BIT 25 (1985), 166–187

INVITED PAPER.

A SURVEY OF PRECONDITIONED ITERATIVE METHODS FOR LINEAR SYSTEMS OF ALGEBRAIC EQUATIONS

O. AXELSSON

Mathematical Institute, University of Nijmegen, Toernooiveld, 6525 ED Nijmegen, The Netherlands *)

Dedicated to Carl-Erik Fröberg, a pioneer in Numerical Methods.

Abstract.

We survey preconditioned iterative methods with the emphasis on solving large sparse systems such as arise by discretization of boundary value problems for partial differential equations.

We discuss shortly various acceleration methods but the main emphasis is on efficient preconditioning techniques. Numerical simulations on practical problems have indicated that an efficient preconditioner is the most important part of an iterative algorithm. We report in particular on the state of the art of preconditioning methods for vectorizable and/or parallel computers.

1. Introduction.

We shall consider the numerical solution of very large but sparse linear algebraic systems

$$Ax = b, \qquad x, b \in R^N. \tag{1.1}$$

Although the methods we shall present do not always require this, for ease of presentation we shall assume that A is a nonsingular diagonally dominant M-matrix (i.e., $a_{ij} \leqq 0$, $i \neq j$ and the entries of the inverse, A^{-1} are positive).

Such matrices may, for instance, arise when we apply a difference or the lowest order finite element method for a diffusion equation. They also arise at each correction step of a defect-correction method for convection-diffusion problems, where the correction operator is derived from an artificial diffusion or upwind difference (or finite element) operator.

The order N may be very large. For example, for a scalar equation in three space dimensions (3D) on a $64 \times 64 \times 64$ mesh we get $N \approx 250000$. However, A has typically a band structure or skyline structure and in each row (and column) of the matrix, only a few, typically 5 or 7 entries are nonzeros.

Direct solution methods based on a factorization, $A = \hat{L}\hat{U}$ of A into lower

Received October 1984. Revised December 1984.

*) Present address: Department of Computer Science and Numerical Mathematics, Lund University, P.O. Box 118, S-22100 Lund, Sweden.

and upper triangular factors, $\hat{L}$ and $\hat{U}$, respectively, produce, however, fill-in within most of the band or skyline and are hence costly. For large problems the matrix must be stored on peripheral storage and the I/O (i.e., disk file read/write) costs tend to dominate.

Another, sometimes severe, drawback of direct solution methods is that round-off errors and errors in given data tend to increase faster than the condition numbers; for elliptic difference equations they increase as $O(h^{-3})$, where h is a stepsize parameter (see, for instance, [1]). This means that multiple precision arithmetic computations are mostly needed which adds even more to the overall costs (storage and hence I/O, and arithmetics). However, as is well known, by use of iterative refinement, one can partly avoid multiple precision arithmetic.

Iterative solution methods on the other hand do not suffer from fill-in and with effective preconditioned and accelerated methods one may derive algorithms of almost optimal order of computational complexity. By a carefui evaluation of the residuals, the influence of round-off errors may be decreased by several orders of magnitude as compared to direct methods (see [1]).

Let $C = LU$ be an approximation, for instance, an incomplete factorization of A. The solution of linear systems with C must be relatively inexpensive. C is often called a *preconditioning* matrix.

A basic iterative method has the form

$$C\boldsymbol{\delta}^{l+1} = -\boldsymbol{r}^l, \qquad \boldsymbol{x}^{l+1} = \boldsymbol{x}^l + \boldsymbol{\delta}^{l+1}, \qquad l = 0, 1, 2, \ldots, \tag{1.2}$$

of a *defect-correction method* where $\boldsymbol{r}^l = A\boldsymbol{x}^l - \boldsymbol{b}$ is the defect or residual and $\boldsymbol{\delta}^{l+1}$ is the correction at stage l. Further, $\boldsymbol{x}^0$ is arbitrary, but a good choice is $\boldsymbol{x}^0 = C^{-1}\boldsymbol{b}$.

Consider the splitting

$$A = C - R \tag{1.3}$$

of A, where R is the defect matrix. Then (1.2) takes the form

$$C\boldsymbol{x}^{l+1} = R\boldsymbol{x}^l + \boldsymbol{b}, \qquad l = 0, 1, 2, \ldots.$$

which converges for all choices of x^0 if and only if $\varrho(C^{-1}R) < 1$, where $\varrho(\cdot)$ is the spectral radius. The theory of (weak) regular splittings may be applied (see [2]). The rate of convergence as measured in the number of iterations to reach a relative error, $\|\boldsymbol{x} - \boldsymbol{x}^k\|/\|\boldsymbol{x} - \boldsymbol{x}^0\| \leqq \varepsilon$ is

$$k \cong ln(1/\varepsilon)/ln(1/\varrho_0), \tag{1.4}$$

where $\varrho_0 = \|C^{-1}R\|$. For second order elliptic problems and if $C = D_A$, the (block) diagonal part of A, one gets $\varrho_0 = 1/(1+\zeta h^2)$, for some positive ζ,

independent of h. Hence $k = O(h^{-2})$ which is unacceptable, because this would involve too much computational work.

The efficiency of the simplest iterative methods may be improved in two ways:

(i) by use of some accelerated form of iterative methods,

(ii) by a proper choice of the preconditioning matrix C.

In section 2 we discuss shortly some iterative acceleration methods of polynomial acceleration type.

In section 3 we survey some preconditioning methods of classical type. More recently developed preconditioners of block-matrix incomplete factorization type are discussed in Section 4.

In section 5 we discuss modifications needed to utilize preconditioners efficiently on vector and parallel computers, a topic of much current interest. We comment on preconditioners of matrix polynomial type and show a negative result for them. We then also comment on the gain in efficiency that sometimes can be achieved by proper reorderings of the equations or by utilizing special properties of the matrices.

2. Accelerated iterative methods.

Consider first the iterative method

$$x^{l+1} = x^l - \tau_l C^{-1} r^l, \qquad r^l = Ax^l - b, \qquad l = 0, 1, \ldots \tag{2.1}$$

for the solution of (1.1). Here $\{\tau_l\}$ is a sequence of iteration (acceleration) parameters. If $\tau_l = \tau$, $l = 0, 1, \ldots$, we talk about a stationary iterative method, otherwise about a nonstationary or semi-iterative method.

Let $e^l = x - \mathrm{x}^l$, the iteration error. Then it follows from (2.1) that $e^{l+1} = (I - \tau_l C^{-1} A) e^l$, $l = 0, 1, \ldots$, so $e^m = P_m(C^{-1}A)e^0$ (and $r^m = AP_m(C^{-1}A)A^{-1}r^0 = P_m(AC^{-1})r^0$). Here $P_m(\lambda) = \prod_{l=0}^{m}(1 - \tau_l \lambda)$ is an mth order polynomial having zeros at $1/\tau_l$ and satisfying $P_m(0) = 1$.

We want to choose the parameters $\{\tau_l\}$ such that $\|e^m\|$ is minimized. However, this would mean that in general the parameters would depend on e^0, which is actually not known. We also assume that the eigenvalues of $C^{-1}A$ are not known. We then take the philosophy of minimizing $\|e^m\|/\|e^0\|$ for all e^0, i.e. we want to minimize $\|P_m(C^{-1}A)\|$.

In case the eigenvalues of $C^{-1}A$ are real and positive and if a positive lower (a) and an upper bound (b) are known of the spectrum, then we find that $\{\tau_l\}$ should be chosen such that $\max_{a \leq \lambda \leq b} |P_m(\lambda)|$ is minimized over all $P_m \in \Pi_m^0$, i.e. over the set of polynomials of degree m satisfying $P_m(0) = 1$.

The solution to this min-max problem is well known,

$$P_m(\lambda) = T_m\left(\frac{b+a-2\lambda}{b-a}\right) \Big/ T_m\left(\frac{b+a}{b-a}\right),$$

where $T_m(z) = \frac{1}{2}[(z+(z^2-1)^{1/2})^m + (z-(z^2-1)^{1/2})^m] = \cos(m \arccos z)$ are the Chebyshev polynomials of the first kind. The corresponding values of τ_l satisfy

$$\frac{1}{\tau_l} = \frac{b-a}{2}\cos\theta_l + \frac{b+a}{2}, \qquad \theta_l = \frac{2l-1}{2m}\pi, \qquad l = 1, 2, \ldots, m,$$

the zeros of the polynomial. The corresponding method is referred to as the Chebyshev (one-step) acceleration method. It is an easy matter to show that

$$1/T_m\left(\frac{b+a}{b-a}\right) \leqq 2\varrho^m, \quad \text{where } \varrho = (b^{1/2}-a^{1/2})/(b^{1/2}+a^{1/2}).$$

This implies that if the number of iterations satisfies $m \geqq \ln \varrho^{-1} \ln(2/\varepsilon)$, i.e. in particular if,

$$m \geqq \tfrac{1}{2}(b/a)^{1/2} ln(2/\varepsilon) \qquad \varepsilon > 0, \tag{2.2}$$

then $\|e^m\|/\|e^0\| \leqq \varepsilon$.

The disadvantage with this method is that to make it effective one needs accurate estimates of a and b and we need to determine m beforehand (which, however, can be done by (2.2)). The method cannot utilize any special distribution of the eigenvalues in the spectrum (as opposed to the conjugate gradient method). More important, however, is that this method is actually numerically unstable (similarly to an explicit time stepping method for initial value problems when too large time steps are taken). However, one may prove (see [3]) that with some particular permutation of the parameters, their instability effect can be avoided.

There exists a three term recursion form of the Chebyshev acceleration method,

$$x^{l+1} = \alpha_l x^l + (1-\alpha_l)x^{l-1} - \beta_l C^{-1} r^l, \qquad l = 1, 2, \ldots,$$

where $x^1 = x^0 - \frac{1}{2}\beta_0 C^{-1} r^0$.

Here the parameters are chosen as $\beta_0 = 4/(a+b)$,

$$\alpha_l = \frac{a+b}{2}\beta_l, \qquad \beta_l^{-1} = \frac{a+b}{2} - \left(\frac{b-a}{4}\right)^2 \beta_{l-1}, \qquad l = 1, 2, \ldots.$$

Hence we do not have to determine the number of steps beforehand. It has been shown in [4] that this method is numerically stable. A similar form of method was proposed a long time ago, see Varga [2] and the references cited therein.

It is interesting to note that the parameters approach stationary values. If $C^{-1}A = I - B$ and B has eigenvalues in $[-\varrho, \varrho]$, $\varrho = \varrho(B) < 1$ (the spectral

radius of B), then

$$a = 1-\varrho, b = 1+\varrho \quad \text{and} \quad \alpha_l \to \frac{a+b}{2}\beta_l \to 2/[1+(1-\varrho^2)^{1/2}],$$

which is recognized as the parameter ω_{opt} of the optimal SOR method (see section 3). Young [5] has proved that the asymptotic rate of convergence is retained even if one uses the stationary values throughout the iterations.

For the case of complex eigenvalues of $C^{-1}A$ with positive real parts and contained in an ellipse one may choose parameters similarly. See [4] and [6] for details.

Perhaps the main thrust during the 70's has been in using the conjugate gradient method as an acceleration method. Already much has been written on the subject; we refer to [4] for a historical account, to [1] for a recent exposition of the preconditioned conjugate gradient PCG method and to [7] for a recent survey of generalized and truncated gradient methods for non-symmetric and indefinite matrix problems.

The advantage with conjugate gradient methods is that they are self-adaptive; the optimal parameters are calculated by the algorithm so that the error in energy norm $\|e^l\|_{A^{1/2}} = \{(e^l)^T A e^l\}^{1/2}$ is minimized. This applies to a problem where C and A are symmetric and positive definite (SPD) or more generally, if $C^{-1}A$ is similarly equivalent to an SPD matrix. Hence there is no need to know any bounds for the spectrum. One may prove that $\|x-x^m\|_{A^{1/2}} \leqq \varepsilon\|x-x^0\|_{A^{1/2}}$, if

$$m = \text{int}\{\tfrac{1}{2}\kappa^{1/2} ln(2/\varepsilon)+1\}, \tag{2.3}$$

where $\kappa = \max_i\lambda_i/\min_i\lambda_i$, the spectral condition number. Here $\{\lambda_i\}$ is the set of eigenvalues of $C^{-1}A$. The number of iterations in (2.3) is an upper bound, frequently overly pessimistic, in particular when the spectrum is clustered (see for instance [8] and [4]). For an SPD problem with $C = D_A$ as before we now get $k = O(h^{-1})$, an improvement of one order over the basic method. This may be further improved upon by use of so-called modified incomplete factorization methods (see sections 3 and 4). One may prove that then (for second order selfadjoint elliptic problems, in $\mathbb{R}^d$) $k = O(h^{-1/2})$. This results in a total cost of $O(N^{1.25})$, $d = 2$ and $O(N^{1.17})$, $d = 3$, where N is the degree of freedoms. Storage is of optimal order, $O(N)$, and typically we need about twice as much storage as needed for A alone. For time dependent problems with time step, say $k = O(h)$, we may improve the costs above to $O(N^{1.125})$, $d = 2$ and $O(N^{1.083})$, $d = 3$, respectively. See [1] and [9] for details. This means that such iterative methods are of almost optimal order of computational complexity for that type of problems.

3. Classical preconditioning methods.

Let $A = D-L-U$ be a splitting of A where D is the (block) diagonal part and L, U the lower and upper (block) triangular parts, respectively of A. We get the simplest iterative method of the form (1.2) if we choose $C = D$, the (block) Jacobi method. With $C = \omega^{-1}D-L$, ω a parameter, $0 < \omega < 2$, we get the successive (under or over) relaxation method, studied extensively in [2] and [10]. If A is a nonsingular M-matrix ($a_{ij} \leqq 0, i \neq j, A^{-1} \geqq 0$), then the under-relaxed method ($0 < \omega < 1$) converges with $\tau_l = 1$, because $A = C-R$, $R = (\omega^{-1}-1)D+U$ is a so-called regular splitting of A (see section 4). Under certain other assumptions about A (D nonsingular, eigenvalues of $B = D^{-1}(L+U)$ real, $\varrho(B) < 1$ and A consistently ordered – note that C is a function of the ordering – which means that A may be permuted to block tridiagonal form), then the method converges with the same asymptotic rate of convergence as the Chebyshev acceleration method, if ω is chosen optimally $\omega = \omega_{\text{opt}}$, $1 < \omega_{\text{opt}} < 2$ (hence an overrelaxed method, see comment in the previous section).

The eigenvalues of $C^{-1}A$ are in this case complex and with $\omega = \omega_{\text{opt}}$, they are distributed around a circle. This means that the method cannot be polynomially accelerated then. The efficiency of the method is critical to the choice of ω.

The symmetric successive overrelaxation method (SSOR).

Solving $C\delta = r$ with $C = \omega^{-1}D-L$ means a forward sweap. There is a symmetric version of this relaxation method, where

$$C = (\omega^{-1}D-L)(\omega^{-1}D)^{-1}(\omega^{-1}D-U), \qquad 0 < \omega < 2. \tag{3.1}$$

Hence, in this method, the symmetric (over)relaxation method, we perform first a forward and then a backward sweap every iteration step. Again $A = C-R$ is a convergent regular splitting if A is a nonsingular M-matrix and $0 < \omega < 1$.

More generally, the generalized SSOR method (1.2) with

$$C = (\tilde{D}-L)(2\tilde{D}-D)^{-1}(\tilde{D}-U) \tag{3.2}$$

is convergent with $\tau_l = 1$ if $\tilde{D}$ is symmetric and if A and $2\tilde{D}-D$ are SPD. Further the eigenvalues of $C^{-1}A$ are positive. (For a proof, see [11]. Note that we get (3.1), apart from the factor $1/(2-\omega)$, from (3.2) by letting $\tilde{D} = \omega^{-1}D$.) That the eigenvalues are positive means that we may accelerate the method with a polynomial accelerator, such as described in section 2.

In particular when accelerated by the conjugate gradient method, the SSOR method is remarkably insensitive to the choice of ω; for second order difference equations one chooses $\omega = 2/[1+\zeta h]$, $\zeta > 0$ a parameter and h a mesh size parameter (see [12]).

One may in fact prove (see [12] and [1]) that a lower bound of the eigenvalues of $C^{-1}A$ is $[1+(1-\omega/2)^2\mu/\omega+\omega\delta]^{-1}$ and an upper bound is $1/(2-\omega)$, where $\delta = \max_{x\in\mathbb{R}^N}[x^TLD^{-1}L^Tx-\frac{1}{4}D]/x^TAx$ and $\mu = \max_{x\in\mathbb{R}^N}x^TDx/x^TAx$. If, for a difference equation with variable coefficients, one may order the equations so that the absolutely smaller entries come first, then $\|LD^{-1}L^T\|_\infty \leqq \frac{1}{4}$, i.e. $\delta \leqq 0$ (clearly $\delta \geqq -\frac{1}{4}$). The optimal value of ω which minimizes the spectral condition number of $C^{-1}A$ is $\omega^* = 2/[1+2\mu^{-1/2}(\frac{1}{2}+\delta)^{1/2}]$.

Under certain conditions it is proved in [1] and [12] that the above choice of ω renders a spectral condition number of $C^{-1}A$ of $O(h^{-1})$, versus $O(h^{-2})$ for A, if A is a second order difference matrix. These conditions are satisfied for a Dirichlet problem with smoothly varying coefficients; for other problems, such as with Neuman boundary conditions, the reduction of the condition number is sometimes less dramatic (except for certain orderings), however. For a further discussion about the SSOR method, see [13] and [1].

We shall now describe a class of methods for which one may construct arbitrarily accurate preconditioners. They are simply based on the factorization of A into triangular factors but performed incompletely.

Preconditioning by incomplete (pointwise) factorization.

The classical incomplete factorization methods as originally discussed in [14], [15] and in [16] and [17] can be presented shortly in the following form:

(i) *Incomplete factorization by position.*

Before factorization is started, choose a set of index pairs which is a subset of the complete set $\{(i,j), 1 \leqq i \leqq N, 1 \leqq j \leqq N\}$. Frequently $(i,j)\in J$ if and only if $a_{ij} \neq 0$. J is called the masking set, and it always includes the set of diagonal indices, $(i,i), 1 \leqq i \leqq N$.

The incomplete factorization principle (IC) is the following.

At the rth stage we use the pivot row $a_{rj}^{(r)}$, $j = r,\ r+1,\ldots,N$ to eliminate nonzeros in the lower triangular part, $j = r,\ i = r+1,\ldots,N$ of the remaining part of the matrix. We get $a_{ij}^{(r+1)} = a_{ij}^{(r)} - a_{ir}^{(r)}(a_{rr}^{(r)})^{-1}a_{rj}^{(r)}$, $r+1 \leqq i \leqq N$, $j \leqq r+1$, if $(i,j)\in J$. Otherwise, if $(i,j)\notin J$ we neglect the possible nonzero entry, or in the *modified version* (MIC) of this algorithm, we add it to the diagonal (see [9] and [1]). In the latter case the rowsums are preserved.

In this way, fill-in is neglected and hence cannot give rise to fill-in caused by previous fill-ins.

(ii) *Incomplete factorization by value.*

This is performed as above but nonzero entries are accepted only if their absolute values are large enough

$$|a_{ij}^{(r+1)}| \geqq c\{a_{ii}^{(r)}a_{jj}^{(r)}\}^{1/2},$$

where $c > 0$ is a (small) parameter. For details, see [17a] and [17b].

The existence of such factorizations of unmodified (IC) type (i.e. where fill-in is deleted and not used for modification) follows easily for M-matrices (i.e. it follows that all pivot entries are nonzero). For the modified version, modified according to the rowsum criterion, one assumes that A is a diagonally dominant M-matrix see [9] and [1]. The existence of incomplete factorizations, modified for a general M-matrix (i.e. not necessarily diagonally dominant) was recently proved in [18] and [34]. Here one utilizes an arbitrary positive vector v for which $Av > 0$ in a generalized rowsum criterion, so that the preconditioning matrix satisfies $Cv = Av$ for this vector. It is known that a matrix is an M-matrix if and only if such a vector v exists.

In still more general factorizations, the relaxed incomplete factorization, the rowsum criterion is satisfied partly only and according to a relaxation parameter. These methods have an interesting effect for the clustering of the eigenvalues of $C^{-1}A$, and hence for the rate of convergence of the preconditioned conjugate gradient method, see [32]. For the unmodified method, see [19].

Under quite general assumptions (see [9] and [1]) one may prove that for second order symmetric difference equations, the modified incomplete factorization reduces the spectral condition number by an order of magnitude, as was the case for the SSOR method for particular classes of such problems.

There exists also a simpler version of modified incomplete factorization, which may be looked upon as a generalization of the SSOR method, in the respect that one chooses a diagonal matrix of parameters instead of just one parameter as in the classical method. It turns out, however, that these parameters are best determined by a recursion, depending on only one parameter. The method has the form (see [22])

$$C = (\tilde{D} - L)\tilde{D}^{-1}(\tilde{D} - U), \quad \text{where } \tilde{D} = \operatorname{diag}((\tilde{d}_i))$$

is diagonal and is calculated so that $Ce = (A + \zeta h^2 D)e$ where $e = (1, 1, \ldots, 1)^T$ and $\zeta > 0$ is a perturbation parameter. It follows that $\tilde{D}e = (1 + \zeta h^2)De - L\tilde{D}^{-1}Ue$ or $\tilde{d}_i = (1 + \zeta h^2)d_i - \sum_{k=1}^{i-1} l_{i,k}\tilde{d}_k^{-1}u_{k,i}$, $i = 1, 2, \ldots, n$, where $D = \operatorname{diag}(d_i)$. The method was discussed in [23] and [22] for elliptic difference equations and was based on the strongly implicit (SIP) method in [24]. It is frequently referred to as the DKR method. In [22] it was also used as a PCG method for the first time. The positive perturbation $\zeta h^2 D$ of A enables one to prove that the method reduces the order of the spectral condition number under essentially the same restrictions as the classical SSOR method. However it was found in [22] that it was remarkably insensitive to the choice of the parameter ζ. For a further discussion about these perturbations, see [25a].

It should be noted that this generalized SSOR method is identical with the modified incomplete factorization method for 5-point (2D) or 7-point (3D) elliptic difference matrices, if the masking set $J = \{i, j; a_{i,j} \neq 0\}$ and if a rowwise (or columnwise) ordering of the meshpoints is used.

For an interesting trick to improve the computer implementation of the methods above, see [25b] and [1].

For special types of problems (essentially separable elliptic problems) preconditioners based on fast elliptic solvers have also been used, see [26], [27] and [28].

The results of this section may be further extended to more general classes of symmetric finite element matrices (which are not M-matrices) by the method of spectral equivalence; see [1] for a recent exposure of this topic. The idea goes back to [20] and [21].

4. Incomplete factorization methods for matrices partitioned into block form.

In recent time much effort has been devoted to the construction of still more effective and generally applicable preconditioners than those described in the previous section. It is then natural to look at matrices partitioned into blocks and to construct preconditioners based on approximate factorization of the block matrices where inverses of diagonal block matrices are approximated by sparse matrices.

As an introduction to this consider a tridiagonal $n \times n$ matrix A ($a_{ik} = 0$ if $|i-k| > 1$):

$$A = D_A - L_A - U_A, \tag{4.1}$$

with the usual definition of D_A, L_A and U_A. Let A be factorized as $A = LD^{-1}U$, where $L = D - L_A$, $U = D - U_A$. We then get $A = D - L_A - U_A + L_A D^{-1} U_A$ or $D = D_A - L_A D^{-1} U_A$. From this it follows the recursion

$$\text{(4.2) ALGORITHM 1:} \quad d_1 = a_{1,1}, \quad d_i = a_{i,i} - a_{i,i-1} d_{i-1}^{-1} a_{i-1,i}, \quad i = 2, 3, \ldots, n.$$

(The existence of this will be discussed later.)

A division free variant is $A = LDU$, $L = D^{-1} - L_A$, $U = D^{-1} - U_A$ where

$$\text{(4.3) ALGORITHM 2:} \quad d_0 = 0, \quad d_i = (a_{i,i} - a_{i,i-1} d_{i-1} a_{i-1,i})^{-1}, \quad i = 2, 3, \ldots, n.$$

The solution of $Ax = b$ is performed by the usual forward and backward substitution method, i.e. (for the division free variant),

$$Ly = b, \text{ i.e. } y_1 = d_1 b_1, \quad y_i = d_i (b_i - a_{i,i-1} y_{i-1}), \quad i = 2, \ldots, n$$

and

$$(I - DU_A)x = y, \text{ i.e. } x_n = y_n, \quad x_i = y_i - d_i a_{i,i+1} y_{i+1}, \quad i = n-1, n-2, \ldots, 1.$$

Note that in this variant, no divisions occur during the forward and backward substitutions.

Now consider the case when $a_{i,j}$ are block matrices. Then the above recursions are valid with no change. However, in the recursion (4.2) and (4.3) there appears inverses of matrices and it follows that in general they are full matrices, making too large a demand on computer storage and CPU-time. The idea, first suggested in [29] and later independently in [30a], [30b] and [11], is now to approximate the inverse by sparse matrices. For difference matrices, the matrices $a_{i,j}$ in (4.1) are typically banded matrices and the natural choice turns out to be to approximate all matrix inverses in the recursion by sparse banded matrices. (In the papers above tridiagonal approximations were dealt with, see [31] and [32] for the general case of a p-banded approximation of a not necessarily symmetric matrix.)

There are now two pertinent questions to ask:

(i) How accurate are such matrix approximations?

(ii) Can the p-banded approximation be calculated efficiently (i.e. without having to calculate the whole inverse, for instance)?

The answer to the first question (at least for matrices with a small bandwidth) is found in [33]: The entries $G^{-1}_{i,j}$ of the inverse of a p-banded matrix G decay at least (in the SPD case) away from the main diagonal as

$$C\{[1-\sqrt{(a/b)}]/[1+\sqrt{(a/b)}]\}^{|i-j|/p}, \tag{4.4}$$

for some constant C, where a and b are the extreme eigenvalues of G. Now note that the diagonal block matrices one encounters in the recursion for a difference matrix are frequently strongly diagonally dominant, which implies that their condition numbers b/a are not large.

Consider for instance a tridiagonal matrix with entries $-(b-a)/4$, $(b+a)/2$, $-(b-a)/4$, $0 < a < b$. Then a and b are bounds of the extreme eigenvalues and the entries of the inverse matrix decay precisely as estimated by (4.4) for $p = 1$, showing that the estimate may be sharp. For the model difference matrix, with diagonal block matrix entries -1, 4, -1 we get $a = 2$, $b = 6$ and the rate of decay $(2+\sqrt{3})^{-|i-j|}$, i.e. a fast decay.

The second question above is answered affirmatively. An efficient algorithm for the calculation of the p-(block) bounded part of the inverse of a q-(block) banded (block) matrix H is based on the following steps (see [31]).

Algorithm ABI (approximate block inverse):

STEP 1. Calculate an (approximate) (block) factorization G of H where $G = LB^{-1}U$, $L = I - \tilde{L}B$, $U = I - B\tilde{U}$, B is (block) diagonal and $\tilde{L}$, $\tilde{U}$ are strictly lower and upper (block) triangular, respectively.

During this (by use of algorithm 1, say) use the same *ABI*-algorithm to calculate approximations of the inverses that occur (implicit recursive definition).

STEP 2. Observe that $G^{-1} = BL^{-1} + B\tilde{U}G^{-1}$ and $G^{-1} = U^{-1}B + G^{-1}\tilde{L}B$. These relations may be used for a recursive calculation of the p-banded (block) entries of G^{-1}, the lower and upper triangular parts in turn, starting with the bottom row and rightmost column and working upwards. The algorithm is given in [31]. (The order the calculations must take is not unique.) The computational complexity of the algorithm is not more than $(2p^2+p)n$ (matrix) multiplications and additions for a (block) matrix of order n. Compare this with the computational complexity $(2p+1)n$ of the calculation of a full *single* column of the inverse.

It is shown in [32] that the method based on algorithm 1 (combined with algorithm *ABI* is more robust than that based on algorithm 2. For instance, for problems with anisotropy, the diagonal dominance of the diagonal block matrices can be weak implying that their condition numbers are large. As it turns out, this causes less trouble in algorithm 1. On the other hand, if algorithm 2 has been used, only matrix-vector multiplications occur in the forward and backward solution, which means that this latter algorithm is better vectorizable (see section 5).

For a general matrix partitioned into block form, the pointwise incomplete factorization method as presented in Section 3, can be applied if we let the entries $a_{i,j}$ be block matrices. For the inverses of diagonal block matrices that occur during the factorization one may use the banded approximation described above (algorithm *ABI*). This method was presented in [18] and generalizes all methods previously described in this survey, such as the DKR-method, the pointwise incomplete factorization method and the above methods for block matrix incomplete factorization of tridiagonal block matrices.

Existence of incomplete block factorizations.

For the class of M-matrices, it has been proved independently in [34] and [18] that if $0 \leqq x_r \leqq (a_{rr}^{(r)})^{-1}$ and x_r is the approximation to be used for $(a_{rr}^{(r)})^{-1}$, then the factorization exists, i.e. all diagonal block entries $a_{rr}^{(r)}$ that occur during the factorization are nonsingular and nonnegative.

In [18] a simple proof of existence for modified versions also appeared. More recently, in [32] this result has been extended to the relaxed methods, which are methods where the modification is done only partly and subject to a parameter.

One may also prove that the incomplete versions (with no modification) leads to convergent regular splittings of the given matrix, see [34] and [18] but this is not the case for the modified versions. However, for symmetric positive definite matrices, the theory of regular splittings is less important. We want instead to prove that the preconditioning matrix is *SPD*.

However, it is easy to see that the bandmatrix part of an *SPD* matrix need not be *SPD*. Hence during the factorization we may loose positive definiteness. It follows that this cannot happen with algorithm 1 but it may happen with

algorithm 2 and similar algorithms, such as the one in [35], where the approximate inverse appears directly in the diagonal blocks. To see this, let B be an *SPD* M-matrix and let $[B]^{(p)}$ be the p-banded part of B. Then in general $R^{(p)} = B - [B]^{(p)}$ may be indefinite.

However, if we now modify this approximation, i.e. let $\tilde{B} = [B]^{(p)} + D^{(p)}$, where $D^{(p)}e = R^{(p)}e$, $e = (1, 1, \ldots, 1)^T$, then $\tilde{B}e = Be$. It follows that $\tilde{B}$ is then *SPD*. Hence, the modification of a p-banded approximate inverse preserves positive definiteness and is hence to be recommended. Besides it has the property of decreasing the spectral condition number by an order of magnitude, a property not shared by the unmodified method.

Other block-matrix preconditioning techniques.

Other effective preconditioning methods for elliptic difference or finite element matrices are based on

a) a particular ordering of the mesh points,

b) particular choices of finite element basis functions.

The recursive odd-even reduction ordering turns out to be very effective for problems with several identical substructures and for parallel computation. For a recent discussion, see [36]. The similar ordering, the checker board ordering, also reduces the matrix order significantly at little cost and may also be used recursively, if it is coupled with the technique of spectral equivalence, to get problems of increasingly smaller orders to solve for. Work on this is in progress.

In [37] a hierarchical choice of basis functions (so called p-version) was combined with preconditioning to form a multigrid method of two-level type. A recursive choice of basis functions (h-version) was used in [38] to derive an algorithm of almost optimal order $O(N \log N)$ of computational complexity, where N is the degree of freedoms. These methods do not require any regularity of the solution as opposed to multigrid methods of classical type.

For the use of a block incomplete factorization method as a smoother for multigrid methods, see [30b].

Several methods based on substructuring of the given domain and coupled with fast elliptic solvers, have appeared in recent years, see for instance [39] and [40a]. It may be proved that, for certain problems, methods of optimal order of computational complexity may be derived in this way.

For subdomain decomposition methods based on projections, see [40b] and the references cited therein.

5. Vectorizable and parallelizable preconditioning methods.

In this section we shall describe some vectorizable variants of preconditioning methods. Because supercomputers with vector and/or parallel processing facilities

are becoming available to an increasing extent for scientific computation, this is a topic of much current interest.

At first it may be appropriate to describe shortly what the distinction between vector processors and parallel processors is. Parallel computers consist of an array of processors that is capable of applying the same instruction (e.g. a multiplication) to one operand-pair per processor. This is called single instruction multiple data stream. A pipeline computer is a single processor that is capable of executing the same instruction very fast a number of times in a row provided the operands are stored in consecutive or equi-distant memory locations. If an instruction consists of q subparts, one may let part $1, 2, \ldots, q$ of the instruction be performed in one cycle time on each pair of q operands. Pipeline computers need an intitial time to get the "assembly line" started up.

Methods for achieving parallel operation depend upon replicating the instruction stream and/or data stream. Some examples of calculations which can be performed efficiently in parallel are:

A scalar times a vector can be performed in parallel in one cycle time, provided one has N processors.

Matrix-vector multiplications can be performed in parallel as inner products; for a full matrix each row can be multiplied by the vector independently of each other; for a sparse matrix it is frequently more appropriate to use inner products of sub- and super-diagonals with the vector. For a discussion about this, see [41]. Note that if $\lceil n/2 \rceil$ ($\lceil a \rceil$ indicates the smallest integer larger or equal to a, sometimes called the ceiling of a) processors are available an inner product can theoretically be performed in a CPU-time of $\lceil \log_2 n \rceil$ cycle times, if the processors are connected as in a binary tree.

We see that convergence acceleration methods such as the conjugate gradient method are quite suitable for vector and parallel computation. Each iteration of the unpreconditioned CG-method involves two innerproducts, three scalar-vector multiplications, three vector additions, and one matrix-vector multiplication. In the preconditioned conjugate gradient method, however, each iteration requires also solving a system of linear equations with preconditioning matrix C such as in (1.2). The algorithms for incomplete factorization and for the forward and backward substitutions we have described in earlier sections are however recursive, i.e. sequential and of length n, if the order of the matrix is n. Using such algorithms on a vector machine in vector mode yields very low speedup as compared to solving them in scalar mode.

To alleviate this inefficiency several modifications of the algorithms have been proposed, some of which we shall describe here.

Consider at first the case of pointwise incomplete factorization. Then we may write C in the form

$$C = (I - \tilde{L})D(I - \tilde{U}), \tag{5.1}$$

where D is diagonal and $\tilde{L}, \tilde{U}$ are sparse and strictly lower and upper triangular matrices, respectively. Then

$$C^{-1} = (I-\tilde{U})^{-1}D^{-1}(I-\tilde{L})^{-1}$$

and $(I-\tilde{U})^{-1}$ and $(I-\tilde{L})^{-1}$ are now approximated by *truncated* Neumann series. This method was proposed in [42] for unmodified (IC) methods. It has the disadvantage of deteriorating the quality of an already from the beginning less accurate preconditioner. For a discussion, see [36].

Vectorization of forward and backward substitutions for block-matrices.

We shall now consider vectorization (or parallelization) of block-matrix incomplete factorizations. As an introduction we consider first the scalar tridiagonal matrix (4.1). Let it be factorized as $A = (I-\tilde{L})D(I-\tilde{U})$, where now $\tilde{L}, \tilde{U}$ consists of only one nonzero sub- and superdiagonal, respectively. Then $A^{-1} = (I-\tilde{U})^{-1}D^{-1}(I-\tilde{L})^{-1}$. Consider the first factor. Since $\tilde{U}$ is nilpotent of index n, $\tilde{U}^k = 0$, $k \geqq n$ and we have

$$(I-\tilde{U})^{-1} = I+\tilde{U}+\tilde{U}^2+\ldots+\tilde{U}^{n-1}. \tag{5.2}$$

We rewrite now this finite geometric expansion, formally adding the zero terms $\tilde{U}^k, k = n, \ldots, v$ if necessary, as

$$(I-\tilde{U})^{-1} = \prod_{r=0}^{s}(I+\tilde{U}^{2^r}) \tag{5.3}$$

where $s = \lceil \log_2 n \rceil - 1$ (and $v = 2^{s+1}-1$). Similarly we have $(I-\tilde{L})^{-1} = \prod_{r=0}^{s}(I+\tilde{L}^{2^r})$.

In solving $Ax = b$ we hence perform the recursive matrix-vector multiplications as $y = (I-\tilde{L})^{-1}b$ and $x = (I-\tilde{U})^{-1}Dy$, where y is calculated by *Algorithm PE (product expansion):*
$z = b$, $V = \tilde{L}$ (b and $\tilde{L}$ are input), for $r = 0$ step 1 until s do $z = (I+V)z$, $V = V^2$; ($y = (I-\tilde{L})^{-1}b$ is output).

In a similar way, $x = (I-\tilde{U})^{-1}Dy$ is calculated.

Note that all matrices V that occur in the algorithm, are single sub- (super-) diagonal. The length of the recursion in Algorithm PE is $s+1$ and the cost to calculate x is $2s+3$ inner products (assuming that we perform the matrix-vector multiplications as sub- (super-) diagonals times vector).

REMARK 5.1. When we apply algorithm PE on a general strictly triangular matrix L (or U), it generalizes the method proposed in [42].

Consider the length (v) of the recursions that occur in a computer algorithm (α)

performing a specific computational task for the solution of a problem (P). The shortest length, $\min_\alpha v$ of all algorithms performing this task is called the *depth* of P.

The depth of the problem $Ax = b$, A defined by (4.1), is $2\lceil \log_2 n \rceil + 1$ and an algorithm achieving this depth is the PE algorithm. Note that the length of the algorithm based on the Neumann expansion (5.2) is $2n - 1$. (We assume of course that $n \geqq 2$.) The CPU-time consists of as many cycles as the length, if n parallel processors are available. Note that on a sequential machine, these algorithms are less efficient than the forward and backward solution algorithms.

The algorithm PE (and extensions to block matrices) was presented in [31] and [32]. It is equivalent to the recursive doubling algorithm, presented for tridiagonal matrices in [43]. See also [44] and [45].

Consider now a block incomplete factorization matrix C, where the inverses of pivot diagonal block matrices $a_{rr}^{(r)}$ that occur are assumed to be tridiagonal matrices. This would be the case for the elliptic difference equation for instance, if we use approximations $[B]^{(p)}$ with halfbandwidth $p = 1$ for the inverses. This means that for each block diagonal matrix we have avoided the forward and backward solution algorithms. We call the combination of incomplete block matrix factorization with diagonal blocks approximated by tridiagonal matrices and with use of algorithm PE for the solution, algorithm 3. Hence

ALGORITHM 3 Incomplete block matrix factorization with a tridiagonal array of diagonal block matrices, factorized and stored, and solved for by algorithm PE.

Note that in algorithm 3, positive definiteness will be preserved (assuming the given matrix A is positive definite). As we know from section 4, algorithm 2 (the inversefree variant) is also parallel in this respect (no forward and backward recursions occur within the diagonal block matrices). In fact, algorithm 2 is less expensive, because each matrix-vector multiplication costs only $2p+1$ multiplications per diagonal block. Furthermore it is applicable not only for tridiagonal matrices (where the halfbandwidth $p = 1$). However, algorithm 2 is less robust as we pointed out in section 4.

In this way, using algorithm 2 or 3, we have achieved vectorization within each block (on the local level). On the global level, between the block rows, there will however still occur recursions in the forward and backward solutions.

To achieve parallelism also here we could use the algorithm PE, assuming we are dealing with a block tridiagonal matrix, i.e. the entries $a_{i,j}$ in (4.1) are block matrices. However, now it is clear that the corresponding matrices $\tilde{L}, \tilde{U}$ and D in (5.1) consist of sub-, super- and diagonal *block* matrices, which may even be full. If we have used a sparse block factorization based on algorithm 2, they are sparse, however, but the powers in (5.3) for instance quickly become full. To preserve sparsity we could then use a masking operation, similar to the one used for the incomplete factorization, when we calculate the powers. The

corresponding algorithm 4 (*APE*) approximate product expansion is discussed in [32] and [31].

Vectorization of incomplete block matrix factorizations.

Up till now we have not discussed the recursiveness which occurs in the factorization itself. Note that this is of length n for a matrix of order n. For problems with many right hand sides, this initial cost becomes less important and, furthermore, the forward and backward substitutions occur for each iteration but the factorization is done only once. It could however still be of interest to consider vectorization of the factorization, in particular if the resulting factorizations are also efficient for the solution itself.

To this end we shall discuss three methods to avoid a recursive factorization. The first one turns out to be less efficient but the other two are also quite efficient as solution methods.

On polynomial preconditioners.

The obvious method to avoid recursion in the factorization is to avoid the factorization itself. This can be achieved by polynomial preconditioners, i.e. matrix polynomials approximating A^{-1}. We consider here only the simplest choice which is based on Neumann series expansion, and we assume that A is *SPD*. For a discussion of more general polynomial preconditioners and their relation to the conjugate gradient method, see [46], [47] and [36].

We have $A = (I-B)D$ where D is the (block) diagonal part of A and $B = I - AD^{-1}$. Hence $A^{-1} = D^{-1}(I-B)^{-1}$ and we approximate $(I-B)^{-1}$ by $Q_r(B) = I + B + B^2 + \ldots + B^r$. Then when solving $Ax = b$ we consider

$$Q_r(B)(Ax-b) = Q_r(B)[(I-B)y-b] = 0, \tag{5.4}$$

where $y = Dx$ and use iteration on (5.4).

First we consider the spectral condition number of $F(B) = Q_r(B)(I-B)$. Assume then that B has real eigenvalues in the interval $[-1+\delta, 1-\delta]$, $0 < \delta < 1$ (i.e. the spectral radius of B satisfies $\varrho(B) \leqq 1-\delta$), where δ is very small. We have

$$F(B) = I - B^{r+1}. \tag{5.5}$$

Hence the eigenvalues of $F(B)$ are in the interval $[1-(1-\delta)^{r+1}, 1]$ if r is odd and in $[-1+(1-\delta)^{r+1}, 1+(1-\delta)^{r+1}]$ if r is even.

We consider only the most favourable case, i.e. r is odd. Then for the spectral condition number q of $F(B)$ we have

$$q \leqq 1/[1-(1-\delta)^{r+1}] \sim \frac{1}{(r+1)\delta}, \qquad \delta \to 0 \quad (r \text{ is fixed}). \tag{5.6}$$

Assume now that as convergence acceleration method we have used a Chebyshev or conjugate gradient method for the calculation of the solution y

of (5.4). Then by (2.2), the number of iterations to achieve a relative accuracy of ε, is $\sim \frac{1}{2}q^{1/2}\ln(2/\varepsilon) = c_0(r+1)^{-1/2}$, where $c_0 = \frac{1}{2}\delta^{-1/2}\ln(2/\varepsilon)$. Assume that we calculate $Q_r(B)b$ initially. This can be done by an obvious algorithm at the cost of r matrix-vector multiplications by B (plus additions of vectors). For the calculation of $F(B)y^{(l)} = (I - B^{r+1})y^{(l)}$, where $y^{(l)}$ is the lth iteration, we have to perform $(r+1)$ matrix-vector multiplications. (Alternatively we may calculate B^{r+1} initially, which means we increase demand of storage and do not gain much in computation, because B^{r+1} is much fuller than B.)

Assume that the cost to perform a matrix multiplication by B is a_0 and the cost to perform all calculations during one iteration step, except the matrix-vector multiplication, is a_1. Then the total cost, initial and the cost to perform $c_0(r+1)^{-1/2}$ iterations is about

$$f(r) = a_0 r + [a_1 + (r+1)a_0]c_0(r+1)^{-1/2}.$$

We want to determine r such that $f(r)$ is minimized. We get

$$f'(r) = a_0 + \tfrac{1}{2}c_0[a_0(r+1) - a_1]/(r+1)^{3/2}. \tag{5.7}$$

We see that $r > 0$ only if $a_1 \geqq 2a_0$. In practice c_0 is quite large, so the first term a_0 in (5.7) can be neglected. Hence the optimal value of r is less than $a_1/a_0 - 1$, which means that it cannot be very large.

The algorithm above is an improvement over that in [48]. As pointed out in [48], we may also use other approximations of A instead of the diagonal part D. However, we see from the analysis above that this increases the value of a_0, hence making the optimal value of r smaller.

It was found in [48] by numerical tests for the elliptic difference equation that $r = 1$ was close to optimal. Hence $Q_r(B) = I + B$. In fact, we see that the spectral condition number of $I - B$ is about $2/\delta$, while that of $Q_1(B)(I - B)$ is about $\frac{1}{4}$ of this number. Hence the number of iterations will be about half of the number for $(I - B)y = b$ while the cost per iteration is somewhat less than double.

REMARK 5.2. We have seen that the optimal value of r is small because the cost a_0 for B is relatively large. Hence in this case it makes no sense to use algorithm PE for calculation of $Q_r(B)$ times a vector at each iteration.

A similar analysis as the one above will actually also show that for incomplete factorizations it is not effective in general to choose a very large bandwidth p, because the decrease in the number of iterations will not outweigh the, essentially linear, increase in cost with p per iteration. This was indeed found in [49] by numerical tests for the modified pointwise incomplete factorization (where p plays the rôle of the number of nonzero subdiagonals in L and U).

Recursive odd-even reduction orderings.

Here we mention an algorithm to achieve vectorization in the factorization for matrices which are initially given on block tridiagonal form. It is based on recursively reordering the unknowns in an odd-even fashion. It is described in [50] and a modification thereof in [36]. Because of space constraints we shall not go into details here. Numerical tests indicate that even on a sequential computer it can be about as efficient as incomplete block matrix factorization methods. The robustness of the method has not been examined yet.

An inversefree factorization for special problems.

Consider a block tridiagonal matrix

$$A = (A_{i,k}) \quad \text{with} \quad A_{i,k} = 0 \quad \text{for} \quad |i-k| > 1, \tag{5.8}$$

where the subdiagonal block matrices $A_{i,i-1}$ are assumed to be nonsingular (i.e. in particular square matrices) and having a sparse inverse.

This means that they are essentially diagonal matrices. Now we multiply the ith row of (5.8) by $A_{i,i-1}^{-1}$, $i = 2, 3, \ldots, n$ so that the resulting matrix has identity matrices as subdiagonal blocks. Hence we now consider a matrix A in (5.8) where $A_{i,i-1} = I$, $i = 2, \ldots, n$. The following factorization was suggested in [51] following an idea in [52] and [53]. The same trick has also been used in [43].

We factor A as in (4.2) and get diagonal block matrices

$$D_1 = A_{1,1}, \qquad D_r = A_{r,r} - A_{r,r-1}D_{r-1}^{-1}A_{r-1,r} = A_{r,r} - D_{r-1}^{-1}A_{r-1,r}, \quad r = 2, 3, \ldots, n. \tag{5.9}$$

Write now

$$D_r = X_{r-1}^{-1}X_r, \qquad X_0 = I, \qquad X_1 = A_{1,1}. \tag{5.10}$$

Then from (5.9) it follows

$$X_r = X_{r-1}A_{r,r} - X_{r-2}A_{r-1,r}, \qquad r = 2, 3, \ldots, n. \tag{5.11}$$

Hence in this way we have factorized A without having to invert any matrices. The matrices X_r get fuller and fuller, but for a difference matrix, where $A_{r,r}$ has halfbandwidth 1, the bandwidth of X_r grows only linearly. The halfbandwidth of X_r is r if $A_{i-1,i}$ are bandmatrices with halfbandwidth $\leqq 2$.

More severe is that for some problems the condition number of the matrices

grows exponentially and hence the calculation becomes numerically unstable. This was pointed out in [53] where it was suggested to break the problem into subproblems (actually similar to the parallel shooting or marching method). To get some insight into this, consider an elliptic difference equation, where $A_{i,i+1} = I$ and

$$A_{i,i} = \begin{bmatrix} -4 & 1 & 0 & \dots & 0 \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ 0 & & 1 & & -4 \end{bmatrix}, i = 2, 3, \dots, n,$$

$A_{1,1} = \frac{1}{2}A_{2,2}$ and all matrices are of order n. Then

$$X_r = 2X_{r-1}G - X_{r-2}, \quad r = 2, 3, \dots, \quad X_0 = I, \quad X_1 = G = \tfrac{1}{2}A_{2,2}. \tag{5.12}$$

This is the recursion for the Chebyshev polynomials $T_r(z)$ so $X_r = T_r(G)$. The condition number of X_r grows as $T_r(b)/T_r(a)$, where a and b are the extreme eigenvalues of G, i.e. $b \simeq 3$, $a \simeq 1$. Hence $T_r(b)/T_r(a) \sim \frac{1}{2}(3+\sqrt{8})^r \simeq \frac{1}{2}(5.83)^r$, $r \to \infty$.

Now we observe that in elliptic problems with constant coefficients (at least in parts of the domain) the diagonal matrices D_i approach a stationary value quickly. (Its limit value is $G+(G^2-1)^{1/2}$.) Hence we may stop early say for $r = 3$ and let $D_s = D_3$ for $s = 3, \dots, n$.

The resulting factorization (which has a negligible cost) will now only be approximate but we will use it as a preconditioner and as such it is very accurate indeed. More about this method will be reported elsewhere.

For some other works on parallel and vectorized computations for iterative methods, see [54], [55], [56], [57], [58] and [59].

Acknowledgement.

I thank the referee for his suggestion to add some references in the text.

REFERENCES

1. O. Axelsson and V. A. Barker, *Finite Element Solutions of Boundary Value Problems. Theory and Computation.* Academic Press, New York, 1984.
2. R. S. Varga, *Matrix Iterative Analysis.* Prentice Hall, New Jersey, 1962.
3. V. I. Lebedev and S. A. Finogenov, *On the order of choice of the iteration parameters in the Chebyshev cyclic iteration method.* Zh. Vychislit. Mat. i Mat. Fiz. 11 (1971, p. 425, and 13 (1973), p. 18.
4. O. Axelsson, *Solution of linear systems of equations: Iterative methods* in *Sparse Matrix Techniques* (editor V. A. Barker), LNM #572, pp. 1–51, Springer-Verlag, 1977.
5. D. M. Young, *Second degree iterative methods for the solution of large linear systems*, J. Approx. Theory 5 (1972), pp. 137–148.

6. T. A. Manteuffel, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math. 28 (1977), pp. 307–327.
7. Y. Saad and M. H. Schultz, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*. Research Report RR-283, Department of Computer Science, Yale University, 1983.
8. D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, Mass., 1973.
9. I. Gustafsson, *Modified incomplete Cholesky (MIC) methods*, in *Preconditioning Methods: Analysis and Applications* (editor D. J. Evans), pp. 265–293, Gordon and Breach, New York, 1983.
10. D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.
11. O. Axelsson, S. Brinkkemper and V. P. Il'in, *On some versions of incomplete block-matrix factorization iterative methods*, Lin. Alg. Appl. 58 (1984), pp. 3–15.
12. O. Axelsson, *On preconditioning and convergence acceleration in sparse matrix problems*, CERN 74–10, Geneva, 1974.
13. L. A. Hageman and D. M. Young, *Applied Iterative Methods*. Academic Press, New York, 1981.
14. R. S. Varga, *Factorizations and normalized iterative methods*, in *Boundary Problems in Differential Equations* (editor R. E. Langer), pp. 121–142, The University of Wisconsin Press, Madison, Wisconsin, 1960.
15. J. A. Meijerink and H. A. van der Vorst, *An iterative method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp. 31 (1977), pp. 148–162.
16. O. Axelsson and N. Munksgaard, *A class of preconditioned conjugate gradient methods for the solution of a mixed finite-element discretization of the biharmonic operator*, Int. J. Numer. Math. Eng. 14 (1978), pp. 1001–1019.
17a. D. S. Kershaw, *The incomplete Choleski-conjugate gradient method for the iterative solution of systems of linear equations*, J. Comput. Phys. 26 (1978), 43–65.
17b. O. Axelsson and N. Munksgaard, *Analysis of incomplete factorizations with fixed storage allocation*, in *Preconditioning Methods: Analysis and Applications* (editor D. J. Evans), pp. 219–241, Gordon and Breach, New York, 1983.
18. O. Axelsson, *A general incomplete block-matrix factorization method*, Lin. Alg. Appl., to appear.
19. H. A. van der Vorst, *Preconditioning by incomplete decompositions*, Ph.D. Thesis, University of Utrecht, 1982.
20. E. G. D'Yakonov, *On the iterative method for the solution of finite difference equations*, Dokl. Akad. Nauk SSSR, 138 (1961), 522–525.
21. J. E. Gunn, *The solution of elliptic difference equations by semi-explicit iterative techniques*, SIAM J. Numer. Anal. Ser B2 (1964), 24–45.
22. O. Axelsson, *A generalized SSOR method*, BIT 12 (1972), 443–467.
23. T. Dupont, R. P. Kendall and H. H. Rachford, Jr., *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal. 5 (1968), 559–573.
24. H. L. Stone, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Numer. Anal. 5 (1968), 530–558.
25a. R. Beauwens, *On Axelsson's perturbations*, Linear Algebra Appl., to appear.
25b. S. C. Eisenstat, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Stat. Comput. 2 (1981), 1–4.
26. P. Concus, G. H. Golub and D. P. O'Leary, *A generalized conjugate gradient iteration for the numerical solution of elliptic partial differential equations*, pp. 304–322 in *Sparse Matrix Computations* (J. R. Bunch and D. J. Rose, eds), Academic Press, New York, 1976.
27. J. Douglas, Jr. and T. Dupont, *Preconditioned conjugate gradient iteration applied to Galerkin methods for a mildly-nonlinear Dirichlet problem*, pp. 323–333 in same volume as ref. [26].
28. R. Bank, *An automatic scaling procedure for a D'Yakonov-Gunn iteration scheme*, Linear Algebra Appl. 28 (1979), 17–33.
29. R. R. Underwood, *An approximate factorization procedure based on the block Cholesky decomposition and its use with the conjugate gradient method*, Report NEDO-11386, General Electric Co., Nuclear Energy Div., San Jose, California (1976).
30a. P. Concus, G. H. Golub and G. Meurant, *Block preconditioning for the conjugate gradient method*, Report LBL-14856, Lawrence Berkeley Lab., University of California (1982), to appear (abridged) in SISSC.
30b. R. Kettler, *Analysis and comparison of relaxed schemes in robust multigrid and preconditioned*

conjugate gradient methods, in *Multigrid Methods* (eds W. Hackbusch and U. Trottenberg), LNM 960, pp. 502–534, Springer Verlag, 1982.
31. O. Axelsson, *A vectorizable variant of a block incomplete factorization method*, Proceedings of the VIth International Conference on the Numerical Solution of Fuid Flow Problems, Jan. 1984, The University of Texas at Austin, John Wiley and Sons, to appear.
32. O. Axelsson, *Incomplete block matrix factorization preconditioning methods. The ultimate answer?* to appear in J. Comp. Appl. Math.
33. S. Demko, W. F. Moss and P. W. Smith, *Decay rates for inverses of band matrices*, to appear in Math. Comp.
34. R. Beauwens and M. BenBouzid, *On block-OBV methods*, Part I, Commissariat aux energies nouvelles, Physique Théorique et Méthodes Numériques, Alger, 1983.
35. G. Meurant, *The block preconditoned conjugate gradient method on vector computers*, BIT 24 (1984), 623–633.
36. O. Axelsson, *A survey of vectorizable preconditioning methods for large scale finite element matrix problems*, CNA-190, February 1984, Center for Numerical Analysis, The University of Texas at Austin, Texas.
37. O. Axelsson and I. Gustafsson, *Preconditioning and two-level multigrid methods of arbitrary degree of approximation*, Math. Comp. 40 (1983), 214–242.
38. H. Yserentant, *On the multilevel splitting of finite element spaces*. Bericht Nr. 21, 1983, Institut für Geometrie und Praktische Mathematik, R.W.T.H., Aachen, Germany.
39. J. H. Bramble, J. E. Pasciak and A. H. Schatz, *Preconditioners for interface problems on mesh domains*, preprint, Cornell University, 1963.
40a. G. H. Golub and D. Mayers, *The use of pre-conditioning over irregular regions*, NA-83-27, Computer Science Department, Stanford University, California, December 1983.
40b. Q. V. Dinh, R. Glowinski, B. Mantel, J. Periaux and P. Perrier, *Subdomain solutions of non-linear problems in fluid dynamics on parallel processors*, in *Computing Methods in Applied Sciences and Engineering*, V, R. Glowinski and J. L. Lions (editors), North-Holland Publ. Comp., 1982.
41. N. K. Madson, G. H. Rodrigue and J. I. Karush, *Matrix multiplication by diagonals on a vector/parallel processor*, Information Processing Lett. 5 (1976), 41–45.
42. H. van der Vorst, *A vectorizable variant of some ICCG methods*, SIAM J. Stat. Comput. 3 (1982), 86–96.
43. H. S. Stone, *An efficient parallel algorithm for the solution of a tridiagonal linear system of equations*, J.A.C.M. 20 (1973), 27–38.
44. D. Heller, *A survey of parallel algorithms in numerical linear algebra*, SIAM Review 20 (1978), 740–776.
45. P. Dubois and G. Rodrigue, *An analysis of the recursive doubling algorithm*, in D. J. Kuck et al. (eds), *High Speed Computer and Algorithm Organization*, Academic Press, New York, 1977.
46. O. G. Johnson, C. A. Micchelli and G. Paul, *Polynomial preconditioners for conjugate gradient calculations*, RC 9208, IBM Th. J. Watson Research Center, Yorktown Heights, New York, 1982.
47. Y. Saad, *Practical use of polynomial preconditionings for the conjugate gradient method*. Research Report RR 283, Yale University, Department of Computer Science, 1983.
48. P. F. Dubois, A. Greenbaum and G. H. Rodrigue, *Approximating the inverse of a matrix for use in iterative algorithms on vector processors*, Computing 22 (1979), 257–268.
49. I. Gustafsson, *Stability and rate of convergence of modified incomplete Cholesky factorization methods*, Ph.D. Thesis, Department of Computer Science, Göteborg, Sweden, 1979.
50. G. Rodrigue and D. Wolitzer, *Incomplete block cyclic reduction*, 2nd IMACS International Symposium on Parallel Computation, Newark, Delaware, 1981.
51. O. Axelsson, *Error estimates over infinite intervals of some discretizations of evolution equations*, BIT 24 (1984), 413–424.
52. A. F. Cornock, *The numerical solution of Poisson's and the bi-harmonic equations by matrices*, Proc. Cambridge Phil. Soc. 50 (1954), 524–535.
53. S. Schechter, *Quasi-tridiagonal matrices and type-insensitive difference equations*, Quart. Appl. Math. 18 (1960), 285–295.
54. G. I. Marchuk, and V. P. Il'in, *Parallel computations in grid methods for solving mathematical*

physics problems, Information Processing 80, S. H. Lavington (ed.), North-Holland Publ. Comp. IFIP, 1980.

55. D. J. Evans, *New parallel algorithms in linear algebra*, EDF., Bulletin de la Direction des Etudes et des Recherches, Serie C, No. 1, Paris, 1983.
56. M. H. Schultz, *Solving elliptic problems on an array processor system*, Research Report YALEU/DCS/RR-272, June 1983.
57. T. J. Jordan, *A guide to parallel computation and some CRAY-1 experiences*, LANL Report, Los Alamos, 1981.
58. C. Kamath and A. Sameh, *The preconditioned conjugate gradient algorithm on a multiprocessor*, in *Advances in Computer Methods for Partial Differential Equations*, V. R. Vichnevetsky and R. S. Stepleman (eds), Publ. IMACS, Newark, Delaware, 1984.
59. D. R. Kincaid, T. C. Oppe and D. M. Young, *Vector computations for sparse linear systems*, Center for Numerical Analysis, CNA-189, The University of Texas at Austin, February 1984.