hence is theoretically a finite process. Together with the fact, that the method does not require the choice of any parameter to accelerate the convergence, the algorithm of conjugate gradients seemed to be highly suitable for the solution of the large linear systems under consideration.

However, the theoretical property of the algorithm to produce a sequence of mutually orthogonal residual vectors and at the same time a sequence of mutually conjugate relaxation directions is numerically not satisfied so that the algorithm does not need to terminate after n steps with the solution. The numerical deviation from the theory depends on the condition number $\kappa(\mathbf{A})$ of the system matrix $\mathbf{A}$ and gets worse with increasing condition number. The systems of linear equations arising from finite difference approximations are characterized by extremely bad conditions caused by the theoretical fact that the difference equations represent with increasing refinement a finite approximation of an unbounded operator. The condition number increases markedly in case of difference equations of biharmonic boundary value problems [4]. As a consequence, the *cg*-algorithm needed much more than n iteration steps and this practical experience diminished the advantage of the algorithm, and nowadays it is not often used to solve large and sparse symmetric equations.

With the advent of the modern computers with large main storages and offering fast accessible auxiliary storage facilities, the iterative methods lost their significance to some extent, since the direct elimination processes in combination with their proper adjusted application concerning the sparsity of the systems solve the problems in the most efficient way. On the other hand, there exist modern small computers with restricted main storage and with no or slow external storage facilities, for which the advantages of the iterative methods count again.

The aim of the present paper is to show that the method of conjugate gradients is well suited for the solution of the sparse symmetric equations arising from the finite element method for elliptic and biharmonic problems if appropriate measures are taken. Some surprising and even fascinating experimental results will be presented showing that the *cg*-algorithm is capable of yielding the solution with an extremely small number of steps in case of two typical problems. The results reveal that the finite element approximations produce much better conditioned linear systems for which the method of conjugate gradients converges extremely fast.

2. The Basic Algorithm

A system of linear equations

$$\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0} \tag{1}$$

with a symmetric and positive definite matrix $\mathbf{A}$ of order n represents the necessary and sufficient condition for minimizing the quadratic functional

$$F(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x}. \tag{2}$$

The minimum of $F(\mathbf{x})$ is determined iteratively by the method of conjugate gradients of Hestenes and Stiefel [6] by using in each step the gradient of $F(\mathbf{x})$ given by

$$\operatorname{grad} F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{r} \tag{3}$$

and being equal to the residual $\mathbf{r}$ of the vector $\mathbf{x}$. The gradient is used in an ingenious way for determining the direction of relaxation in which the minimum of the functional (2) is searched. Without going into the details the essential points are outlined in the following leading to the well-known basic algorithm. For more details see [7], from where the notation is taken.

Let $\mathbf{v}^{(0)}$ be a starting vector. For decreasing the value $F(\mathbf{v}^{(0)})$ the first relaxation vector $\mathbf{p}^{(1)}$ is set equal to the negative residual vector $\mathbf{r}^{(0)}$.

$$\mathbf{p}^{(1)} = -\mathbf{r}^{(0)} = -\operatorname{grad} F(\mathbf{v}^{(0)}) = -(\mathbf{A}\mathbf{v}^{(0)} + \mathbf{b}). \tag{4}$$

The minimum of $F(\mathbf{v})$ for the candidates

$$\mathbf{v}^{(1)} = \mathbf{v}^{(0)} + q_1\mathbf{p}^{(1)} \tag{5}$$

is reached for the value of the parameter q_1 given by

$$q_1 = -\frac{\mathbf{p}^{(1)T}\mathbf{r}^{(0)}}{\mathbf{p}^{(1)T}\mathbf{A}\mathbf{p}^{(1)}} = \frac{\mathbf{r}^{(0)T}\mathbf{r}^{(0)}}{\mathbf{p}^{(1)T}\mathbf{A}\mathbf{p}^{(1)}}. \tag{6}$$

In the general kth relaxation step the relaxation vector $\mathbf{p}^{(k)}$ is chosen as a linear combination of $-\mathbf{r}^{(k-1)}$ and the previous relaxation vector $\mathbf{p}^{(k-1)}$ as

$$\mathbf{p}^{(k)} = -\mathbf{r}^{(k-1)} + e_{k-1}\mathbf{p}^{(k-1)} \tag{7}$$

in such a way that the two successive relaxation vectors are conjugate or A-orthogonal, i.e. they satisfy

$$\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k-1)} = 0. \tag{8}$$

From (8) the value for e_{k-1} in (7) follows to be

$$e_{k-1} = \frac{\mathbf{r}^{(k-1)T}\mathbf{A}\mathbf{p}^{(k-1)}}{\mathbf{p}^{(k-1)T}\mathbf{A}\mathbf{p}^{(k-1)}}, \quad (k \geq 2). \tag{9}$$

The approximation $\mathbf{v}^{(k)}$ is determined in the direction of $\mathbf{p}^{(k)}$ in analogy to the first step

$$\mathbf{v}^{(k)} = \mathbf{v}^{(k-1)} + q_k\mathbf{p}^{(k)}. \tag{10}$$

The scalar q_k follows from the minimizing condition for $F(\mathbf{v}^{(k)})$ to be

$$q_k = -\frac{\mathbf{p}^{(k)T}\mathbf{r}^{(k-1)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}}, \quad (k \geq 2). \tag{11}$$

The formulas (9) and (11) for e_{k-1} and q_k can be modified into simpler expressions taking into account that the residual vector $\mathbf{r}^{(k)}$ is orthogonal to the two-dimensional subspace defined by $\mathbf{r}^{(k-1)}$ and $\mathbf{p}^{(k)}$. We get

$$e_{k-1} = \frac{\mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}}{\mathbf{r}^{(k-2)T}\mathbf{r}^{(k-2)}}, \qquad q_k = \frac{\mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}}, \quad (k \geq 2). \tag{12}$$

The representation for q_k is also valid in case of $k = 1$ due to (6). Finally, the residual vector $\mathbf{r}^{(k)}$ can be computed recursively on the base of the relation

$$\mathbf{r}^{(k)} = \mathbf{A}\mathbf{v}^{(k)} + \mathbf{b} = \mathbf{A}(\mathbf{v}^{(k-1)} + q_k\mathbf{p}^{(k)}) + \mathbf{b} = \mathbf{r}^{(k-1)} + q_k(\mathbf{A}\mathbf{p}^{(k)}).$$

The method of conjugate gradients is summarized in the following algorithm.

```
Start: Choice of $\mathbf{v}^{(0)}$;
$\mathbf{r}^{(0)} = \mathbf{A}\mathbf{v}^{(0)} + \mathbf{b}$; $\mathbf{p}^{(1)} = -\mathbf{r}^{(0)}$.                         (13)
General relaxation step ($k = 1, 2, \ldots$):
$e_{k-1} = \mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}/\mathbf{r}^{(k-2)T}\mathbf{r}^{(k-2)}$  \  if $k \geq 2$      (14)
$\mathbf{p}^{(k)} = -\mathbf{r}^{(k-1)} + e_{k-1}\mathbf{p}^{(k-1)}$                /                (15)
  $q_k = \mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}/\mathbf{p}^{(k)T}(\mathbf{A}\mathbf{p}^{(k)})$                      (16)
$\mathbf{v}^{(k)} = \mathbf{v}^{(k-1)} + q_k\mathbf{p}^{(k)}$                                          (17)
$\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} + q_k(\mathbf{A}\mathbf{p}^{(k)})$.                                    (18)
```

A general relaxation step requires the multiplication of the matrix **A** with the vector $\mathbf{p}^{(k)}$. For a sparse matrix **A** the computational effort for this operation is directly proportional to the number N of nonzero matrix elements. For large sparse matrices **A** the number $N = \gamma n$ is only proportional to the order n of **A**, where γ denotes the average number of nonzero elements in each row of **A**. The two inner products and the three multiplications of vectors of dimension n by scalars together with the addition of vectors require $5n$ additional multiplications and additions. The total amount of computational work per iteration step consists of about $Z_{cg} = (\gamma + 5)n$ multiplications and additions.

3. Improving the Condition and the Convergence Property

A bad condition of the matrix **A** causes that the sequence of residual vectors are not mutually orthogonal such that the iterative process has to be continued far beyond the theoretically required n steps [4]. Hence the convergence properties of the *cg*-algorithm depend highly on the condition number of **A** and can be influenced by reducing the condition number by proper measures.

The simplest attempt to reduce the spectral condition number [4, 7] consists in scaling the given matrix **A** into an equilibrated or at least almost equilibrated matrix $\hat{\mathbf{A}}$ [5, 9]. In order to preserve the symmetry of the given matrix **A** only simultaneous row and column scaling operations may be considered, expressed by the matrix equation

$$\hat{\mathbf{A}} = \mathbf{DAD}, \tag{19}$$

where **D** denotes a diagonal matrix with positive diagonal elements d_i. The determination of the d_i such that $\hat{\mathbf{A}}$ is equilibrated from the set of nonlinear equations

$$d_i\left[\sum_{j=1}^{n} (a_{ij}d_j)^2\right]^{1/2} = 1, \quad (i = 1, 2, \ldots, n)$$

requires a large computational effort. For practical purposes in connection with large sparse matrices **A** we shall restrict the scaling to the simple choice of scaling factors

$$d_i = 1/\sqrt{a_{ii}}, \quad (i = 1, 2, \ldots, n). \tag{20}$$

With these scaling factors the diagonal elements of $\hat{\mathbf{A}}$ get the value one, and due to necessary condition of positive definiteness of $\hat{\mathbf{A}}$ the off-diagonal elements are bounded by $\hat{a}_{ij}^2 < 1$, and hence the row and column norms of the scaled matrix $\hat{\mathbf{A}}$ satisfy the inequalities

$$1 \le \left[\sum_{j=1}^{n} \hat{a}_{ij}^2\right]^{1/2} \le \sqrt{\gamma_i}, \quad (i = 1, 2, \ldots, n). \tag{21}$$

In (21) γ_i denotes the number of nonzero elements of the ith row.

Although the scaling (20) yields only almost equilibrated matrices, the improvement of the condition number can be quite essential in certain finite element applications. If the nodal values are all function values as e.g. in case of linear and quadratic triangles or bilinear and quadratic parallelograms [8, 10], the simple scaling (20) does not reduce the condition number essentially. The same remark holds for finite difference approximations. However, if the nodal values comprise function values as well as partial derivatives, the situation is quite different. Such cubic triangles and parallelograms are used for elliptic boundary value problems as well as for plate bending problems [8, 10]. In these applications of the cubic elements the diagonal entries of the matrix $\mathbf{A}$, i.e. the stiffness matrix of the problem, are multiplied by different powers of a quantity which characterizes the size of the element. As a consequence, the diagonal elements differ in size more and more in refining the mesh. Since the spectral condition number of a symmetric and positive definite matrix is greater or equal to the quotient of the largest and the smallest diagonal element [7], the condition number of the (unscaled) matrix increases unduly fast. The scaling (20) provides in these cases an excellent and simple remedy (see examples below).

In generalization of the scaling process by means of a diagonal matrix, a reduction of the condition number can be achieved by an equivalence transformation of $\mathbf{A}$ with an appropriate nondiagonal regular matrix $\mathbf{H}$. Let $\mathbf{C}$ be the symmetric and positive definite matrix defined by

$$\mathbf{C} = \mathbf{H}\mathbf{H}^T. \tag{22}$$

The given system of linear equations (1) is brought into the equivalent form by inserting $\mathbf{I} = \mathbf{H}^{-T}\mathbf{H}^T$ and multiplying from the left by $\mathbf{H}^{-1}$ ($\mathbf{H}^{-T}$ denotes the transpose of the inverse).

$$\mathbf{H}^{-1}\mathbf{A}\mathbf{H}^{-T}\mathbf{H}^T\mathbf{x} + \mathbf{H}^{-1}\mathbf{b} = \mathbf{0}. \tag{23}$$

With the auxiliary quantities

$$\tilde{\mathbf{A}} = \mathbf{H}^{-1}\mathbf{A}\mathbf{H}^{-T}, \qquad \tilde{\mathbf{x}} = \mathbf{H}^T\mathbf{x}, \qquad \tilde{\mathbf{b}} = \mathbf{H}^{-1}\mathbf{b} \tag{24}$$

the transformed system of equations (23) reads as

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} + \tilde{\mathbf{b}} = \mathbf{0}. \tag{25}$$

The matrix $\tilde{\mathbf{A}}$ is still symmetric and positive definite. The transformation matrix $\mathbf{H}$, respectively the matrix $\mathbf{C}$ should be chosen such that the condition number $\kappa(\tilde{\mathbf{A}})$ is essentially smaller than $\kappa(\mathbf{A})$. An indication for an appropriate choice of $\mathbf{C}$ is given by the observation, that the matrix $\tilde{\mathbf{A}}$ is similar to

$$\mathbf{H}^{-T}\tilde{\mathbf{A}}\mathbf{H}^T = \mathbf{H}^{-T}\mathbf{H}^{-1}\mathbf{A}\mathbf{H}^{-T}\mathbf{H}^T = \mathbf{C}^{-1}\mathbf{A}. \tag{26}$$

With the optimal choice $\mathbf{C} = \mathbf{A}$ the matrix $\tilde{\mathbf{A}}$ would be similar to the identity $\mathbf{I}$ with a condition number $\kappa(\tilde{\mathbf{A}}) = 1$. From practical reasons this choice is not meaningful as we shall see, but it follows from (26) that $\mathbf{C}$ should be an approximation to $\mathbf{A}$ in some sense. A proper choice for $\mathbf{C}$ will be discussed below.

The method of conjugate gradients is now formally applied to the so-called preconditioned system of Eqns. (25). With a starting vector $\tilde{\mathbf{v}}^{(0)}$ we get from (13) through (18) the following set of defining equations.

$$\tilde{\mathbf{r}}^{(0)} = \tilde{\mathbf{A}}\tilde{\mathbf{v}}^{(0)} + \tilde{\mathbf{b}}, \qquad \tilde{\mathbf{p}}^{(1)} = -\tilde{\mathbf{r}}^{(1)} \tag{13'}$$

$$\left.\begin{aligned} \tilde{e}_{k-1} &= \tilde{\mathbf{r}}^{(k-1)T}\tilde{\mathbf{r}}^{(k-1)}/\tilde{\mathbf{r}}^{(k-2)T}\tilde{\mathbf{r}}^{(k-2)} && (14') \\ \tilde{\mathbf{p}}^{(k)} &= -\tilde{\mathbf{r}}^{(k-1)} + \tilde{e}_{k-1}\tilde{\mathbf{p}}^{(k-1)} && (15') \end{aligned}\right\} \text{for } k \geq 2$$

$$\tilde{q}_k = \tilde{\mathbf{r}}^{(k-1)T}\tilde{\mathbf{r}}^{(k-1)}/\tilde{\mathbf{p}}^{(k)T}(\tilde{\mathbf{A}}\tilde{\mathbf{p}}^{(k)}) \tag{16'}$$

$$\tilde{\mathbf{v}}^{(k)} = \tilde{\mathbf{v}}^{(k-1)} + \tilde{q}_k\tilde{\mathbf{p}}^{(k)} \tag{17'}$$

$$\tilde{\mathbf{r}}^{(k)} = \tilde{\mathbf{r}}^{(k-1)} + \tilde{q}_k(\tilde{\mathbf{A}}\tilde{\mathbf{p}}^{(k)}). \tag{18'}$$

However, the algorithm is not performed on the basis of the actually transformed matrix $\tilde{\mathbf{A}}$, which would be in general full but with respect to the originally given matrix $\mathbf{A}$, i.e. the preconditioning is done implicitly. The formulas (13′) through (18′) are restated with the quantities referring to the original system (1). According to (23), (24), (13′) and (15′) the relations hold

$$\tilde{\mathbf{r}}^{(k)} = \mathbf{H}^{-1}\mathbf{r}^{(k)}, \qquad \tilde{\mathbf{v}}^{(k)} = \mathbf{H}^T\mathbf{v}^{(k)}, \qquad \tilde{\mathbf{p}}^{(k)} = \mathbf{H}^{-1}\mathbf{p}^{(k)}. \tag{27}$$

Equation (17′) yields after multiplication with $\mathbf{H}^{-T}$ from the left

$$\mathbf{v}^{(k)} = \mathbf{v}^{(k-1)} + \tilde{q}_k(\mathbf{C}^{-1}\mathbf{p}^{(k)}). \tag{28}$$

In analogy, Eqn. (18′) reads after multiplication with $\mathbf{H}$ from the left according to (24)

$$\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} + \tilde{q}_k\mathbf{A}(\mathbf{C}^{-1}\mathbf{p}^{(k)}). \tag{29}$$

From (15′) we get after multiplication by $\mathbf{H}^{-T}$ from the left

$$\mathbf{C}^{-1}\mathbf{p}^{(k)} = -\mathbf{C}^{-1}\mathbf{r}^{(k-1)} + \tilde{e}_{k-1}(\mathbf{C}^{-1}\mathbf{p}^{(k-1)}). \tag{30}$$

This represents a recursion formula for the auxiliary vector

$$\mathbf{g}^{(k)} = \mathbf{C}^{-1}\mathbf{p}^{(k)}. \tag{31}$$

At the same time we introduce the second auxiliary vector

$$\boldsymbol{\rho}^{(k)} = \mathbf{C}^{-1}\mathbf{r}^{(k)}. \tag{32}$$

With (32) the inner products in (14′) and (16′) can be written as follows taking into account the symmetry of $\mathbf{C}^{-1}$

$$\tilde{\mathbf{r}}^{(k)T}\tilde{\mathbf{r}}^{(k)} = \mathbf{r}^{(k)T}\mathbf{H}^{-T}\mathbf{H}^{-1}\mathbf{r}^{(k)} = \mathbf{r}^{(k)T}\boldsymbol{\rho}^{(k)}, \tag{33}$$

$$\tilde{\mathbf{p}}^{(k)T}\tilde{\mathbf{A}}\tilde{\mathbf{p}}^{(k)} = \mathbf{p}^{(k)T}\mathbf{H}^{-T}\mathbf{H}^{-1}\mathbf{A}\mathbf{H}^{-T}\mathbf{H}^{-1}\mathbf{p}^{(k)} = \mathbf{g}^{(k)T}\mathbf{A}\mathbf{g}^{(k)}. \tag{34}$$

In the preconditioned algorithm of conjugate gradients the original relaxation vectors $\mathbf{p}^{(k)}$ are replaced by the vectors $\mathbf{g}^{(k)}$. In order to perform the preconditioning the vectors $\boldsymbol{\rho}^{(k)}$ defined in (32) have to be used in addition to the original residual vectors $\mathbf{r}^{(k)}$. The auxiliary vector $\boldsymbol{\rho}^{(k)}$ is determined from the system of linear equations

$$\mathbf{C}\boldsymbol{\rho}^{(k)} = \mathbf{r}^{(k)}. \tag{35}$$

The preconditioned method of conjugate gradients is summarized in the following algorithm.

```
Start: Choice of $\mathbf{C} = \mathbf{H}\mathbf{H}^T$ and $\mathbf{v}^{(0)}$;
$\mathbf{r}^{(0)} = \mathbf{A}\mathbf{v}^{(0)} + \mathbf{b}$;   $\mathbf{C}\boldsymbol{\rho}^{(0)} = \mathbf{r}^{(0)}$;   $\mathbf{g}^{(1)} = -\boldsymbol{\rho}^{(0)}$.        (36)
General relaxation step ($k = 1, 2, \ldots$):
    $\tilde{e}_{k-1} = \mathbf{r}^{(k-1)T}\boldsymbol{\rho}^{(k-1)}/\mathbf{r}^{(k-2)T}\boldsymbol{\rho}^{(k-2)}$   } for $k \geq 2$        (37)
    $\mathbf{g}^{(k)} = -\boldsymbol{\rho}^{(k-1)} + \tilde{e}_{k-1}\mathbf{g}^{(k-1)}$                  }                  (38)
    $\tilde{q}_k = \mathbf{r}^{(k-1)T}\boldsymbol{\rho}^{(k-1)}/[\mathbf{g}^{(k)T}(\mathbf{A}\mathbf{g}^{(k)})]$        (39)
    $\mathbf{v}^{(k)} = \mathbf{v}^{(k-1)} + \tilde{q}_k\mathbf{g}^{(k)}$        (40)
    $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} + \tilde{q}_k(\mathbf{A}\mathbf{g}^{(k)})$        (41)
$\mathbf{C}\boldsymbol{\rho}^{(k)} = \mathbf{r}^{(k)}$.        (42)
```

Each step of the preconditioned *cg*-algorithm requires the solution of the system (42). The computational effort for solving (42) should not be too high. This requirement restricts the choice of the matrix $\mathbf{C}$, or equivalently of the matrix $\mathbf{H}$ to sparse lower triangular matrices. According to an idea of Axelsson [1, 2, 3] the matrix $\mathbf{C}$ is directly derived from the given matrix $\mathbf{A}$, which is decomposed into the sum of a strict lower triangular matrix $\mathbf{E}$, a diagonal matrix $\mathbf{D}$ with the positive diagonal element of $\mathbf{A}$ and a strict upper triangular matrix $\mathbf{F} = \mathbf{E}^T$ as follows

$$\mathbf{A} = \mathbf{E} + \mathbf{D} + \mathbf{F}, \tag{43}$$

$$\mathbf{C} = (\mathbf{D} + \omega\mathbf{E})\mathbf{D}^{-1}(\mathbf{D} + \omega\mathbf{F}) = (\mathbf{D}^{1/2} + \omega\mathbf{E}\mathbf{D}^{-1/2})(\mathbf{D}^{1/2} + \omega\mathbf{D}^{-1/2}\mathbf{F}) = \mathbf{H}\mathbf{H}^T. \tag{44}$$

In the definition (44) of the lower triangular matrix $\mathbf{H}$ a relaxation parameter ω appears, which has to be chosen properly. The matrix $\mathbf{H}$ has the identical sparsity distribution as the lower part of the given matrix $\mathbf{A}$. The definition of the matrix $\mathbf{H}$ has the important advantage that no additional storage is required. Due to the definition (44) of $\mathbf{C}$ the solution of the system $\mathbf{C}\boldsymbol{\rho} = \mathbf{r}$ is performed in the two essential steps

$$(\mathbf{D} + \omega\mathbf{F})\mathbf{y} = \mathbf{r}, \qquad (\mathbf{D} + \omega\mathbf{E})\boldsymbol{\rho} = \mathbf{D}\mathbf{y}, \tag{45}$$

corresponding to the forward and backward substitution. Since the sparsity of the triangular matrices can be exploited to full extent, the computational effort involved in (45) corresponds exactly to the multiplication of $\mathbf{A}$ with a vector. The total computational effort of one step of the preconditioned *cg*-algorithm is essentially doubled in comparison with the usual *cg*-algorithm. Hence the preconditioning reduces the total amount of arithmetic only, if the number of steps is at least halved by a proper choice of ω. For larger problems this is indeed the case, thus justifying the double arithmetic effort per step.

The conditioning matrix $\mathbf{C}$ (44) represents indeed for $\omega \neq 0$ an approximation of $\mathbf{A}$ in some sense, since we have

$$\mathbf{C} = \mathbf{D} + \omega\mathbf{E} + \omega\mathbf{F} + \omega^2\mathbf{E}\mathbf{D}^{-1}\mathbf{F} = \omega[\mathbf{A} + (\omega^{-1} - 1)\mathbf{D} + \omega\mathbf{E}\mathbf{D}^{-1}\mathbf{F}]. \tag{46}$$

For $\omega = 0$ the matrix $\mathbf{C}$ reduces to $\mathbf{D}$. Hence the preconditioning simplifies with $\mathbf{H} = \mathbf{D}^{1/2}$ to the scaling of $\mathbf{A}$ into $\hat{\mathbf{A}}$ according to (20), thus relating the two methods together.

4. Examples

The effect of scaling and preconditioning on the convergence of the *cg*-algorithm is illustrated in two typical applications of the finite element method. The two examples show the surprising and experimental fact, that the *cg*-algorithm may solve the large and sparse linear equations with an extremely small number of steps. This result is certainly in contrast with the well-known slow convergence observed in solving the linear equations arising in finite difference approximation.

Example 1: We consider the elliptic boundary value problem of determining the stationary temperature distribution in the interior of the region G of Figure 1, under the assumption that in the region G a continuous heat source exists, that on the boundary AB the temperature is held constant equal to zero, that the rest of the rectilinear boundary is isolated, whereas on the half circle there is convection of heat. The elliptic boundary problem to be solved is

$$\Delta u = -20 \quad \text{in } G \tag{47}$$

$$u = 0 \quad \text{on } AB \tag{48}$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } BC, CD, DE, HI, IA \tag{49}$$

$$\frac{\partial u}{\partial n} + 2u = 0 \quad \text{on } EFH. \tag{50}$$

Figure 1 contains the dimensions in dimensionless units and shows at the same time the approximation of the region by finite elements. The curved boundary is approximated by rectilinear segments, the endpoints of which are equally distributed on the half circle. The approximating region is hence slightly larger than the given region.

The problem is treated firstly with the combination of quadratic triangles (six

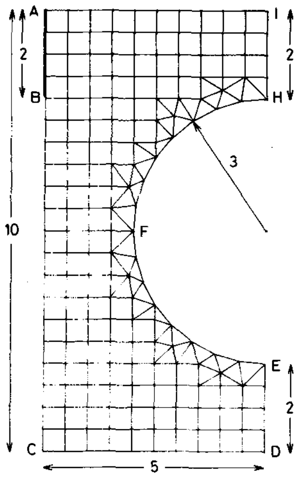

Figure 1
Region G with subdivision in finite elements.

nodes per triangle) and quadratic rectangular elements of the Serendipity class (eight nodes per rectangle) [8]. The number of nodal values is $n = 537$, nine of which are given by the boundary condition (48). Using the same subdivision into finite elements, the problem is treated secondly with cubic elements. Cubic triangular elements of Zienkiewicz [8, 10] (nine nodal values per triangle) are combined with cubic rectangular elements of the Serendipity class [8] (twelve nodal values per rectangle). In each corner of the elements the function value u together with the two first partial derivatives u_x and u_y act as nodal variables. With 181 nodes of the discretization the number of nodal values is $n = 543$, ten of which are given by the boundary condition (48). The order of the two resulting systems of equations is comparable.

Table 1 contains the number of iteration steps for the *cg*-algorithm if it is applied directly to the resulting systems and to the scaled linear systems and finally the required steps for the preconditioned *cg*-algorithm when using the optimal value of ω. The termination criterion used was

$$\mathbf{r}^{(k)T}\mathbf{r}^{(k)} \leq 10^{-20}\mathbf{r}^{(0)T}\mathbf{r}^{(0)}. \tag{51}$$

Scaling the matrix $\mathbf{A}$ in case of the quadratic elements does not reduce the number of steps in a substantial way, whereas the preconditioning gives a marked reduction

Table 1
Elliptic Boundary Value Problem

Type of Elements	n	*cg*-Algorithm		Preconditioned *cg*	
		Unscaled	Scaled		ω_{opt}
Quadratic	537	180	170	41	1.4–1.5
Cubic	543	223	69	27	1.1–1.3

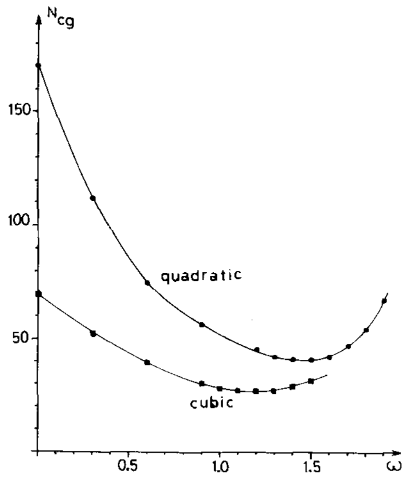

Figure 2
Iteration steps of the preconditioned *cg*-algorithm, elliptic problem.

of the steps. In case of the cubic elements the scaling operation yields already a substantial saving in iteration steps, whereas the preconditioning does not reduce the total effort essentially, since the 27 steps are equivalent to 54 steps of the unconditioned *cg*-algorithm.

Figure 2 shows the graph of the required steps of the preconditioned *cg*-algorithm in function of the parameter ω. The number of steps is not very sensitive on the choice of ω near the optimal value. A rough value gives satisfactory results.

There is a somehow surprising result to be pointed out: In case of quadratic elements, the optimal value of ω increases with the mesh refinement. For cubic elements, however, the optimal value increases quite slowly and rests in the neighbourhood of one!

Example 2: A square-shaped plate of sidelength $L = 2$ is considered, clamped on the left-hand side, supported on the right-hand side and free on the two horizontal boundaries (see Figure 3). The plate is loaded in the hatched square region of Figure 3

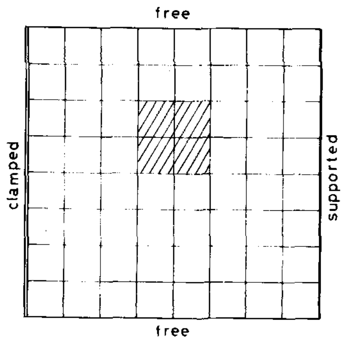

Figure 3
Test plate.

Table 2
Biharmonic Boundary Value Problem

Case	Type of Element	n	*cg*-Algorithm		Preconditioned *cg*	
			Unscaled	Scaled		ω_{opt}
I	Conforming	324	>972(!)	128	58	1.00–1.10
II	Nonconforming	243	294	121	55	1.00–1.10
III	Zienkiewicz	435	631	239	100	1.10–1.15

by a constant load $p = 1$. Poisson's number is taken to be $\nu = 1/6$ and the plate constant $D = Ed^3/[12(1 - \nu^2)]$ is set equal to unity, where d denotes the constant thickness of the plate. Figure 3 shows the discretization of the region into 64 square elements. This test plate has been considered in [4].

Using the variational approach [8], the plate bending problem is treated with three different finite elements. The first type is the conforming cubic rectangular element with 16 nodal variables, corresponding to a bicubic displacement model [8]. In each of the four corners the values w, w_x, w_y and w_{xy} act as nodal variables. With 81 nodal points the finite element model has $n = 324$ nodal variables, 54 of which are prescribed by the geometric boundary conditions. The second type is the non-conforming cubic rectangular element with 12 nodal variables, corresponding to the cubic displacement model of the Serendipity class [8]. The values w, w_x, w_y act as nodal variables in each node. With the 81 nodal points the finite element model is described by $n = 243$ nodal variables, 45 of which are given by the boundary conditions. The third type is the non-conforming cubic triangular element of Zienkiewicz with 9 nodal variables [8, 10]. In order to apply this element, each square of the discretization shown in Figure 2 is subdivided into four subtriangles with the center of gravity as additional nodal point. The total number of nodes is increased to 145,

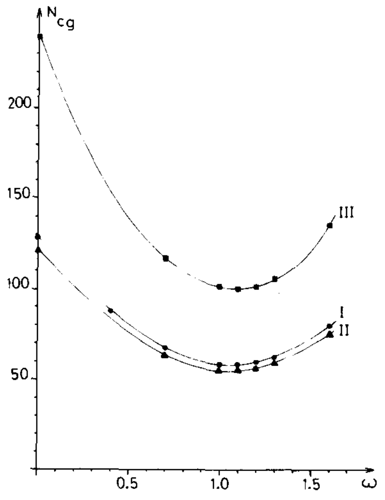

Figure 4
Iteration steps of the preconditioned *cg*-algorithm, biharmonic problem.

Table 3
Testplate with Uniformly Distributed Force

Case	Type of Element	n	cg-Algorithm		Preconditioned	
			Unscaled	Scaled		ω_{opt}
I	Conforming	324	>972(!)	84	57	1.05–1.10
II	Nonconforming	243	217	77	55	1.1 –1.2
III	Zienkiewicz	435	477	148	100	1.0 –1.2

leading to $n = 435$ nodal variables, 45 of which are again prescribed by the boundary conditions.

Table 2 contains the experimental results concerning the required iteration steps of the cg-algorithm as well as of the preconditioned cg-algorithm with the optimal value for ω. The stopping criterion (51) is used.

If the unscaled matrix is used for the cg-algorithm, the number of iteration steps exceeds the order n of the system of equations. The scaling operation reduces the computational effort, whereas the preconditioning does no more reduce the amount of work essentially, since the number of iteration steps is approximately halved. The observation that ω_{opt} for the preconditioning lies in the region of 1.10 is astonishing. Figure 4 illustrates the iteration steps in the three cases of the preconditioned cg-algorithm in function of the parameter ω revealing again the flat minimum.

The situation seems to be slightly different if the plate is loaded by a uniformly distributed force over the whole region. The corresponding results are summarized in Table 3. The preconditioning does reduce the number of iteration steps in comparison with the scaling, but the total effort is even increased.

References

[1] O. Axelsson, *A Generalized SSOR Method*, BIT *13*, 443–457 (1972).
[2] O. Axelsson, *A Class of Iterative Methods for Finite Element Equations*, Comp. Meth. Appl. Mech. Engin. *9*, 123–137 (1976).
[3] O. Axelsson, *Solution of Linear Systems of Equations: Iterative Methods*, in, Lect. Not. Maths, No. 572, Springer, Berlin-Heidelberg-New York (1977).
[4] M. Engeli, Th. Ginsburg, H. Rutishauser, and E. Stiefel, *Refined Iterative Methods for the Computation of the Solution and the Eigenvalues of Selfadjoint Boundary Value Problems*, Basel-Stuttgart 1959, Mitt. Inst. f. angew. Math. ETH Zürich, Nr. 8.
[5] G. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs (1967).
[6] M. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, J. Res. Nat. Bur. Standards *49*, 409–436 (1952).
[7] H. R. Schwarz, H. Rutishauser, and E. Stiefel, *Numerik symmetrischer Matrizen*, 2. Aufl., Teubner, Stuttgart (1972).
[8] H. R. Schwarz, *Methode der finiten Elemente*, Teubner, Stuttgart (1979).
[9] H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford (1965).
[10] O. C. Zienkiewicz, *Methode der finiten Elemente*, VEB, Leipzig (1974).

Summary

The method of conjugate gradients is applied to solve the linear equations arising from the finite element approximations for an elliptic and a biharmonic boundary value problem. It is shown experimentally that a proper scaling of the matrix may reduce the required number of iteration steps substantially. The preconditioned *cg*-algorithm shows even faster convergence.

Zusammenfassung

Die Methode der konjugierten Gradienten wird zur Lösung der linearen Gleichungssysteme angewandt, wie sie aus der Methode der finiten Elemente zur Approximation eines elliptischen und biharmonischen Randwertproblems resultieren. Es wird das experimentelle Ergebnis gezeigt, dass die Zahl der erforderlichen Iterationsschritte gelegentlich allein durch eine geeignete Skalierung der Matrix wesentlich reduziert werden kann. Der vorkonditionierte *cg*-Algorithmus weist noch bessere Konvergenz auf.

(Received: October 2, 1978)