SIAM J. SCI. STAT. COMPUT.
Vol. 10, No. 1, pp. 36-52, January 1989

© 1989 Society for Industrial and Applied Mathematics
004

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

CGS, A FAST LANCZOS-TYPE SOLVER FOR NONSYMMETRIC LINEAR SYSTEMS*

PETER SONNEVELD†

Abstract. A Lanczos-type method is presented for nonsymmetric sparse linear systems as arising from discretisations of elliptic partial differential equations. The method is based on a polynomial variant of the conjugate gradients algorithm. Although related to the so-called bi-conjugate gradients (Bi-CG) algorithm, it does not involve adjoint matrix-vector multiplications, and the expected convergence rate is about twice that of the Bi-CG algorithm. Numerical comparison is made with other solvers, testing the method on a family of convection diffusion equations, on various grids, and with the use of two different preconditioning methods. Upwind as well as central differencing is used in the experiments.

Key words. preconditioned conjugate gradients, Lanczos algorithm, iterative methods, CG, CGS, SPD, Bi-CG

AMS(MOS) subject classifications. 65F10, 65N20

1. Introduction. For solving large sparse linear systems with a real symmetric positive definite coefficient matrix (referred to as an SPD-matrix), the Conjugate Gradients algorithm, when used with a suitable preconditioning technique, seems to be one of the most efficient methods in the class of simple iterative techniques. However, when the system is not SPD, as with discretisations of nonself-adjoint elliptic differential equations, the algorithm is not applicable. There are some generalisations available, such as Generalised Conjugate Gradients (Axelsson [1], [2]), Orthomin (Vinsome [23]), Orthodir (Young and Jea [26]), Manteuffel-Chebyshev (Manteuffel [12], [13], Van der Vorst [21]), Bi-Conjugate Gradients (Fletcher [8]), and Induced Dimension Reduction (Wesseling and Sonneveld, [24]). Some of these methods can be regarded as variants of a so-called Petrov-Galerkin approach (Saad and Schultz [18]). Also, hybrid combinations of CG generalisations and Chebyshev have been proposed (Elman, Saad, and Saylor [7]). All these generalisations use residuals lying in a Krylov subspace associated with the coefficient matrix of the system. Other generalisations use a Krylov subspace associated with the coefficient matrix of the normal equations (Paige and Saunders [17]), or use a Krylov subspace associated with related skew symmetric operator (Concus and Golub [4], Widlund [25]). Just as with application of ordinary CG on SPD problems, these methods are frequently combined with preconditioning techniques, such as incomplete factorisations (Meijerink and Van der Vorst [15], [16]), and we could say that without preconditioning these methods are hardly competitive.

The difficulty of solving nonsymmetric linear systems by means of CF-type methods can be understood as follows. In the CG algorithm, the residual vector is minimised with respect to some suitable vector norm. As a consequence, the residual vectors during the process are constructed such that they build a system orthogonal with respect to some related inner product. An essential property of the algorithm is the three-term recurrence relationship between the residuals, yielding an algorithm that is economical in both storage requirements and computing time. If operations with the coefficient matrix **A** are to be restricted to matrix vector multiplications, then, if **A** is real and not SPD, the only way for maintaining the three-term relations as well as the minimisation property involves the solution of the normal equations with the coefficient

* Received by the editors April 24, 1984; accepted for publication (in revised form) February 2, 1988.
† Delft University of Technology, Department of Mathematics, Delft, the Netherlands.

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

matrix $\mathbf{A}^T\mathbf{A}$. This can be done either by using the CG algorithm (see, e.g., Kershaw [11]), or, preferably, by applying Paige's bi-diagonalisation algorithm for the solution of sparse least-square problems (Paige and Saunders [17]). If the matrix $\mathbf{A}$ arises from a discretisation of a nonself-adjoint elliptic operator, the matrix $\mathbf{A}^T\mathbf{A}$ may have a much higher spectral condition number than the original matrix $\mathbf{A}$, causing a considerably slower convergence. There are several other possibilities for generalisations of the CG algorithm to nonsymmetric systems:

(1) We can maintain the minimisation property by choosing the next search direction as a linear combination of $\mathbf{r}_n$ and s previous search directions, such that the $\mathbf{A}$-image of the new search direction is orthogonal to the $\mathbf{A}$-images of the previous ones. Minimising the Euclidean norm of the next residual along this search direction yields the ORTHOMIN procedure. Other methods of this kind are ORTHODIR and Axelsson's method, *Generalised Conjugate Gradients.*

(2) We can maintain the three-term recursion by suitable re-interpretation of the CG algorithm. This is done in *Bi-Conjugate Gradients*, as well as in some sense in the *Induced Dimension Reduction* algorithm. In the general case nothing is minimised at all, which can be regarded as a theoretical weakness. In practice however, things turn out to be far less serious than we would expect, as numerical experiments have shown.

(3) We can maintain a three-term recursion, but abandon the entire CG idea. So we leave the class of Lanczos-type methods, and enter that of the Chebyshev-like methods. A disadvantage here is the necessity for finding optimal values for some scalar parameters (as when using SOR methods). So the method often is a slow starter, and therefore is slow compared with sprinters. However, if the spectrum of the coefficient matrix is in the right half of the complex plane (which is a common requirement for an iterative method to be successful anyway), the *Manteuffel-Chebyshev* algorithm with optimal parameters will certainly converge. The convergence rate depends on a "smallest ellipse" around the spectrum, and therefore cannot be influenced by using detailed information about the distribution of eigenvalues. This may be a second disadvantage.

(4) We can maintain a three-term recursion by splitting off the symmetric part of the matrix, if this part is an SPD-matrix, and applying some imaginary axis variant of the CG algorithm on a related skew symmetric system. This procedure requires the solution of an SPD problem at each iteration step [4], [25].

In this paper the second alternative is chosen. Via a polynomial equivalent of the CG algorithm, an algorithm for the squares of Lanczos polynomials is derived, which yields the *Conjugate Gradients Squared* algorithm (CGS). This method is found to be more efficient, for a relevant class of problems, than Bi-CG and ORTHOMIN, and, in our experience, competitive with the Manteuffel-Chebyshev algorithm.

2. The polynomial equivalent of the CG method. Let $\mathbf{A}$ be an $N \times N$ SPD-matrix, and let $\mathbf{Ax} = \mathbf{b}$ be a system to be solved. Let $\mathbf{x}_0$ be an initial guess for the solution $\mathbf{x}$; then the usual CG algorithm for this problem reads

$$
\begin{aligned}
&\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0;\ \mathbf{p}_{-1} := 0;\ \rho_{-1} := 1;\\
&n := 0;\\
&\textbf{while } \textit{residual} > \textit{tolerance} \textbf{ do}\\
&\textbf{begin}\\
&\quad \rho_n := \mathbf{r}_n^T\mathbf{r}_n;\ \beta_n := \frac{\rho_n}{\rho_{n-1}};\\
&\quad \mathbf{p}_n := \mathbf{r}_n + \beta_n\mathbf{p}_{n-1};\\
&\quad \sigma_n := \mathbf{p}_n^T\mathbf{A}\mathbf{p}_n;\ \alpha_n := \frac{\rho_n}{\sigma_n};
\end{aligned}
\tag{2.1}
$$

$\mathbf{r}_{n+1} := \mathbf{r}_n - \alpha_n \mathbf{A}\mathbf{p}_n;$
$\mathbf{x}_{n+1} := \mathbf{x}_n + \alpha_n \mathbf{p}_n;$
$n := n+1$
end;

It has been proved that this algorithm theoretically gives the exact answer after at most N steps. The vectors $\mathbf{r}_n$ satisfy $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$, and are called residuals. The vectors $\mathbf{p}_n$ are called search directions.

An important property of this algorithm is that the function $E_n = \mathbf{r}_n^T \mathbf{A}^{-1} \mathbf{r}_n$ is minimised at each step. This quantity can be regarded as the square of a norm for the residual $\mathbf{r}_n$. It has also been proved that the residuals are mutually orthogonal, and that the search directions are mutually $\mathbf{A}$-orthogonal:

$$\mathbf{r}_n^T \mathbf{r}_m = \rho_n \delta_{nm}, \qquad \mathbf{p}_n^T \mathbf{A}\mathbf{p}_m = \sigma_n \delta_{nm}$$

where δ_{nm} is the Kronecker symbol.

From the structure of the algorithm it can be seen that the vectors $\mathbf{r}_n$ and $\mathbf{p}_n$ can be written as follows:

$$\mathbf{r}_n = \varphi_n(\mathbf{A})\mathbf{r}_0, \qquad \mathbf{p}_n = \psi_n(\mathbf{A})\mathbf{r}_0$$

where φ_n and ψ_n are polynomials of degree less than or equal to n. In fact this can be proved simply by induction if we define: $\varphi_0(\tau) \equiv 1$, $\psi_{-1}(\tau) \equiv 0$, which is consistent with the CG algorithm. Substituting

$$\mathbf{r}_n = \varphi_n(\mathbf{A})\mathbf{r}_0, \qquad \mathbf{p}_{n-1} = \psi_{n-1}(\mathbf{A})\mathbf{r}_0 \tag{2.2}$$

for some $n \geqq 0$, we find

$$\mathbf{p}_n = \varphi_n(\mathbf{A})\mathbf{r}_0 + \beta_n \psi_{n-1}(\mathbf{A})\mathbf{r}_0 = \psi_n(\mathbf{A})\mathbf{r}_0$$

with

$$\psi_n(\tau) \equiv \varphi_n(\tau) + \beta_n \psi_{n-1}(\tau) \quad \forall \tau \in \mathbb{R}.$$

Substituting this in the computation step for $\mathbf{r}_{n+1}$ we find

$$\mathbf{r}_{n+1} = \varphi_n(\mathbf{A})\mathbf{r}_0 - \alpha_n \mathbf{A}\psi_n(\mathbf{A})\mathbf{r}_0 = \varphi_{n+1}(\mathbf{A})\mathbf{r}_0$$

with

$$\varphi_{n+1}(\tau) \equiv \varphi_n(\tau) - \alpha_n \tau \psi_n(\tau) \quad \forall \tau \in \mathbb{R}.$$

Hence the validity of hypothesis (2.2) for n implies the validity for $n+1$. So (2.1) can be re-interpreted as an algorithm for generating a system of (orthogonal) polynomials.

The computation of the scalar quantities ρ_n and σ_n can be carried out in terms of the polynomials as well. Denote the set of real polynomials of degree up to N by Π^N, and define the symmetric bilinear form $(\cdot, \cdot)$ on Π^N by

$$(\varphi, \psi) = [\varphi(\mathbf{A})\mathbf{r}_0]^T \psi(\mathbf{A})\mathbf{r}_0.$$

Obviously we have $(\varphi, \varphi) \geqq 0$, for all $\varphi \in \Pi^N$. Since $\mathbf{A}$ is symmetric, we also can write

$$(\varphi, \psi) = \mathbf{r}_0^T \varphi(\mathbf{A})\psi(\mathbf{A})\mathbf{r}_0.$$

Furthermore, from the associativity law for matrix products follows

$$(\varphi\chi, \psi) = (\varphi, \chi\psi)$$

for any polynomials φ, χ, ψ. So this bilinear form satisfies all but one of the requirements for a proper inner product on the space Π^N. Only positive definiteness is not guaranteed for all polynomials, since for some nonzero polynomial φ the vector $\varphi(\mathbf{A})\mathbf{r}_0$ may be

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

zero; this implies $(\varphi, \varphi) = 0$. However, as long as **A** is real symmetric, $(\cdot, \cdot)$ is semi-definite in any case.

Now the polynomial equivalent of (2.1) reads

$$
\begin{aligned}
&\varphi_0 \equiv 1;\ \psi_{-1} \equiv 0;\\
&\rho_{-1} := 1;\ n := 0;\\
&\textbf{while } \textit{residual} > \textit{tolerance} \textbf{ do}\\
&\textbf{begin}\\
&\quad \rho_n := (\varphi_n, \varphi_n);\ \beta_n := \frac{\rho_n}{\rho_{n-1}};\\
&\quad \psi_n := \varphi_n + \beta_n \psi_{n-1};\\
&\quad \sigma_n := (\psi_n, \vartheta\psi_n);\ \alpha_n := \frac{\rho_n}{\sigma_n};\\
&\quad \varphi_{n+1} := \varphi_n - \alpha_n \vartheta\psi_n;\\
&\quad n := n+1\\
&\textbf{end};
\end{aligned}
\tag{2.3}
$$

where

$$\vartheta(\tau) \equiv \tau. \tag{2.4}$$

The minimisation property now simply reads

$$E_n = (\varphi_n, \vartheta^{-1}\varphi_n) = \min_{\varphi \in \Pi^N} \frac{(\varphi, \vartheta^{-1}\varphi)}{\varphi(0)^2}.$$

(Note that all algorithms of the type (2.3), with freedom in the choice of α_n and β_n, share the property $\varphi_n(0) = 1$ for all n.)

The orthogonality of $\mathbf{r}_n$ and the **A**-orthogonality of $\mathbf{p}_n$ translate into orthogonality of the polynomials φ_n and ψ_n with respect to the inner products $(\cdot, \cdot)$ and $(\cdot, \vartheta\,\cdot)$ respectively.

The polynomial CG algorithm has a meaning in its own right, i.e., apart from its relation with the vectorial algorithm. We prove the following theorem.

THEOREM. *Let* $[\cdot, \cdot]$ *be any symmetric bilinear form defined on* Π^N, *satisfying*

$$[\varphi\psi, \chi] = [\varphi, \psi\chi] \quad \forall \varphi, \psi, \chi \in \Pi^N. \tag{2.5}$$

Let the sequence of polynomials φ_n *and* ψ_n *be constructed according to algorithm* (2.3), *but using* $[\cdot, \cdot]$ *instead of* $(\cdot, \cdot)$. *Then, as long as the algorithm does not break down by zero division, the polynomials* φ_n *and* ψ_n *satisfy*

$$[\varphi_n, \varphi_m] = \rho_n \delta_{nm}, \quad [\psi_n, \vartheta\psi_m] = \sigma_n \delta_{nm} \quad \forall n, m \tag{2.6}$$

with ϑ *defined by* (2.4).

Proof. We prove by induction the following statement:

$$[\psi_{n-1}, \vartheta\psi_k] = \sigma_{n-1}\delta_{n-1,k}, \quad [\varphi_n, \psi_k] = 0 \quad \forall n \geqq 0, \quad -1 \leqq k \leqq n-1 \tag{2.7}$$

with $\sigma_{-1} = 0$. If $n = 0$, this statement is true since $\psi_{-1}(\tau) \equiv 0$. Suppose (2.7) holds for $n \leqq m$, and let $k < m$; then according to (2.3) we have

$$[\psi_m, \vartheta\psi_k] = [\varphi_m, \vartheta\psi_k] + \beta_m [\psi_{m-1}, \vartheta\psi_k].$$

Now the second right-hand term vanishes for $k < m-1$ (by hypothesis). In the first term we substitute $\psi_k = (\varphi_k - \varphi_{k+1})/\alpha_k\vartheta$, yielding

$$[\psi_m, \vartheta\psi_k] = \frac{[\varphi_m, \varphi_k] - [\varphi_m, \varphi_{k+1}]}{\alpha_k} + \beta_m \sigma_{m-1}\delta_{m-1,k} \quad \forall k \leqq m-1.$$

By hypothesis this is zero if $k < m-1$, while for $k = m-1$ we have

$$[\psi_m, \vartheta\psi_{m-1}] = -\frac{\rho_m}{\alpha_{m-1}} + \beta_m\sigma_{m-1} = 0,$$

which proves the first part of (2.7) for $n = m+1$.

Proceeding with proving the second part, we can write

$$[\varphi_{m+1}, \psi_k] = [\varphi_m, \psi_k] - \alpha_m[\psi_m, \vartheta\psi_k] \quad \forall k \leqq m.$$

If $k \leqq m-1$ the right-hand side vanishes by hypothesis. Using the algorithm and choosing $k = m$ we get

$$[\varphi_{m+1}, \psi_m] = [\varphi_m, \varphi_m + \beta_m\psi_{m-1}] - \alpha_m[\psi_m, \vartheta\psi_m] = \rho_m - \alpha_m\sigma_m = 0,$$

which proves the second part of (2.7). Hence by induction (2.7) is valid for all natural n.

Finally, writing $\varphi_k = \psi_k - \beta_k\psi_{k-1}$, for all $k \geqq 0$ we have

$$[\varphi_n, \varphi_k] = [\varphi_n, \psi_k] - \beta_n[\varphi_n, \psi_{k-1}] = 0 \quad \forall k < n,$$

which together with the first part of (2.7) proves the theorem.

In general the polynomial algorithm could possibly break down by zero division in the computation of α_n and β_n. However, as long as the algorithm runs properly, the above theorem is valid. For this reason we shall use the term *orthogonal polynomials* for φ_n and ψ_n, whether or not the bilinear forms involved are inner products.

We are now in a position to generalise the CG algorithm to nonsymmetric cases, by defining $[\cdot, \cdot]$ in an appropriate way. Let $\mathbf{Ax} = \mathbf{b}$ be a nonsingular system to be solved. Let $\mathbf{x}_0$ be an initial guess for the solution $\mathbf{x}$, with corresponding residual $\mathbf{r}_0 = \mathbf{Ax}_0 - \mathbf{b}$, and let $\tilde{\mathbf{r}}_0$ be a suitably chosen vector. Now define $[\cdot, \cdot]$ by means of

$$[\varphi, \psi] = \tilde{\mathbf{r}}_0^T\varphi(\mathbf{A})\psi(\mathbf{A})\mathbf{r}_0 = (\varphi(\mathbf{A}^T)\tilde{\mathbf{r}}_0)^T\psi(\mathbf{A})\mathbf{r}_0 \tag{2.8}$$

and define the vectors $\mathbf{r}_n$, $\tilde{\mathbf{r}}_n$, $\mathbf{p}_n$, and $\tilde{\mathbf{p}}_n$ by

$$\begin{aligned} \mathbf{p}_{-1} &= \tilde{\mathbf{p}}_{-1} = \mathbf{0}, \\ \mathbf{r}_n &= \varphi_n(\mathbf{A})\mathbf{r}_0, \qquad \tilde{\mathbf{r}}_n = \varphi_n(\mathbf{A}^T)\tilde{\mathbf{r}}_0, \\ \mathbf{p}_n &= \psi_n((\mathbf{A})\mathbf{r}_0, \qquad \tilde{\mathbf{p}}_n = \psi_n(\mathbf{A}^T)\tilde{\mathbf{r}}_0 \end{aligned} \tag{2.9}$$

with φ_n and ψ_n according to (2.3). Then it can easily be shown that these vectors can be produced by the following so-called Bi-Conjugate Gradients algorithm (Bi-CG, Fletcher [8]):

$$\begin{aligned} &\mathbf{r}_0 := \mathbf{b} - \mathbf{Ax}_0;\ \mathbf{p}_{-1} := \tilde{\mathbf{p}}_{-1} := \mathbf{0}; \\ &n := 0;\ \rho_{-1} := 1; \\ &\textbf{while } \textit{residual} > \textit{tolerance} \textbf{ do} \\ &\textbf{begin} \\ &\quad \rho_n := \tilde{\mathbf{r}}_n^T\mathbf{r}_n;\ \beta_n := \frac{\rho_n}{\rho_{n-1}}; \\ &\quad \mathbf{p}_n := \mathbf{r}_n + \beta_n\mathbf{p}_{n-1};\ \tilde{\mathbf{p}}_n := \tilde{\mathbf{r}}_n + \beta_n\tilde{\mathbf{p}}_{n-1}; \\ &\quad \sigma_n := \tilde{\mathbf{p}}_n^T\mathbf{Ap}_n;\ \alpha_n := \frac{\rho_n}{\sigma_n}; \\ &\quad \mathbf{r}_{n+1} := \mathbf{r}_n - \alpha_n\mathbf{Ap}_n;\ \tilde{\mathbf{r}}_{n+1} := \tilde{\mathbf{r}}_n - \alpha_n\mathbf{A}^T\tilde{\mathbf{p}}_n; \\ &\quad \mathbf{x}_{n+1} := \mathbf{x}_n + \alpha_n\mathbf{p}_n; \\ &\quad n := n+1 \\ &\textbf{end}; \end{aligned} \tag{2.10}$$

Again (as long as no breakdown occurs) we have $\mathbf{r}_n = \mathbf{b} - \mathbf{Ax}_n$, meaning that convergence of $\mathbf{r}_n$ to $\mathbf{0}$ implies convergence of $\mathbf{x}_n$ to the solution $\mathbf{x}$. In practice the vector $\tilde{\mathbf{r}}_0$ is often

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

chosen equal to $\mathbf{r}_0$. Then, if $\mathbf{A}$ is not too far from being SPD, the bilinear expressions $[\cdot, \cdot]$ and $[\cdot, \vartheta \cdot]$ will be positive semi-definite, and the algorithm will converge in the same way, and by the same argument as does the ordinary CG algorithm in the SPD-case.

3. Squaring the CG algorithm: The CGS algorithm. Assume that the Bi-CG algorithm is converging well in some case. Then $\mathbf{r}_n \to 0$ as n increases, which can be interpreted as a property of the matrices $\varphi_n(\mathbf{A})$: since $\mathbf{r}_n = \varphi_n(\mathbf{A})\mathbf{r}_0$ these matrices behave like contracting operators, at least as far as their effect on the start residual is concerned. Then it may be expected that the matrices $\varphi_n(\mathbf{A}^T)$ are contracting too, with the consequence that also $\tilde{\mathbf{r}}_n \to \mathbf{0}$. So we may expect the two sequences of vectors $\mathbf{r}_n$ and $\tilde{\mathbf{r}}_n$ to converge at a comparable rate. However, the convergence of the "*quasi-residuals*" $\tilde{\mathbf{r}}_n$ is not exploited. But on the other hand, we need the computation of these vectors because they are involved in the computation of the scalar quantities ρ_n and σ_n. Now the amount of work needed for the computation of the vectors $\tilde{\mathbf{r}}_n$ and $\tilde{\mathbf{p}}_n$ is about the same as that for the original CG method itself, so we observe that the computational work of the Bi-CG algorithm is twice the computational work for the CG algorithm. Apart from this observation, another disadvantage is the occurrence of the operation $\mathbf{A}^T\mathbf{v}$, which may cause very cumbersome programming codes, especially if $\mathbf{A}$ is stored with a general data structure.

In order to improve the situation, we return to the polynomial algorithm (2.3). Here the scalars are calculated by $\rho_n = [\varphi_n, \varphi_n]$, $\sigma_n = [\psi_n, \vartheta\psi_n]$. Now since $[\cdot, \cdot]$ has the property (2.5) it follows that ρ_n and σ_n could be calculated just as well by the following formulae:

$$\rho_n = [\varphi_0, \varphi_n^2], \qquad \sigma_n = [\varphi_0, \vartheta\psi_n^2].$$

This suggests the possibility of an algorithm that generates the polynomials φ_n^2 and ψ_n^2 rather than φ_n and ψ_n. Consequently, we will try to interpret the vectors $\varphi_n^2(\mathbf{A})\mathbf{r}_0$ as residuals of estimates $\mathbf{x}_n$ of the solution $\mathbf{x}$. The computation of the quantities ρ_n for instance could be done by $\rho_n = \tilde{\mathbf{r}}_0^T\mathbf{r}_n$, where $\mathbf{r}_n$ now stands for $\varphi_n^2(\mathbf{A})\mathbf{r}_0$. Then a process for generating the sequences $\tilde{\mathbf{r}}_n$ and $\tilde{\mathbf{p}}_n$ is no longer needed, which is an important simplification of the programming code. Furthermore, the contracting effect of the matrices $\varphi_n(\mathbf{A})$ is "doubled."

To arrive at an algorithm for the squared polynomials we take the squares of the computational steps for ψ_n and φ_{n+1} in (2.3):

$$\psi_n^2 = \varphi_n^2 + 2\beta_n\varphi_n\psi_{n-1} + \beta_n^2\psi_{n-1}^2, \qquad \varphi_{n+1}^2 = \varphi_n^2 - 2\alpha_n\vartheta\varphi_n\psi_n + \alpha_n^2\vartheta^2\psi_n^2$$

where ϑ again is defined by (2.4). Obviously the computation of ψ_n^2 and φ_{n+1}^2 requires the products $\varphi_n\psi_{n-1}$ and $\varphi_n\psi_n$ as well. However, $\varphi_n\psi_n$ can be expressed as $\varphi_n^2 + \beta_n\varphi_n\psi_{n-1}$, and is therefore not explicitly needed. Furthermore, multiplication of the computational step for φ_{n+1} with ψ_n yields

$$\varphi_{n+1}\psi_n = \varphi_n\psi_h - \alpha_n\vartheta\psi_n^2,$$

which also is expressible in φ_n^2, $\varphi_n\psi_{n-1}$ and ψ_{n-1}^2. Hence an algorithm with the required properties can be constructed. Define for $n \geqq 0$:

$$\begin{aligned} \Phi_n &= \varphi_n^2, \\ \Theta_n &= \varphi_n\psi_{n-1}, \\ \Psi_{n-1} &= \psi_{n-1}^2. \end{aligned} \tag{3.1}$$

Then it is easily verified that the following algorithm has the desired properties:

$$
\begin{aligned}
&\Phi_0 \equiv 1;\ \Theta_0 \equiv \Psi_{-1} \equiv 0;\ \rho_{-1} := 1;\\
&n := 0;\\
&\textbf{while } \textit{residual} > \textit{tolerance} \textbf{ do}\\
&\textbf{begin}\\
&\quad \rho_n := [1, \Phi_n];\ \beta_n := \frac{\rho_n}{\rho_{n-1}};\\
&\quad \Upsilon_n := \Phi_n + \beta_n \Theta_n;\\
&\quad \Psi_n := \Upsilon_n + \beta_n(\Theta_n + \beta_n \Psi_{n-1});\\
&\quad \sigma_n := [1, \vartheta \Psi_n];\ \alpha_n := \frac{\rho_n}{\sigma_n};\\
&\quad \Theta_{n+1} := \Upsilon_n - \alpha_n \vartheta \Psi_n;\\
&\quad \Phi_{n+1} := \Phi_n - \alpha_n \vartheta(\Upsilon_n + \Theta_{n+1});\\
&\quad n := n+1\\
&\textbf{end};
\end{aligned}
\tag{3.2}
$$

We obtain a vectorial variant of this algorithm by substituting $\mathbf{A}$ into the polynomials, and making the appropriate parts act as linear operators on the starting residual $\mathbf{r}_0$. Defining $\mathbf{r}_n = \Phi_n(\mathbf{A})\mathbf{r}_0$, $\mathbf{q}_n = \Theta_n(\mathbf{A})\mathbf{r}_0$, and $\mathbf{p}_n = \Psi_n(\mathbf{A})\mathbf{r}_0$ for all appropriate values of n, in this way we arrive at the Conjugate Gradients Squared (CGS) algorithm.

Let $\mathbf{x}_0$ be a starting estimate of the solution $\mathbf{x}$ of the nonsingular linear system $\mathbf{Ax} = \mathbf{b}$, and let $\tilde{\mathbf{r}}_0$ be suitably chosen. Then the CGS algorithm reads:

$$
\begin{aligned}
&\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0;\\
&\mathbf{q}_0 := \mathbf{p}_{-1} := \mathbf{0};\ \rho_{-1} := 1;\\
&n := 0;\\
&\textbf{while } \textit{residual} > \textit{tolerance} \textbf{ do}\\
&\textbf{begin}\\
&\quad \rho_n := \tilde{\mathbf{r}}_0^T \mathbf{r}_n;\ \beta_n := \frac{\rho_n}{\rho_{n-1}};\\
&\quad \mathbf{u}_n := \mathbf{r}_n + \beta_n \mathbf{q}_n;\\
&\quad \mathbf{p}_n := \mathbf{u}_n + \beta_n(\mathbf{q}_n + \beta_n \mathbf{p}_{n-1});\\
&\quad \mathbf{v}_n := \mathbf{A}\mathbf{p}_n;\\
&\quad \sigma_n := \tilde{\mathbf{r}}_0^T \mathbf{v}_n;\ \alpha_n := \frac{\rho_n}{\sigma_n};\\
&\quad \mathbf{q}_{n+1} := \mathbf{u}_n - \alpha_n \mathbf{v}_n;\\
&\quad \mathbf{r}_{n+1} := \mathbf{r}_n - \alpha_n \mathbf{A}(\mathbf{u}_n + \mathbf{q}_{n+1});\\
&\quad \mathbf{x}_{n+1} := \mathbf{x}_n + \alpha_n(\mathbf{u}_n + \mathbf{q}_{n+1});\\
&\quad n := n+1\\
&\textbf{end};
\end{aligned}
\tag{3.3}
$$

Since $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{r}_{n+1} - \mathbf{r}_n = \mathbf{A}(\mathbf{x}_n - \mathbf{x}_{n+1})$ for all n it follows that $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$. So this algorithm produces iterates $\mathbf{x}_n$ of which the residuals satisfy

$$\mathbf{r}_n = \varphi_n^2(\mathbf{A})\mathbf{r}_0.$$

Each step requires twice the amount of work necessary for the symmetric CG method, however, the contracting effect of $\varphi_n(\mathbf{A})$ is used twice each step. The work is not more than for Bi-CG, and working with $\mathbf{A}^T$ is avoided.

4. Convergence and preconditioning. For the CG method, when applied to self-adjoint positive definite problems, the following theoretical result holds for the

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

residuals:

$$0 < \frac{E_n}{E_0} \leqq 4\left[\frac{\sqrt{K_A}-1}{\sqrt{K_A}+1}\right]^{2n} \tag{4.1}$$

with

$$E_n = \mathbf{r}_n^T \mathbf{A}^{-1} \mathbf{r}_n$$

where K_A denotes the spectral condition number of $\mathbf{A}$, defined by $K_A = \lambda_{\max}/\lambda_{\min}$. Since E_n can be regarded as the square of a norm for $\mathbf{r}_n$, this formula means that CG converges at least as fast as a linearly convergent process with convergence factor $\rho = (1-2/\sqrt{K_A})$. For the proof see, for instance, Axelsson [3]. More refined theoretical results are presented in Van der Vorst [22], which takes into account more information about the distribution of eigenvalues of $\mathbf{A}$. The convergence of the CG algorithm turns out to be superlinear, with a convergence rate that depends on the distribution of the (mainly smallest) eigenvalues of $\mathbf{A}$, rather than on the spectral condition number.

It is common to apply *preconditioning techniques* in order to reduce the spectral condition number (Gustafsson [9]), or to improve the distribution of the smallest eigenvalues (Van der Vorst [22]). Although Bi-CG and CGS algorithms do not require an SPD-matrix $\mathbf{A}$, it may be expected that, if the bilinear form $[\cdot, \cdot]$ does not deviate too much from a proper inner product (i.e., from being positive definite), the convergence of both methods will be largely similar to the SPD case. It can easily be proved that in absence of roundoff both Bi-CG and CGS terminate after at most N steps by zero division, since ρ_n will be zero for some $n \leqq N$. If the vector $\tilde{\mathbf{r}}_0$ is not orthogonal to any invariant subspace of the Krylov subspace associated with $\mathbf{A}$ and $\mathbf{r}_0$ (which is true in "almost all" cases with random choice for $\tilde{\mathbf{r}}_0$), this implies $\mathbf{r}_n = \mathbf{0}$; hence both processes are theoretically finite in the same way as CG is in the SPD case. An essential requirement for convergence in considerably less than N steps is that *no eigenvalues of the matrix* $\mathbf{A}$ *are in the left half of the complex plane*, i.e., that the matrix $\mathbf{I} - \alpha\mathbf{A}$ is a contraction for some positive α. In fact it can be proved that if $\mathbf{A}$ is not defective and has positive real eigenvalues, a vector $\tilde{\mathbf{r}}_0$ exists such that both Bi-CG and CGS converge at a rate comparable to that of CG (however, it is not easy to find this "suitable" choice for $\tilde{\mathbf{r}}_0$ in practice). We may also expect (and this has been verified numerically) that preconditioning will improve the behaviour of the Bi-CG and CGS algorithms, in the same way as with CG.

A global description of preconditioning will now be given. Let $\mathbf{P}_L$ and $\mathbf{P}_R$ be two linear operators mapping $\mathbb{R}^N$ onto itself, being "*sparse*" in the following sense: for arbitrary $\mathbf{x} \in \mathbb{R}^N$ the products $\mathbf{P}_L\mathbf{x}$ and $\mathbf{P}_R\mathbf{x}$ require about the same number of operations as would be required if $\mathbf{P}_L$ and $\mathbf{P}_R$ were sparse *matrices*. In practice $\mathbf{P}_L$ and $\mathbf{P}_R$ are often chosen to be inverses of sparse, triangular matrices; in that case the action of these operators can be carried out as back-substitutions, which require the same amount of work as is needed for a sparse matrix-vector multiplication.

Define $\tilde{\mathbf{x}}$ by $\mathbf{P}_R\tilde{\mathbf{x}} = \mathbf{x}$ and $\tilde{\mathbf{b}}$ by $\tilde{\mathbf{b}} = \mathbf{P}_L\mathbf{b}$; then the preconditioned version of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ reads:

$$\mathbf{B}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{4.2}$$

where $\mathbf{B} = P_L\mathbf{A}\mathbf{P}_R$ is the preconditioned matrix. The operators $\mathbf{P}_L$ and $\mathbf{P}_R$ are chosen such that $\mathbf{B}$ has better spectral properties with respect to the CG algorithm. We can regard the product $\mathbf{P}_R\mathbf{P}_L$ as an approximate inverse of $\mathbf{A}$ in some sense.

There are several possibilities for implementing the preconditioning matrices in the CGS algorithm. The variant used in our numerical experiments reads as follows:

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

$\mathbf{r}_0 := \mathbf{P}_L(\mathbf{b} - \mathbf{A}\mathbf{x}_0)$; $\tilde{\mathbf{r}}_0 := \mathbf{r}_0$;
$\mathbf{q}_0 := \mathbf{p}_{-1} := \mathbf{0}$;
$\rho_{-1} := 1$; $n := 0$;
while *residual* $>$ *tolerance* **do**
begin

$\rho_n := \tilde{\mathbf{r}}_0^T \mathbf{r}_n$; $\beta_n := \dfrac{\rho_n}{\rho_{n-1}}$;
$\mathbf{u}_n := \mathbf{r}_n + \beta_n \mathbf{q}_n$;
$\mathbf{p}_n := \mathbf{u}_n + \beta_n(\mathbf{q}_n + \beta_n \mathbf{p}_{n-1})$;
$\mathbf{v}_n := \mathbf{P}_L \mathbf{A} \mathbf{P}_R \mathbf{p}_n$;
$\sigma := \tilde{\mathbf{r}}_0^T \mathbf{u}_n$; $\alpha_n := \dfrac{\rho_n}{\sigma_n}$;
$\mathbf{q}_{n+1} := \mathbf{u}_n - \alpha_n \mathbf{v}_n$;
$\mathbf{v}_n := \alpha_n \mathbf{P}_R(\mathbf{u}_n + \mathbf{q}_{n+1})$;
$\mathbf{x}_{n+1} := \mathbf{x}_n + \mathbf{v}_n$;
$\mathbf{r}_{n+1} := \mathbf{r}_n - P_L \mathbf{A} \mathbf{v}_n$;
$n := n + 1$
end;

(4.3)

In this algorithm, $\mathbf{x}_n$ approximates the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ (not $\tilde{\mathbf{x}}$ in (4.2)).

An important class of preconditioning methods is based on *incomplete factorisations* of the matrix **A**. An incomplete factorisation of **A** consists of a lower triangular matrix **L** and an upper triangular matrix **U** with a prescribed (sparse) nonzero pattern and satisfying

$$\mathbf{L}\mathbf{U} = \mathbf{A} + \mathbf{E}.$$

The deviation matrix **E** is often very sparse itself, but this is not necessary in general (as in the case of *incomplete line LU factorisation*). The incomplete *LU factorisation* used most frequently can be obtained by a slightly modified Gaussian elimination procedure. Let $NZ(\mathbf{A})$ denote the set of pairs $[i, j]$ for which the entries a_{ij} of the matrix **A** are nonzero, the *nonzero pattern of* **A**. Then the incomplete Gauss algorithm reads:

for $k := 1$ **step** 1 **until** N **do**
begin
  for $i := k+1$ **step** 1 **until** N **do**
  begin
    $a_{ik} := \dfrac{a_{ik}}{a_{kk}}$;
    for $j := k+1$ **step** 1 **until** N **do**
    begin
      if $[i, j] \in NZ(\mathbf{A})$ **then**
      begin
        $corr := -a_{ik} a_{kj}$;
        $\alpha_{ij} := a_{ij} + corr$
      end
    end
  end
end;

(4.4)

In the case of a compactly stored matrix **A**, the integers i and j do not get all values in the domain of the loops; only values of i for which $(i, k) \in NZ(\mathbf{A})$ and values of j

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

for which $(k, j) \in NZ(\mathbf{A})$ are necessary. Coding this is the most difficult part of the incomplete Gauss algorithm.

The upper triangular part of the resulting matrix $\mathbf{A}$ is now to be understood as the matrix U, the strictly lower triangular part together with the unit diagonal matrix as the matrix L in the factorisation. We denote the preconditioning based on the above algorithm by ILU.

Other possibilities will not be shown in detail in this paper. We mention Gustaffson's variant, presented in [9], *modified incomplete LU factorisation*, denoted by MILU, in which variant the quantity *corr* in the algorithm is added to the main diagonal of the matrix whenever (i, j) is not in the nonzero pattern. Generalisations can also be obtained by choosing other sets of pairs (i, j) instead of $NZ(\mathbf{A})$ (Meijerink and Van der Vorst [15], [16]).

With the matrices $\mathbf{L}$ and $\mathbf{U}$ as obtained with the above algorithm, we can define preconditionings as follows:

$$\begin{aligned} &\mathbf{P}_L = \mathbf{L}^{-1}, \qquad \mathbf{P}_R = \mathbf{U}^{-1}, \\ &\mathbf{P}_L = \mathbf{U}^{-1}\mathbf{L}^{-1}, \qquad P_R = \mathbf{I}, \\ &\mathbf{P}_L = \mathbf{I}, \qquad \mathbf{P}_R = \mathbf{U}^{-1}\mathbf{L}^{-1}. \end{aligned}$$

Each of these alternatives yields a matrix $\mathbf{B}$ with the same spectrum. The choice between them depends on arguments concerning the implementation. The existence and the numerical stability of ILU factorisations have been proved for cases where $\mathbf{A}$ is an M-matrix. In practice this is not a severe restriction. Van der Vorst in [21] and [22] has shown that in some important cases where $\mathbf{A}$ is not an M-matrix, a suitable preconditioning is obtained by applying the ILU algorithm to the matrix $\mathbf{A}+\mathbf{D}$ where $\mathbf{D}$ is some positive diagonal matrix.

Another type of incomplete factorisation, used as preconditioning, can be applied if $\mathbf{A}$ is a block-tridiagonal matrix. This is often the case if $\mathbf{A}$ arises from discretisations of second-order elliptic partial differential equations. It is called *incomplete line LU factorisation*, and denoted by ILLU. Let $\mathbf{L}$ and $\mathbf{U}$ be the strictly *block-lower* and strictly *block-upper* parts of $\mathbf{A}$ and let $\mathbf{D}_\mathbf{A}$ be the *block-diagonal* part. A block-diagonal matrix $\mathbf{D}$ is constructed, with the same nonzero pattern as $\mathbf{D}_\mathbf{A}$, satisfying

$$\mathbf{A} = (\mathbf{L}+\mathbf{D})\mathbf{D}^{-1}(\mathbf{U}+\mathbf{D})+\mathbf{E}$$

where $\mathbf{E}$ is a (block-diagonal) matrix, having a nonzero pattern which is complementary to that of $\mathbf{D}_\mathbf{A}$, thus having a zero tridiagonal part. Details are described in [5], [14], and [19].

The ILLU factorisation has the advantage of apparently being less sensitive to special directions in the problem, as, for instance, the convection direction in the convection-diffusion equation.

Preconditioning with ILLU is done by choosing (for instance):

$$\mathbf{P}_L = (\mathbf{L}+\mathbf{D})^{-1}, \qquad \mathbf{P}_R = (\mathbf{U}+\mathbf{D})^{-1}\mathbf{D}.$$

Numerical experiments in this paper were carried out using ILU as well as ILLU preconditioning.

5. Numerical experiments. In this section the applicability of the CGS algorithm is demonstrated for some test examples belonging to the class of convection diffusion equations. In examples (1) and (2), the behaviour of the preconditioned CGS is compared with the author's implementation of the Bi-CG and the ORTHOMIN

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

algorithms. Examples (3) and (4) are taken from Van der Vorst [21], for a comparison of CGS with the Manteuffel variant of the CHEBYSHEV algorithm.

5.1. Operation counts and storage requirements. If the algorithms used in this section are implemented carefully in order to avoid double computations of the same quantities, the computational complexities are:

$$
\begin{aligned}
W_{\text{Bi-CG}} &= W_{\text{PREC}} + 1 + W_{\mathbf{P}_L} + W_{\mathbf{A}} + \text{steps} \times [15 + 2 \times (W_{\mathbf{P}_L} + W_{\mathbf{A}} + W_{\mathbf{P}_R})], \\
W_{\text{ORTHOMIN}} &= W_{\text{PREC}} + 1 + W_{\mathbf{A}} + \text{steps} \times [40 + W_{\mathbf{P}_L} + W_{\mathbf{A}} + W_{\mathbf{P}_R}], \\
W_{\text{CGS}} &= W_{\text{PREC}} + 1 + W_{\mathbf{P}_L} + W_{\mathbf{A}} + \text{steps} \times [18 + 2 \times (W_{\mathbf{P}_L} + W_{\mathbf{A}} + W_{\mathbf{P}_R})].
\end{aligned} \tag{5.1}
$$

The quantity W_{PREC} denotes the computational work for constructing the preconditioning operators $\mathbf{P}_L$ and $\mathbf{P}_R$. The quantities $W_{\mathbf{P}_L}$, $W_{\mathbf{A}}$, and $W_{\mathbf{P}_R}$ denote the computational work required for the evaluation of $\mathbf{P}_L\mathbf{x}$, $\mathbf{Ax}$, and $\mathbf{P}_R\mathbf{x}$, respectively. All quantities *work* are measured in *floating point operations* (*flops*) *per degree of freedom.* The quantity *steps* denotes the number of iteration steps. It must be noted that the ORTHOMIN procedure requires a different amount of work during the first five steps. The precise operation count reads as follows:

$$
W_{\text{ORTHOMIN}}(\text{step}_k) = 4 + 6k + W_{\mathbf{P}_L} + W_{\mathbf{A}} + W_{\mathbf{P}_R}. \tag{5.2}
$$

In the *work* an extra inner product for checking the norm reduction of the residuals is included. In ORTHOMIN the Euclidean norm of the (recursively obtained) true residual is measured each step, whereas in the other two algorithms the preconditioned residuals ($\mathbf{P}_L(\mathbf{Ax}_n - \mathbf{b})$) are considered. After termination of the process the true residual reduction was compared with the measured quantities; in all cases the differences were not significant.

Apart from storage requirements for storing the operators $\mathbf{P}_L$, $\mathbf{P}_R$, and $\mathbf{A}$, and the vectors $\mathbf{x}$ and $\mathbf{b}$, the number S of N-vectors used by the three algorithms is listed below:

$$
S_{\text{Bi-CG}} = 6, \quad S_{\text{ORTHOMIN}} = 13, \quad S_{\text{CGS}} = 6. \tag{5.3}
$$

The storage requirements for the preconditioning operators as well as the amounts of work W_{PREC}, $W_{\mathbf{P}_L}$, $W_{\mathbf{A}}$, and $W_{\mathbf{P}_R}$ are strongly dependent on the implementation. Dependent on the type of grid (5-point- or 7-point finite difference star, or a, possibly irregular, finite-element grid) several saving devices for computing time have been developed (see, for instance, Eisenstat [6]). We consider the possibility of computing the product $\mathbf{P}_L\mathbf{AP}_R\mathbf{x}$ by $\mathbf{x} - \mathbf{P}_L\mathbf{EP}_R\mathbf{x}$, in the case of ILU preconditioning, saving operations as well as storage (a similar trick exists for the ILLU preconditioning). In Table 1 these reduced amounts of work are given between brackets.

In the general case of a finite-element discretisation of an elliptic PDE the cheap variants are not always applicable, so, except in the comparison with the CHEBYSHEV

TABLE 1
Computational work and storage.

DISCR	PREC	W_{PREC}	$W_{\mathbf{P}_L}$	$W_{\mathbf{A}}$	$W_{\mathbf{P}_R}$	$W_{\mathbf{P}_L\mathbf{AP}_R}$	STORAGE
5-pts	ILU	10	4	9 (13)	5	18 (13)	5 (3)
5-pts	ILLU	19	7	9 (9)	12	28 (26)	3 (3)
7-pts	ILU	21	6	13 (17)	7	26 (17)	7 (3)

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

algorithm, we use the straightforward amounts of work throughout. This acts slightly in favour of the operation count for ORTHOMIN, as can be understood from (5.1).

In the tables of results the total work per degree of freedom is based on (5.1) and (5.2), using Table 1.

5.2. Test examples.

PROBLEM 1. Convection Diffusion equation on the unit square $\Omega = (0, 1) \times (0, 1)$.

$$-\varepsilon\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\cos(\alpha)\frac{\partial u}{\partial x}+\sin(\alpha)\frac{\partial u}{\partial y}=0, \qquad u(x, y)=x^2+y^2 \quad \text{and } \partial\Omega.$$

This equation is used for comparison of the Bi-CG, ORTHOMIN, and CGS algorithms. A 5-point hybrid difference scheme is used on several square grids. That is, upwind differencing is done only if necessary for maintaining diagonal dominance; otherwise central differencing is applied. The dimension of the algebraic system of equations N corresponds to the number of interior points of the grid

$$N=\left[\frac{1}{\Delta x}-1\right]^2$$

with $\Delta x = \Delta y$.

PROBLEM 2. Berkeley problem on $\Omega = (-1, +1) \times (0, 1)$.

$$-\varepsilon\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+v_x\frac{\partial u}{\partial x}+v_y\frac{\partial u}{\partial y}=0$$

with

$$v_x = 2y(1-x^2), \qquad v_y = -2x(1-y^2),$$

$$u(x, y)=0, \quad \begin{array}{ll} x=-1, & 0 \leqq y \leqq 1, \\ x=+1, & 0 \leqq y \leqq 1, \\ y=+1, & -1 \leqq x \leqq 1, \end{array}$$

$$u(x, 0) = 1+\tanh(10(2x+1)), \qquad y=0, \quad -1 \leqq x \leqq 0,$$

$$\frac{\partial u}{\partial n}=0, \qquad y=0, \quad 0 \leqq x \leqq 1.$$

This problem, described in [10] and [20], is discretised on a rectangular grid, using 5-point finite differences, the boundary $\partial\Omega$ chosen between the appropriate exterior two gridlines (virtual boundary). Central as well as hybrid differencing is applied. Since the tests on Problem 1 show that Bi-CG is far less efficient if compared with ORTHOMIN and CGS, we have compared only the latter two methods.

In both Problem 1 and Problem 2 the starting vector is chosen uniformly quasi-randomly distributed between -2 and 2.

PROBLEM 3. Van der Vorst problem (5.1) on $\Omega = (0, 30) \times (0, 30)$.

$$-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}+\beta\left(\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}\right)+u=1$$

with

$$u(x, y) \equiv 1 \quad \text{on } \partial\Omega.$$

This equation, taken from [21], is discretised on a square grid with $\Delta x = \Delta y = 1$. As starting vector the null-vector is chosen.

PROBLEM 4. Van der Vorst problem (5.4) on $\Omega = (0, 1) \times (0, 1)$.

$$-\frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial x} + (1+y^2)\left(-\frac{\partial^2 u}{\partial y^2} + \frac{\partial u}{\partial y}\right) = f(x, y)$$

with Dirichlet type boundary condition and right-hand side such that

$$u(x, y) = e^{x+y} + x^2(1-x)^2 \ln(1+y^2).$$

This equation, also taken from [21], is discretised on square grids with $\Delta x = \Delta y = 1/16$ and $\Delta x = \Delta y = 1/30$, respectively. Again the null vector is chosen as the starting vector.

5.3. Numerical results. Table 2 shows the behaviour of Bi-CG, ORTHOMIN, and CGS on Problem 1, for values of ε, α, and N as listed. Five orthogonal vectors are used in the ORTHOMIN procedure. The two versions of the ILU preconditioning are different in the choice of the nonzero pattern. In the case of ILU (7) the nonzero

TABLE 2
Problem 1 with $\alpha = -30$ deg.

	Average work per digit, required to obtain a reduction $< 10^{-12}$.							
			Reduction, iterations			Work per digit		
ε	N	Prec.	B-CG	ORTH	CGS	Bi-CG	ORTH	CGS
10^{-1}	100	ILU(5)	13 (18)	11 (20)	13 (11)	75	100	48
		ILU(7)	13 (12)	13 (13)	15 (8)	66	62	40
		ILLU	12 (9)	13 (10)	13 (5)	55	47	32
	400	ILU(5)	6 (20)	6 (20)	14 (20)	172	192	81
		ILU(7)	12 (20)	10 (20)	12 (12)	114	125	71
		ILLU	12 (15)	13 (17)	13 (10)	91	86	58
	1600	ILU(5)	3 (20)	4 (20)	5 (20)	336	290	203
		ILU(7)	5 (20)	5 (20)	9 (20)	289	239	153
		ILLU	9 (20)	9 (20)	14 (17)	171	152	95
	6400	ILLU	9 (40)	9 (40)	13 (32)	304	305	189
10^{-4}	100	ILU(5)	12 (11)	12 (20)	15 (8)	48	91	31
		ILU(7)	13 (8)	12 (15)	23 (7)	44	77	24
		ILLU	15 (4)	15 (4)	22 (3)	21	13	12
	400	ILU(5)	11 (20)	5 (20)	12 (13)	92	206	60
		ILU(7)	13 (12)	11 (20)	12 (8)	63	115	45
		ILLU	15 (5)	15 (5)	16 (3)	26	18	16
	1600	ILU(5)	4 (20)	3 (20)	8 (20)	268	380	131
		ILU(7)	7 (20)	2 (20)	15 (18)	189	561	89
		ILLU	14 (6)	15 (6)	12 (3)	34	24	21
	6400	ILLU	14 (8)	13 (7)	12 (4)	45	33	27

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

patterns of **L** and **U** correspond to a 7-point finite-difference star, whereas in the case of ILU (5), these patterns correspond to the ordinary 5-point star. In all cases the processes are terminated either at a residual reduction by a factor of 10^{-12}, or after 20 steps, except for the tests on an 80×80 grid, where a maximum of 40 iterations is admitted. Although the convergence behaviour of both Bi-CG and CGS is not linear, the efficiency of the methods can be measured very well by the mean work per degree of freedom necessary for gaining one decimal digit (corresponding to a reduction by a factor 10^{-1}). This quantity, denoted by *work per digit*, is defined as total work/$-\log_{10}$(reduction) and measured in flops; the quantity *reduction* is measured in decimal digits; the numbers between parentheses denote the numbers of iteration steps actually used.

Comparing the efficiency of Bi-CG and CGS, we see that the latter method requires about 60 percent of the work required by the former method. This is in reasonable agreement with the theoretically expected 50 percent. A reason for the deviation might be the superlinear convergence of these algorithms. Comparing the three methods we see that, especially in the "slower" cases, CGS is significantly more efficient than the other two methods.

Furthermore the superiority of ILLU preconditioning is shown (at least for this problem). The chosen value of α (-30 degrees) belongs to the class of medium difficult angles for this problem. In [19], an extensive comparison of ILU and ILLU preconditionings is made for rather many different values of α. In the case of the 80×80 grid, only the ILLU preconditioning is used, since the other preconditioners yield rather poor results for the three methods.

Finally, it is interesting to observe that ORTHOMIN suffers more from poor preconditioning (as ILU sometimes is) than does CGS.

In Fig. 1, the convergence characteristics of the methods are plotted for a few cases. The ORTHOMIN curves show the most linear convergence. Since the first steps of ORTHOMIN are considerably cheaper than the later ones, ORTHOMIN may be regarded as an efficient choice for results of low accuracy.

In some cases the residuals produced by Bi-CG and CGS make quite large jumps in the "wrong" direction. This has to do with the nonpositive definiteness of the "inner products" used. Since this phenomenon may be responsible for a severe loss of significant digits, it is necessary, in the author's opinion, to develop an escape from this problem.

Table 3 shows the behaviour of ORTHOMIN and CGS, with ILLU preconditioning, in solving Problem 2. The equation is discretised on a 100×50 grid. As may be expected, when using central differences, for small ε the matrix loses too much diagonal dominance, and the iterations do not converge. In the case $\varepsilon = 10^{-3}$, the convergence of ORTHOMIN has to be considered as spurious: the residual norm did not change after the seventh step. Apparently CGS is less sensitive to loss of diagonal dominance. For $\varepsilon \leqq 10^{-3.5}$ both methods diverge, however, ORTHOMIN does not show this as clearly as CGS does.

In Table 4 CGS is compared with Manteuffel's version of the CHEBYSHEV algorithm, with adaptive parameter estimation. The results for the latter algorithm were presented by Van der Vorst in [21]. Since we do not know the exact operation count for the adaptive part of the algorithm, only estimates of the quantity *work* are given. We do not take initial work into account for both methods. As preconditioning we used the ILU factorisation in the so-called *Eisenstat* implementation [6], [21]. In this implementation, $W_{\mathbf{P}_L\mathbf{A}\mathbf{P}_R}$ is 11 flops. The work for the CHEBYSHEV algorithm, including the Euclidean norm calculation, is 7 flops. The total work per iteration step

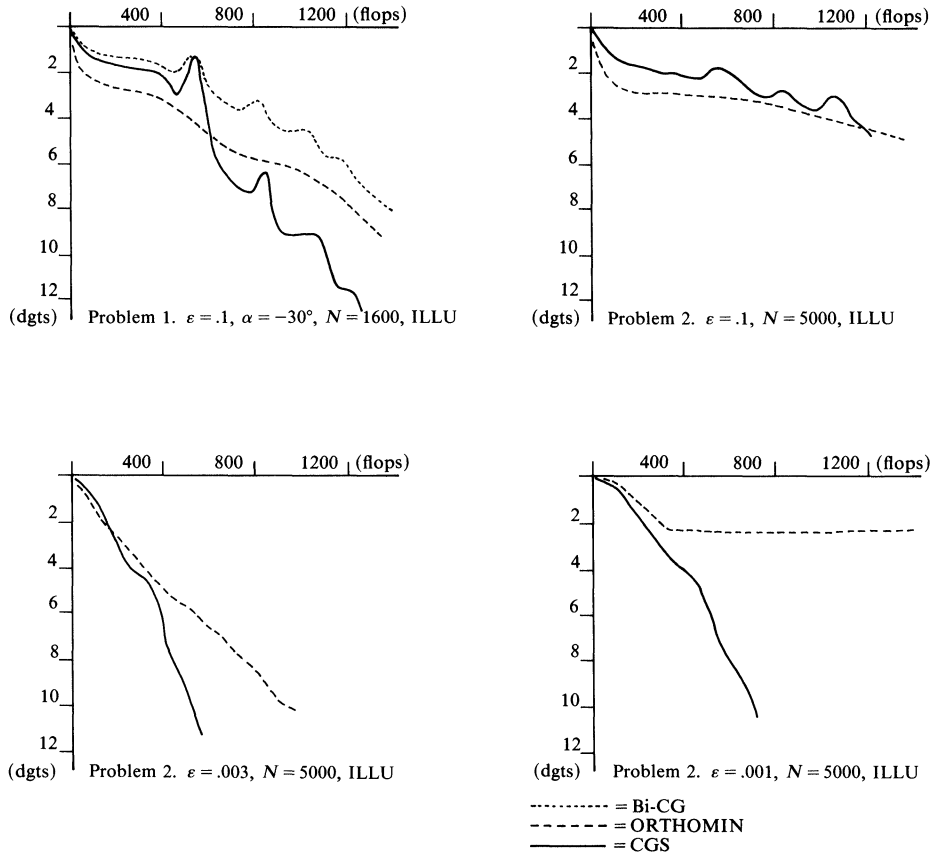

FIG. 1. *Convergence characteristics.*

TABLE 3
Problem 2 on a 100×50 *grid.*

Differences	ε	Reduction, iterations		Work per digit	
		ORTH	CGS	ORTH	CGS
Central	10^{-1}	4.8 (20)	4.3 (15)	273	270
	$10^{-1.5}$	4.5 (20)	4.9 (15)	267	236
	10^{-2}	10.4 (20)	11.7 (11)	125	73
	$10^{-2.5}$	10.2 (15)	11.2 (7)	94	49
	10^{-3}	2.4 (20)	10.7 (9)	538	66
Hybrid	10^{-2}	9.9 (20)	11.1 (11)	132	77
	$10^{-2.5}$	10.9 (13)	11.1 (7)	76	50
	10^{-3}	10.7 (8)	12.3 (5)	45	33
	$10^{-3.5}$	10.4 (6)	13.6 (4)	33	24
	10^{-4}	11.1 (5)	12.8 (3)	25	20

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

TABLE 4
Problems 3 and 4.

Problem	Reduction, iterations		Work per digit	
	CHEB	CGS	CHEB	CGS
3, $\beta = 4.0$	$9.0\,10^{-9}$ (25)	$4.5\,10^{-10}$ (9)	56	39
3, $\beta = 20.0$	$8.3\,10^{-9}$ (37)	$8.1\,10^{-9}$ (16)	82	79
4, $N = 15\,15$	$2.0\,10^{-8}$ (38)	$4.3\,10^{-9}$ (12)	89	57
4, $N = 29\,29$	$9.1\,10^{-9}$ (45)	$2.6\,10^{-9}$ (22)	100	102

is therefore 18 flops for the CHEBYSHEV algorithm, and 40 flops for the CGS algorithm.

5.4. Conclusions. For large, sparse, nonsymmetric linear systems, we may regard the combination of CGS with ILLU preconditioning as a competitive solver, at least for problems that are not too large, and when high accuracies are required. From Table 2 we may conclude that the average work for solving convection-diffusion equations with medium ε in two dimensions is roughly proportional to $N\sqrt{N}$. Although this is comparable with the SOR method, the proportionality coefficient is much smaller, and also the (critical) choice for a parameter is avoided. Nevertheless, for very large N we may expect that *multiple-grid* methods will be preferable to this combination of CGS with ILLU preconditioning (see, for instance, [19] and [24]). As has been shown many times, the quality of a method is strongly dependent on the quality of the preconditioning used. For that reason, it is important that codes be developed for ILLU factorisation of more complicated matrices, such as those arising from finite-element discretisations.

Acknowledgments. The author thanks Professors Piet Wesseling and Henk Van der Vorst and others at Delft for their stimulating discussions, and is grateful to Koos Meijerink for testing the CGS method under "real life" circumstances. Finally the author is indebted to the referees for some valuable remarks.

REFERENCES

[1] O. AXELSSON, *A Generalised Conjugate Direction Method and Its Application on a Singular Perturbation Problem*, Lecture Notes in Mathematics 773, Springer-Verlag, Berlin, Heidelberg, New York, 1980, pp. 1-11.

[2] ———, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1-16.

[3] ———, *Solution of Linear Systems of Equations: Iterative Methods*, Lecture Notes in Mathematics 572, Springer-Verlag, Berlin, Heidelberg, New York, 1977, pp. 2-51.

[4] P. CONCUS AND G. H. GOLUB, *A Generalised Conjugate Gradient Method for Unsymmetric Systems of Linear Equations*, Lecture Notes in Economics and Mathematical Systems 139, Springer-Verlag, Berlin, Heidelberg, New York, 1976.

[5] P. CONCUS, G. H. GOLUB, AND G. MEURANT, *Block preconditioning for the conjugate gradient method*, Report LBL-14856, Lawrence Berkeley Laboratories, University of California, Berkeley, CA, 1982.

[6] S. C. EISENSTAT, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1-4.

[7] H. C. ELMAN, Y. SAAD, AND P. E. SAYLOR, *A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations*, Tech. Report 301, Department of Computer Science, Yale University, New Haven, CT, 1984; SIAM J. Sci. Statist. Comput., 7 (1986), pp. 840-855.

[8] R. FLETCHER, *Conjugate Gradient Methods for Indefinite Systems*, Lecture Notes in Mathematics 506, Springer-Verlag, Berlin, Heidelberg, New York, 1976, pp. 73–89.
[9] I. GUSTAFSSON, *A class of first order factorisation methods*, BIT, 18 (1978), pp. 142–156.
[10] A. HUTTON AND R. SMITH, *J. Numer. Heat Transfer*, to appear.
[11] D. S. KERSHAW, *The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations*, J. Comp. Phys., 26 (1978), pp. 43–65.
[12] T. A. MANTEUFFEL, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183–208.
[13] ———, *The Tchebyschev iteration for nonsymmetric linear systems*, Numer. Math. 28 (1977), pp. 307–327.
[14] J. A. MEIJERINK, *Iterative methods for the solution of linear equations based on incomplete block factorisation of the matrix*, paper SPE 12262, Proc. 7th SPE Symposium on Reservoir Simulation, San Francisco, CA, November 1983, pp. 297–304.
[15] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[16] ———, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. Comp. Phys., 44 (1981), pp. 134–155.
[17] C. C. PAIGE AND M. A. SAUNDERS, LSQR: *An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[18] Y. SAAD AND H. SCHULTZ, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*, Res. Report 283, Department of Computer Science, Yale University, New Haven CT, 1983.
[19] P. SONNEVELD, P. WESSELING, AND P. M. DE ZEEUW, *Multigrid and conjugate gradient methods as convergence acceleration techniques*, in Proc. Multigrid Conference, H. Holstein and D. Paddon, eds., Bristol, 1983.
[20] C. THOMPSON AND N. WILKES, *Experiments with higher order finite difference formulae*, Report AERE R-10493, United Kingdom Atomic Energy Authority, Harwell, UK, 1983.
[21] H. A. VAN DER VORST, *Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from* PDE-*Problems*, J. Comp. Phys., 44 (1981), pp. 1–19.
[22] ———, *Preconditioning by incomplete decompositions*, Ph.D. thesis, ACCU-series 32, Academic Computer Centre Utrecht, University of Utrecht, the Netherlands, 1982.
[23] P. W. VINSOME, *Orthomin, an Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations*, paper SPE 5729, Society of Petroleum Engineers of AIME, 1976.
[24] P. WESSELING AND P. SONNEVELD, *Numerical Experiments with a Multiple Grid- and a Preconditioned Lanczos Type Method*, Lecture Notes in Mathematics 771, Springer-Verlag, Berlin, Heidelberg, New York, 1980, pp. 543–562.
[25] O. WIDLUND, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.
[26] D. M. YOUNG AND K. C. JEA, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.

Downloaded 10/31/22 to 141.219.226.219 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy