ELSEVIER

Journal of Computational and Applied Mathematics 71 (1996) 125–146

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Generalized conjugate gradient squared

Diederik R. Fokkema *, Gerard L.G. Sleijpen, Henk A. Van der Vorst

Mathematical Institute, Utrecht University, P.O. Box 80.010, NL-3508 TA Utrecht, Netherlands

Received 2 March 1995; revised 19 September 1995

Abstract

The Conjugate Gradient Squared (CGS) is an iterative method for solving nonsymmetric linear systems of equations. However, during the iteration large residual norms may appear, which may lead to inaccurate approximate solutions or may even deteriorate the convergence rate. Instead of squaring the Bi-CG polynomial as in CGS, we propose to consider products of two nearby Bi-CG polynomials which leads to generalized CGS methods, of which CGS is just a particular case. This approach allows the construction of methods that converge less irregularly than CGS and that improve on other convergence properties as well. Here, we are interested in a property that got less attention in literature: we concentrate on retaining the excellent approximation qualities of CGS with respect to components of the solution in the direction of eigenvectors associated with extreme eigenvalues. This property seems to be important in connection with Newton's scheme for nonlinear equations: our numerical experiments show that the number of Newton steps may decrease significantly when using a generalized CGS method as linear solver for the Newton correction equations.

Keywords: Nonsymmetric linear systems; Krylov subspace; Iterative solvers; Bi-CG; CGS; BiCGstab(ℓ); Nonlinear systems; Newton's method

AMS classification: 65F10

1. Introduction

There is no best iterative method for solving linear systems of equations [15]. However, in many applications a particular method is preferred. CGS [24] is a frequently used method, but the popularity of CGS has diminished over time, because of its irregular convergence behavior. Nevertheless, in some situations, for instance, when in combination with Newton's method for nonlinear equations in the context of device simulations, CGS is often still the method of choice. [1]

* Corresponding author. Current address: ISE Integrated Systems Engineering AG, Technopark Zürich, Technoparkstrasse 1, CH-8005 Zurich, Switzerland. E-mail: fokkema@ise.ch.

[1] Personal communication by W. Schilders and M. Driessen, Philips Research Laboratories. They have also observed that for their semiconductor device modeling, where the system is often expressed in terms of voltages, the conservation of currents is better maintained when working with CGS.

SSDI 0377-0427(95)00227-8

The observation is that a Newton scheme in combination with CGS usually solves the nonlinear problem in less Newton steps than a Newton scheme in combination with other iterative methods. And although other methods, e.g., Bi-CGSTAB, sometimes need less iteration steps to solve the linear equations involved, Newton in combination with CGS turns out to be more efficient (see also our examples in Section 7).

For other situations where CGS or CGS-type of methods, i.e., TFQMR [7], are preferred, see [1; 13, pp. 128–133].

However, the large intermediate residuals produced by CGS badly affect its speed of convergence and limit its attainable accuracy [23], and this in turn has a (very) negative effect on the convergence of the overall Newton process.

In this paper we discuss variants of CGS that have improved convergence properties, while still having the important "quadratic reduction" property discussed below.

We will now try to explain why CGS may be so successful as a linear solver in a Newton scheme. In our heuristic arguments the eigensystem, the eigenvalues λ_j, and the eigenvectors v_j, of the local Jacobian matrices (the matrices of partial derivatives of the nonlinear problem, evaluated at the approximation) play a role. We consider the components of the approximate solutions and residuals in the direction of these eigenvectors, distinguishing between components associated with exterior eigenvalues ("exterior components") and components associated with interior eigenvalues ("interior components"). By "exterior" and "interior" we refer to the position of the eigenvalue in the convex hull of the spectrum of the Jacobian matrix.

CGS (cf. Section 2) is based on Bi-CG [6,12]. This linear solver tends to approximate the exterior components of the solution better and faster than the interior components [11, 19]. Any residual of a linear solver that we consider can be represented by a polynomial in the matrix representing the linear system (for instance, the Bi-CG residual can be written as $r_k^{\text{Bi-CG}} = \phi_k(A)r_0$, where ϕ_k is a polynomial of degree k) and the size of the eigenvector components of the residual is proportional to the (absolute) value of the polynomial in the associated eigenvalue (for instance for Bi-CG $r_k^{\text{Bi-CG}} = \sum_{j=1}^{n} \phi_k(\lambda_j)v_j$).

The absolute value of Bi-CG polynomials tends to be smaller in the exterior eigenvalues than in the interior ones. A small component $\phi_k(\lambda_j)v_j$ of the residual r_k means that the corresponding component of the solution x_k is well approximated. CGS polynomials are the squares of Bi-CG polynomials: the residual of CGS can be written as $r_k^{\text{CGS}} = \phi_k^2(A)r_0$. Therefore, CGS approximations tend to have very accurate exterior components. A polynomial associated with, for instance, the BiCGstab methods is the product of a Bi-CG polynomial and another polynomial of the same degree ($r_k^{\text{BiCGstab}} = \tilde{\phi}_k(A)\phi_k(A)r_0$). This other polynomial (a product of locally minimizing degree 1 polynomials for Bi-CGSTAB ($\tilde{\phi}_k(t) = \prod_{i=1}^{k}(1 - \omega_i t)$) and a product of such polynomials of degree ℓ for BiCGstab(ℓ)) does not have this strong tendency of reducing in exterior eigenvalues better than in the interior ones. Therefore, comparing approximations with residuals of comparable size (2-norm), we may expect that approximate solutions as produced by a BiCGstab method have exterior components that are less accurate than those of the CGS approximations, since the error components are larger. Of course, with respect to interior components, the situation will be in favor of the BiCGstab methods.

Now, we come to the implication for Newton's method. The nonlinearity of the problem seems often stronger in the (linear combination of) exterior components than in the (linear combination of) interior ones. This fact explains the nice convergence properties in the outer iteration when CGS is

used in the inner iteration of Newton's process. CGS tends to deliver approximate solutions of which the exterior components are very accurate. With respect to these components, the Newton process, in which the linear systems are solved approximately by CGS, compares to a Newton process in which the linear systems are solved exactly, while this may not be true for the Newton process in combination with the BiCGstab methods (or others).

In summary, we wish to retain in our modifications the attractive property of CGS that it converges faster with respect to exterior components within a Newton method, without losing its efficiency, the fact that it is transpose free, and its fast convergence. However, we wish to avoid irregular convergence and large intermediate residuals, since they may badly affect the speed of convergence of the inner iteration.

Techniques as proposed in, e.g., [26,16,22] smooth down the convergence by operating a posteriori on approximates and residuals. Although they may lead to more accurate approximates (see the "additional note" in Section 7 or [22]), they do not change the speed of convergence. For a detailed discussion, see [22].

The polynomial associated with our new methods is the product of the Bi-CG polynomial with another "nearby" polynomial of the same degree (Section 4). We refer to these methods as *generalized CGS* methods. They are about as efficient as CGS per iteration step (Section 4). We pay special attention to the case where this second polynomial is a Bi-CG polynomial (Section 6.1) of another (nearby) Bi-CG process, or a polynomial closely related to such a Bi-CG polynomial (Section 6.2). The difference between the square of a Bi-CG polynomial and the product of two "nearby" Bi-CG polynomials of the same degree may seem insignificant, but, as we will see in our numerical examples in Section 7, this approach may lead to faster convergence in norm as well as to more accurate results. Moreover, this approach seems to improve the convergence of (exterior components in) nonlinear schemes. A discussion on the disadvantages of squaring the Bi-CG polynomial can be found in Section 3. Since we are working with products of Bi-CG polynomials, the new methods reduce exterior components comparable fast as CGS (Section 6.1). It is obvious that Bi-CG and the ideas behind CGS are essential in deriving the new methods and therefore Bi-CG and CGS are discussed in Section 2. In that section we also introduce most of our notation.

2. Bi-CG and CGS

The Bi-CG method [6, 12] is an iterative solution scheme for linear systems

$$Ax = b$$

in which A is some given nonsingular $n \times n$ matrix and b some given n-vector. Typically n is large and A is sparse. For ease of presentation, we assume A and b to be real.

Starting with an initial guess x_0, each iteration of Bi-CG computes an approximation x_k to the solution. It is well known that the Bi-CG residual $r_k = b - Ax_k$ can be written as $\phi_k(A)r_0$ where ϕ_k is a certain polynomial in the space $\mathcal{P}_k^1$ of all polynomials ψ of degree k for which $\psi(0) = 1$. The Bi-CG polynomial ϕ_k is implicitly defined by the Bi-CG algorithm through a coupled two-term recurrence:

$$u_k = r_k - \beta_k u_{k-1},$$

$$r_{k+1} = r_k - \alpha_k A u_k.$$

The iteration coefficients α_k and β_k follow from the requirement that r_k and Au_k are orthogonal to the Krylov subspace $\mathcal{K}_k(A^{\mathrm{T}};\tilde{r}_0)$ of order k, generated by A^{T} and an arbitrary, but fixed $\tilde{r}_0$.

If $(\tilde{\phi}_k)$ is some sequence of polynomials of degree k with a nontrivial leading coefficient θ_k then (see [24] or [20]):

$$\beta_k = \frac{\theta_{k-1}}{\theta_k}\frac{\rho_k}{\sigma_{k-1}} \quad \text{and} \quad \alpha_k = \frac{\rho_k}{\sigma_k}, \tag{1}$$

where

$$\rho_k = (r_k, \tilde{\phi}_k(A^{\mathrm{T}})\tilde{r}_0) \quad \text{and} \quad \sigma_k = (Au_k, \tilde{\phi}_k(A^{\mathrm{T}})\tilde{r}_0). \tag{2}$$

In standard Bi-CG the polynomial $\tilde{\phi}_k$ is taken to be the same as the Bi-CG polynomial: $\tilde{\phi}_k = \phi_k$, where ϕ_k is such that $r_k = \phi_k(A)r_0$. This leads to another coupled two-term recurrence in the Bi-CG algorithm:

$$\tilde{u}_k = \tilde{r}_k - \beta_k \tilde{u}_{k-1},$$

$$\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^{\mathrm{T}} \tilde{u}_k.$$

Since A and b are assumed to be real, this means that $\tilde{r}_k$ and $A^{\mathrm{T}}\tilde{u}_k$ are orthogonal to the Krylov subspace $\mathcal{K}_k(A;r_0)$, in particular the sequences (r_k) and $(\tilde{r}_k)$ are bi-orthogonal. Of course, other choices for $\tilde{\phi}_k$ are possible. For instance, when A and b are complex and if we still want to have bi-orthogonality, then we should choose $\tilde{\phi}_k = \bar{\phi}_k$.

The leading coefficient of ϕ_k is $(-\alpha_{k-1})(-\alpha_{k-2})\cdots(-\alpha_0)$ and therefore we have that

$$\frac{\theta_{k-1}}{\theta_k} = \frac{-1}{\alpha_{k-1}}$$

and thus

$$\beta_k = \frac{-1}{\alpha_{k-1}}\frac{\rho_k}{\sigma_{k-1}} \quad \text{and} \quad \alpha_k = \frac{\rho_k}{\sigma_k}.$$

A pseudo-code for the standard Bi-CG algorithm is given in Algorithm 1.

It was Sonneveld [24] who suggested to rewrite the inner products so as to avoid the operations with A^{T}, e.g.,

$$\rho_k = (r_k, \tilde{\phi}_k(A^{\mathrm{T}})\tilde{r}_0) = (\tilde{\phi}_k(A)r_k, \tilde{r}_0), \tag{3}$$

and to take advantage of both ϕ_k and $\tilde{\phi}_k$ for the reduction of the residual by generating recurrences for the vectors $\boldsymbol{r}_k = \tilde{\phi}_k(A)r_k$. In fact, he suggested to take $\tilde{\phi}_k = \phi_k$, which led to the CGS method: $\boldsymbol{r}_k = \phi_k^2(A)r_0$. The corresponding search directions $\boldsymbol{u}_k$ for the corresponding approximation $\boldsymbol{x}_k$ can be easily constructed. In this approach the Bi-CG residuals r_k and search directions u_k themselves are not computed explicitly, nor are they needed in the process. See Algorithm 2 for CGS.

As is explained in [25], $\phi_k(A)$ may not be a particularly well suited reduction operator for $\phi_k(A)r_0$. But, as we will see in Section 3, there are more arguments for not selecting $\tilde{\phi}_k = \phi_k$. For instance, we wish to avoid irregular convergence and large intermediate residuals. In [25] it was suggested to choose $\tilde{\phi}_k$ as a product of linear factors, which were constructed to minimize residuals in only one

direction at a time. This led to the Bi-CGSTAB algorithm. This was further generalized to a composite of higher-degree factors which minimize residuals over ℓ-dimensional subspaces: BiCGStab2, for $\ell = 2$, in [10], and BiCGstab(ℓ), the more efficient and more stable variant also for general ℓ, in [20, 23] (see also [21]).

Choose an initial guess x_0 and some $\tilde{r}_0$
$r_0 = b - Ax_0$
$u_{-1} = \tilde{u}_{-1} = 0$, $\alpha_{-1} = \sigma_{-1} = 1$
for $k = 0, 1, 2, \ldots$ do
  $\rho_k = (r_k, \tilde{r}_k)$
  $\beta_k = (-1/\alpha_{k-1})(\rho_k/\sigma_{k-1})$
  $u_k = r_k - \beta_k u_{k-1}$
  $\tilde{u}_k = \tilde{r}_k - \beta_k \tilde{u}_{k-1}$
  $c = Au_k$
  $\sigma_k = (c, \tilde{r}_k)$
  $\alpha_k = \rho_k/\sigma_k$
  $x_{k+1} = x_k + \alpha_k u_k$
  if x_{k+1} is accurate enough, then quit
  $r_{k+1} = r_k - \alpha_k c$
  $\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^T \tilde{u}_k$
end

Algorithm 1. Bi-CG

Choose an initial guess $\boldsymbol{x}_0$ and some $\tilde{r}_0$
$\boldsymbol{r}_0 = b - A\boldsymbol{x}_0$
$\boldsymbol{u}_{-1} = \boldsymbol{w}_{-1} = 0$, $\alpha_{-1} = \sigma_{-1} = 1$
for $k = 0, 1, 2, \ldots$ do
  $\rho_k = (\boldsymbol{r}_k, \tilde{r}_0)$
  $\beta_k = (-1/\alpha_{k-1})(\rho_k/\sigma_{k-1})$
  $\boldsymbol{v}_k = \boldsymbol{r}_k - \beta_k \boldsymbol{u}_{k-1}$
  $\boldsymbol{w}_k = \boldsymbol{v}_k - \beta_k(\boldsymbol{u}_{k-1} - \beta_k \boldsymbol{w}_{k-1})$
  $\boldsymbol{c} = A\boldsymbol{w}_k$
  $\sigma_k = (\boldsymbol{c}, \tilde{r}_0)$
  $\alpha_k = \rho_k/\sigma_k$
  $\boldsymbol{u}_k = \boldsymbol{v}_k - \alpha_k \boldsymbol{c}$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(\boldsymbol{v}_k + \boldsymbol{u}_k)$
  if $\boldsymbol{x}_{k+1}$ is accurate enough, then quit
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A(\boldsymbol{v}_k + \boldsymbol{u}_k)$
end

Algorithm 2. CGS

Obviously, there is a variety of possibilities for the polynomials $\tilde{\phi}_k$. In the next sections we investigate polynomials that are similar to the Bi-CG polynomial, i.e., polynomials that are defined by a coupled two-term recurrence. This leads to a generalized CGS (GCGS) algorithm, of which CGS and Bi-CGSTAB are just particular instances.

3. Disadvantages of squaring the iteration polynomial

For an eigenvalue λ of A, the component of the CGS residual, in the direction of the eigenvector associated with λ, is equal to $v_\lambda \phi_k(\lambda)^2$, where v_λ is the component of r_0 in the direction of the same eigenvector (assuming λ is a semi-simple eigenvalue). The corresponding component of the Bi-CG residual is precisely $v_\lambda \phi_k(\lambda)$ and the tendency of $|\phi_k(\lambda)|$ to be small for nonlarge k and for exterior λ explains the good reduction abilities of CGS with respect to the exterior components.

Unfortunately, squaring has disadvantages: $|\phi_k(\lambda)|^2$ may be large even if $|v_\lambda \phi_k(\lambda)|$ is moderate. This may happen especially during the initial stage of the process (when k is small) and the CGS component will be extremely large. In such a case, the CGS residual $\boldsymbol{r}_k$ is extremely large. Although the next residual $\boldsymbol{r}_{k+1}$ may be moderate, a single large residual is enough to prevent the process of finding an accurate final approximate solution in finite precision arithmetic: in [23, Section 2.2], (see also [21]), it was shown that

$$|\|\boldsymbol{r}_m\|_2 - \|b - A\boldsymbol{x}_m\|_2| \leqslant \bar{\xi}\, \Gamma \max_{k \leqslant m} \|\boldsymbol{r}_k\|_2 \quad \text{with} \quad \Gamma := m\, n_A\, \|A^{-1}\|_2\, \| |A| \|_2, \tag{4}$$

where $\bar{\xi}$ is the relative machine precision and n_A is the maximum number of nonzero entries per row of A. Except for the constant Γ this estimate seems to be sharp in practice: in actual computations we do not see the factor Γ (see [21]). Moreover the local bi-orthogonality, essential for Bi-CG and (implicitly) for CGS, will seriously be disturbed.[2] This will slow down the speed of convergence. CGS is notorious for large intermediate residuals and irregular convergence behavior. The fact that $|\phi_k(\lambda)|^2$ can be large was precisely the reason in [25] to reject the choice $\tilde{\phi}_k = \phi_k$ and to consider a product of degree 1 factors that locally minimize the residual with respect to the norm $\|\cdot\|_2$. As anticipated, this approach usually improves the attainable *accuracy*, (i.e., the distance between $\|\boldsymbol{r}_m\|_2$ and $\|b - A\boldsymbol{x}_m\|_2$; cf. (4)) as well as the rate of convergence. However, the degree 1 factors do not tend to favor the reduction of the exterior components as we wish here.

In summary, we wish to avoid "quadratically large" residual components, while retaining "quadratically small" components.

Of course the selected polynomials $\tilde{\phi}_k$ should also lead to an efficient algorithm. Before specifying polynomials $\tilde{\phi}_k$ in the Sections 5 and 6, we address this efficiency subject in Section 4.

4. Generalized CGS: methods of CGS type

In this section we derive an algorithm that delivers $\boldsymbol{r}_k = \tilde{\phi}_k(A)\phi_k(A)r_0$, where $\tilde{\phi}_k$ is a polynomial defined by a coupled two-term recurrence and where ϕ_k is the Bi-CG polynomial.

[2] The Neumaier trick (cf. the "additional note" in Section 7) cures the loss of accuracy, but it does not improve the speed of convergence [22].

Consider the Bi-CG recurrence for the search directions u_k and the residuals r_{k+1}

$$u_{-1} = 0, \quad r_0 = b - Ax_0, \tag{5}$$

$$u_k = r_k - \beta_k u_{k-1}, \tag{6}$$

$$r_{k+1} = r_k - \alpha_k A u_k, \tag{7}$$

and the polynomial recurrence for the polynomials $\tilde{\phi}_{k+1}$ and $\tilde{\psi}_k$ evaluated in A

$$\tilde{\psi}_{-1}(A) \equiv 0, \quad \tilde{\phi}_0(A) \equiv I, \tag{8}$$

$$\tilde{\psi}_k(A) = \tilde{\phi}_k(A) - \tilde{\beta}_k \tilde{\psi}_{k-1}(A), \tag{9}$$

$$\tilde{\phi}_{k+1}(A) = \tilde{\phi}_k(A) - \tilde{\alpha}_k A \tilde{\psi}_k(A), \tag{10}$$

for scalar sequences $(\tilde{\alpha}_k)$ and $(\tilde{\beta}_k)$. For ease of notation we will write Φ_k for $\tilde{\phi}_k(A)$ and Ψ_k for $\tilde{\psi}_k(A)$ from now on. Our goal is to compute $r_{k+1} = \Phi_{k+1} r_{k+1}$. We will concentrate on the vector updates first, i.e., for the moment we will assume that the iteration coefficients $\tilde{\alpha}_k$ and $\tilde{\beta}_k$ are explicitly given.

Suppose we have the following vectors at step k:

$$\Psi_{k-1} u_{k-1}, \qquad \Psi_{k-1} r_k, \qquad \Phi_k u_{k-1} \quad \text{and} \quad \Phi_k r_k. \tag{11}$$

Note that for $k = 0$ these vectors are well defined. We proceed by showing how the index of vectors in (11) can be increased.

We use the Bi-CG recurrence (6) to update $\Phi_k u_k$:

$$\Phi_k u_k = \Phi_k r_k - \beta_k \Phi_k u_{k-1}.$$

Before we can update $\Psi_k u_k$ a similar way, i.e.,

$$\Psi_k u_k = \Psi_k r_k - \beta_k \Psi_k u_{k-1}, \tag{12}$$

we need the vectors $\Psi_k r_k$ and $\Psi_k u_{k-1}$. These vectors follow from (9):

$$\Psi_k r_k = \Phi_k r_k - \tilde{\beta}_k \Psi_{k-1} r_k, \tag{13}$$

$$\Psi_k u_{k-1} = \Phi_k u_{k-1} - \tilde{\beta}_k \Psi_{k-1} u_{k-1}. \tag{14}$$

This in combination with (12) gives us $\Psi_k u_k$. The vectors $\Psi_k r_{k+1}$ and $\Phi_{k+1} u_k$ follow from (7) and (10):

$$\Psi_k r_{k+1} = \Psi_k r_k - \alpha_k A \Psi_k u_k,$$

$$\Phi_{k+1} u_k = \Phi_k u_k - \tilde{\alpha}_k A \Psi_k u_k.$$

Finally, to obtain $\Phi_{k+1} r_{k+1}$ we apply the recurrences (7) and (10):

$$\Phi_k r_{k+1} = \Phi_k r_k - \alpha_k A \Phi_k u_k, \tag{15}$$

$$\Phi_{k+1} r_{k+1} = \Phi_k r_{k+1} - \tilde{\alpha}_k A \Psi_k r_{k+1}. \tag{16}$$

When α_k and $\tilde{\alpha}_k$ are known before updating (15) and (16), we can avoid one of the matrix-vector products by combining these equations. This leads to

$$\Phi_{k+1}r_{k+1} = \Phi_k r_k - A(\alpha_k \Phi_k u_k - \tilde{\alpha}_k \Psi_k r_{k+1}),$$

and, hence, we only need $A\Psi_k u_k$ and $A(\alpha_k \Phi_k u_k - \tilde{\alpha}_k \Psi_k r_{k+1})$ in order to complete one iteration step, and the corresponding computational scheme needs two matrix-vector multiplications per iteration step, just as CGS.

The iteration coefficients α_k and β_k have to be computed such that r_k and Au_k are orthogonal to the Krylov subspace $\mathcal{K}_k(A^{\mathrm{T}}; \tilde{r}_0)$. According to (1) and (2), these coefficients are determined by θ_{k-1}/θ_k, ρ_k and σ_k. From (10) it follows that the leading coefficient of $\tilde{\phi}_k$ is $(-\tilde{\alpha}_{k-1})(-\tilde{\alpha}_{k-2})\cdots(-\tilde{\alpha}_0)$ and hence

$$\frac{\theta_{k-1}}{\theta_k} = \frac{-1}{\tilde{\alpha}_{k-1}}.$$

For the scalar ρ_k we can rewrite the inner product (cf. (3)):

$$\rho_k = (\Phi_k r_k, \tilde{r}_0).$$

Note that the vector $\Phi_k r_k$ is available. However, for the scalar σ_k rewriting the inner product does not help because $A\Phi_k u_k$ is no longer available, since we have combined (15) and (16). Fortunately, we can replace $A\Phi_k u_k$ by the vector $A\Psi_k u_k$, which is available. It follows from (9) that the degree of $\tilde{\psi}_k - \tilde{\phi}_k$ is smaller than k and thus that

$$(A(\Phi_k - \Psi_k)u_k, \tilde{r}_0) = (Au_k, (\tilde{\phi}_k(A^{\mathrm{T}}) - \tilde{\psi}_k(A^{\mathrm{T}}))\tilde{r}_0) = 0.$$

Therefore, we have that

$$\sigma_k = (A\Psi_k u_k, \tilde{r}_0).$$

The algorithm for this *generalized CGS* (GCGS) method is given as Algorithm 3. In this algorithm, the following substitutions have been made:

$$u_{k-1} = \Phi_k u_{k-1}, \qquad v_k = \Phi_k u_k, \qquad w_k = \Psi_k u_k,$$
$$r_k = \Phi_k r_k, \qquad s_k = \Psi_k r_{k+1} \quad \text{and} \quad t_k = \Psi_k r_k,$$

and Eqs. (12) and (14) are combined to one single equation.

The algorithm is very similar to Algorithm 2. In terms of computational work per iteration step (per 2 matrix-vector multiplications) GCGS needs only two more vector updates than CGS; moreover, GCGS needs only storage for two more vectors.

In Sections 5 and 6 we will discuss some possible choices for the recurrence coefficients of the polynomials $\tilde{\phi}_k$. Section 5 contains the well known CGS and Bi-CGSTAB method. The new methods can be found in Section 6.

5. Well-known methods of CGS type

We present here the CGS and the Bi-CGSTAB method to show that they fit in the frame of generalized CGS methods and to facilitate comparison of efficiencies.

Choose an initial guess x_0 and some $\tilde{r}_0$
$r_0 = b - Ax_0$
$u_{-1} = w_{-1} = s_{-1} = 0,$
$\alpha_{-1} = \sigma_{-1} = \tilde{\alpha}_{-1} = \tilde{\sigma}_{-1} = 1$
for $k = 0, 1, 2, \ldots$ do
  $\rho_k = (r_k, \tilde{r}_0)$
  $\beta_k = (-1/\tilde{\alpha}_{k-1})(\rho_k/\sigma_{k-1})$
  $v_k = r_k - \beta_k u_{k-1}$
  choose $\tilde{\beta}_k$
  $t_k = r_k - \tilde{\beta}_k s_{k-1}$
  $w_k = t_k - \beta_k(u_{k-1} - \tilde{\beta}_k w_{k-1})$
  $c = Aw_k$
  $\sigma_k = (c, \tilde{r}_0)$
  $\alpha_k = \rho_k/\sigma_k$
  $s_k = t_k - \alpha_k c$
  choose $\tilde{\alpha}_k$
  $u_k = v_k - \tilde{\alpha}_k c$
  $x_{k+1} = x_k + \alpha_k v_k + \tilde{\alpha}_k s_k$
  if x_{k+1} is accurate enough, then quit
  $r_{k+1} = r_k - A(\alpha_k v_k + \tilde{\alpha}_k s_k)$
end

Algorithm 3. GCGS

5.1. CGS: using the Bi-CG polynomials

The choice $\tilde{\alpha}_k = \alpha_k$, $\tilde{\beta}_k = \beta_k$ leads to CGS. In Algorithm 3 the vectors v_k and t_k as well as the vectors u_k and s_k are identical in this situation and some computational steps are now redundant.

5.2. Bi-CGSTAB: using products of optimal first-degree factors

As explained in Section 3, to avoid irregular convergence and large intermediate residuals, a product of degree 1 factors that locally minimize the residual is suggested in [25].

In our formulation Bi-CGSTAB can be obtained as follows. Take $\tilde{\beta}_k = 0$, so that the recurrences (12)–(16) reduce to

$$\Phi_k u_k = \Phi_k r_k - \beta_k \Phi_k u_{k-1}, \tag{17}$$

$$\Phi_k r_{k+1} = \Phi_k r_k - \alpha_k A \Phi_k u_k, \tag{18}$$

$$\Phi_{k+1} u_k = \Phi_k u_k - \tilde{\alpha}_k A \Phi_k u_k, \tag{19}$$

$$\Phi_{k+1} r_{k+1} = \Phi_k r_{k+1} - \tilde{\alpha}_k A \Phi_k r_{k+1}, \tag{20}$$

and take $\tilde{\alpha}_k$ such that $\|\Phi_k r_{k+1} - \tilde{\alpha}_k A \Phi_k r_{k+1}\|_2$ is minimal. Hence,

$$\tilde{\alpha}_k = (\Phi_k r_{k+1}, A\Phi_k r_{k+1})/(A\Phi_k r_{k+1}, A\Phi_k r_{k+1}).$$

For efficiency reasons one usually combines (17) and (19). Notice that $\tilde{\phi}_k$ is now a product of the linear factors $(1 - \tilde{\alpha}_k t)$.

A pseudo-code for Bi-CGSTAB is given in Algorithm 4.

Choose an initial guess x_0 and some $\tilde{r}_0$
$r_0 = b - Ax_0$
$u_{-1} = w_{-1} = s_{-1} = 0,$
$\alpha_{-1} = \sigma_{-1} = \tilde{\alpha}_{-1} = \tilde{\sigma}_{-1} = 1$
for $k = 0, 1, 2, \ldots$ do
  $\rho_k = (r_k, \tilde{r}_0)$
  $\beta_k = (-1/\tilde{\alpha}_{k-1})(\rho_k/\sigma_{k-1})$
  $w_k = r_k - \beta_k(w_{k-1} - \tilde{\alpha}_k c_{k-1})$
  $c_k = Aw_k$
  $\sigma_k = (c_k, \tilde{r}_0)$
  $\alpha_k = \rho_k/\sigma_k$
  $s_k = r_k - \alpha_k c_k$
  $t_k = As_k$
  $\tilde{\alpha}_k = (s_k, t_k)/(t_k, t_k)$
  $x_{k+1} = x_k + \alpha_k w_k + \tilde{\alpha}_k s_k$
  if x_{k+1} is accurate enough, then quit
  $r_{k+1} = s_k - \tilde{\alpha}_k t_k$
end

Algorithm 4. Bi-CGSTAB

---

6. New methods of CGS type

The BiCGstab methods generally converge more smoothly than CGS, but they do not approximate the exterior components of the solution as well as CGS. We wish to preserve this approximation property, while smoothing the convergence at the same time.[3]

6.1. CGS2: using related Bi-CG polynomials

We will argue that a related Bi-CG polynomial will meet our conditions to a certain extent: in this subsection, $\tilde{\phi}_k$ is the Bi-CG polynomial generated by r_0 and $\tilde{s}_0$, some vector different from $\tilde{r}_0$. For $\tilde{s}_0$ one may take, for instance, a random vector. The roots of the Bi-CG polynomial ϕ_k converge (for increasing k) towards eigenvalues corresponding to eigenvectors with nonzero weights

[3] As explained in Section 3, we want smooth convergence by avoiding a priori extremely large components. The smoothing techniques in, e.g., [26], work a posteriori and do *not* affect the rate of convergence nor the attainable accuracy.

v_i. The roots of this $\tilde{\phi}_k$ will converge to the same eigenvalues, but, and this is important, in a *different* manner. If both ϕ_k and $\tilde{\phi}_k$ reduce a component of r_0 poorly, as may be the case initially, then both corresponding roots have not yet converged to the corresponding eigenvalue. In other words, both $\phi_k(\lambda)$ and $\tilde{\phi}_k(\lambda)$ are significantly different from zero. But, since both polynomials are distinct, the product $|\tilde{\phi}_k(\lambda)\phi_k(\lambda)|$ is smaller than $\max(|\phi_k(\lambda)|^2, |\tilde{\phi}_k(\lambda)|^2)$ and one may hope that applying $\tilde{\phi}_k(A)\phi_k(A)$ as a reduction operator to r_0 will not lead to as bad an amplification of this component as $\phi_k(A)^2$ (CGS). If both ϕ_k and $\tilde{\phi}_k$ reduce a component of r_0 well, as may be the case later in the iteration process, then both corresponding roots have converged to the same corresponding eigenvalue. In this case we have a quadratic reduction of that component.

We give more details for the resulting scheme. The scheme is very efficient: although we work (implicitly) with two different Bi-CG processes the iteration steps do not require matrix-vector multiplications in addition to the ones in the GCGS scheme (Algorithm 3). As before, we write $\tilde{\phi}_k(A)$ as Φ_k (and $\tilde{\psi}_k(A)$ as Ψ_k).

As with the Bi-CG polynomial ϕ_k (see Section 2), we seek coefficients $\tilde{\alpha}_k$ and $\tilde{\beta}_k$ such that $\Phi_k r_0$ and $A\Psi_k r_0$ are orthogonal to the Krylov subspace $\mathcal{K}_k(A^T; \tilde{s}_0)$. According to (1) and (2) we may write

$$\tilde{\beta}_k = \frac{\tilde{\theta}_{k-1}}{\tilde{\theta}_k}\frac{\tilde{\rho}_k}{\tilde{\sigma}_{k-1}} \quad \text{and} \quad \tilde{\alpha}_k = \frac{\tilde{\rho}_k}{\tilde{\sigma}_k},$$

where

$$\tilde{\rho}_k = (\Phi_k r_0, \chi_k(A^T)\tilde{s}_0) \quad \text{and} \quad \tilde{\sigma}_k = (A\Psi_k r_0, \chi_k(A^T)\tilde{s}_0) \tag{21}$$

and $\tilde{\theta}_k$ is the leading coefficient of some polynomial χ_k of degree k and $\chi_k(0) = 1$. Normally, like in Bi-CG, practically any choice for χ_k would lead to another two-term recurrence in order to construct a basis for $\mathcal{K}_k(A^T; \tilde{s}_0)$. That would make the algorithm expensive, especially since matrix multiplications are involved. Surprisingly, we can avoid this construction, and thus the computational work, because we can use the Bi-CG polynomial ϕ_k, which is already (implicitly) available. Replacing χ_k by ϕ_k in (21) gives us for the iteration coefficients that

$$\tilde{\beta}_k = \frac{-1}{\alpha_{k-1}}\frac{\tilde{\rho}_k}{\tilde{\sigma}_{k-1}} \quad \text{and} \quad \tilde{\alpha}_k = \frac{\tilde{\rho}_k}{\tilde{\sigma}_k},$$

where

$$\tilde{\rho}_k = (\Phi_k r_0, \phi_k(A^T)\tilde{s}_0) = (\Phi_k \phi_k(A) r_0, \tilde{s}_0) = (\boldsymbol{r}_k, \tilde{s}_0)$$

$$\tilde{\sigma}_k = (A\Psi_k r_0, \phi_k(A^T)\tilde{s}_0) = (A\Psi_k \phi_k(A) r_0, \tilde{s}_0) = (A\boldsymbol{v}_k, \tilde{s}_0).$$

Here we used the fact that multiplication with polynomials in A is commutative:

$$\phi_k(A)\,\tilde{\phi}_k(A) r_0 = \tilde{\phi}_k(A)\,\phi_k(A) r_0 = \Phi_k r_k$$

$$\phi_k(A)\,\tilde{\psi}_k(A) r_0 = \tilde{\psi}_k(A)\,\phi_k(A) r_0 = \Psi_k r_k.$$

Choose an initial guess x_0, some $\tilde{r}_0$ and some $\tilde{s}_0$
$r_0 = b - Ax_0$
$u_{-1} = w_{-1} = s_{-1} = 0,$
$\alpha_{-1} = \sigma_{-1} = \tilde{\alpha}_{-1} = \tilde{\sigma}_{-1} = 1$
for $k = 0, 1, 2, \ldots$ do
$\rho_k = (r_k, \tilde{r}_0)$
$\beta_k = (-1/\tilde{\alpha}_{k-1})(\rho_k/\sigma_{k-1})$
$v_k = r_k - \beta_k u_{k-1}$
$\tilde{\rho}_k = (r_k, \tilde{s}_0)$
$\tilde{\beta}_k = (-1/\alpha_{k-1})(\tilde{\rho}_k/\tilde{\sigma}_{k-1})$
$t_k = r_k - \tilde{\beta}_k s_{k-1}$
$w_k = t_k - \beta_k(u_{k-1} - \tilde{\beta}_k w_{k-1})$
$c = Aw_k$
$\sigma_k = (c, \tilde{r}_0)$
$\alpha_k = \rho_k/\sigma_k$
$s_k = t_k - \alpha_k c$
$\tilde{\sigma}_k = (c, \tilde{s}_0)$
$\tilde{\alpha}_k = \tilde{\rho}_k/\tilde{\sigma}_k$
$u_k = v_k - \tilde{\alpha}_k c$
$x_{k+1} = x_k + \alpha_k v_k + \tilde{\alpha}_k s_k$
if x_{k+1} is accurate enough, then quit
$r_{k+1} = r_k - A(\alpha_k v_k + \tilde{\alpha}_k s_k)$
end

Algorithm 5. CGS2

A pseudo-code for this computational scheme, which we will call *CGS2*, is given in Algorithm 5. Compared with CGS, CGS2 needs two more vector updates and two more inner products per iteration and storage for three additional vectors.

6.2. *Shifted CGS: using delayed Bi-CG polynomials*

We wish to avoid extremely large factors $|\tilde{\phi}_k(\lambda)\phi_k(\lambda)|$. Such factors may occur if $\tilde{\phi}_k\phi_k$ has a (nearly) double root corresponding to an eigenvector component that has not converged (yet). The choice of $\tilde{\phi}_k$ in the preceding section does not exclude this possibility. In our approach below, we explicitly try to avoid these unwanted double roots associated with eigenvector components that have not converged yet. As before, we still want to have near double roots corresponding to converged components.

It is well known that for $A = A^T$ the roots of the Bi-CG polynomial ϕ_{k-1} separate those of ϕ_k and a product of these two polynomials seems to be a good candidate for avoiding large residuals. This idea can be implemented as follows.

For some $\mu \in \mathbb{C}$, take

$$\widetilde{\phi}_k(\lambda) = (1 - \mu\lambda)\,\phi_{k-1}(\lambda),$$

or equivalently, take

$$\widetilde{\beta}_k = 0 \quad \text{and} \quad \widetilde{\alpha}_k = \mu, \qquad \text{for } k = 0,$$
$$\widetilde{\beta}_k = \beta_{k-1} \quad \text{and} \quad \widetilde{\alpha}_k = \alpha_{k-1}, \qquad \text{for } k > 0.$$

in the GCGS algorithm in Algorithm 3.

A possible choice for μ is, for instance, the inverse of an approximation for the largest eigenvalue of A. This value for μ can be roughly determined with Gershgorin disks or with a few steps of Arnoldi's algorithm. As explained in the previous section, one may expect a smoother convergence behavior for symmetric problems. Additionally, the choice of μ may reduce the influence of the largest eigenvalue on the convergence behavior. For general nonsymmetric problems, where complex roots may appear, one may hope for a similar behavior of this scheme. We will refer to this approach as *Shifted CGS*. Note that in terms of computational work we save 2 inner products as compared with CGS2.

7. Numerical examples

The new GCGS methods (in Section 6) do not seem to be superior to BiCGstab(ℓ) as solvers of linear equations (although it seems they can compete). This should not come as a surprise, because they were not designed for this purpose. However, as we will see, they can be attractive as linear solver in a Newton scheme for nonlinear equations. The GCGS methods improve on CGS, with smoother and faster convergence, avoiding large intermediate residuals, leading to more accurate approximations (see, Section 7.1 for CGS2, and Section 7.2 for Shifted CGS). At the same time, they seem to maintain the good reduction properties of CGS with respect to the exterior components, and thus improving on BiCGstab methods as solvers in a Newton scheme (see, Sections 7.3 and 7.4).

For large realistic problems, it is hard to compare explicitly the effects of the linear solvers on, say, the exterior components. Here, we support the validity of our heuristic arguments by showing that the convergence behavior in the numerical examples is in line with our predictions.

7.1. Characteristics of CGS2

The CGS2 algorithm (Algorithm 5), which uses a product of two nearby Bi-CG polynomials, was tested with success at IIS in Zürich (this section) and at Philips in Eindhoven (Section 7.3).

At IIS the package `PILS` [18] written in C and FORTRAN was used for solving two linear systems extracted from the simulation of two devices called `dr15c` and `mct70c`, respectively. The computations were done on a Sun Sparc 10 in double precision ($\overline{\xi} \approx 0.5 \times 10^{-15}$) and ILU(0) [18,14] was used as a preconditioner. The plots in Figs. 1 and 2 show the convergence behavior of CGS2,

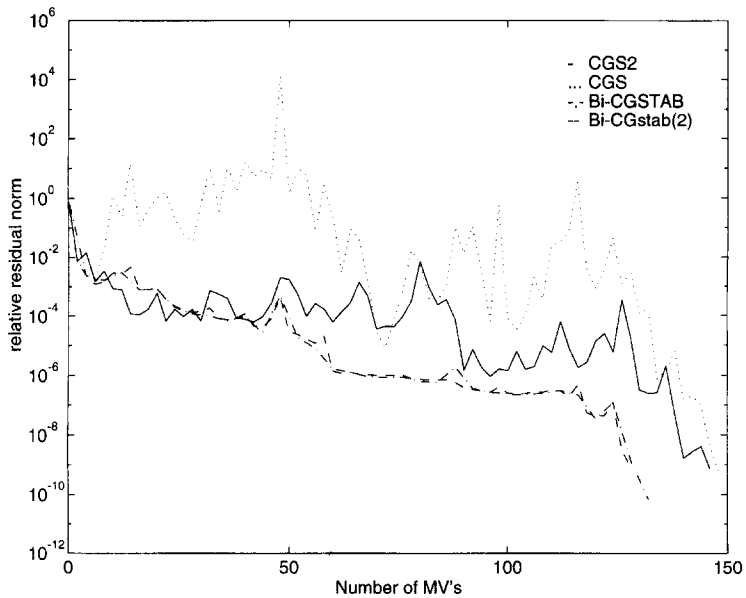

Fig. 1. Convergence history for device dr15c, 469741 nonzeros (ILU(0) preconditioning).

CGS, Bi-CGSTAB, and BiCGstab(2). In CGS2 we took random vectors for both $\tilde{r}_0$ and $\tilde{s}_0$. In the other methods we took the standard choice $\tilde{r}_0 = r_0$. Along the x-axis the number of matrix-vector multiplications is given. The y-axis represents the relative residual norm $\|r_k\|_2/\|r_0\|_2$ on logarithmic scale.

One observation is that CGS2 does not amplify the initial residual as much as CGS does. Its convergence behavior is much smoother. When we tried CGS with a random vector as $\tilde{r}_0$ too, its convergence behavior improved, but still CGS2 was better. Furthermore, the plots show that CGS2 can compete with Bi-CGSTAB and BiCGstab(2). The accuracy of the approximate solution delivered by all methods was of comparable size, except for the accuracy of the approximate solution of CGS in mct70c, which was two orders of magnitude less than the others. Since the iterations were terminated when $\|r_k\|_2/\|r_0\|_2 \leqslant 10^{-9}$ and the relative machine precision is $\bar{\xi} \approx 0.5 \times 10^{-15}$, these results are in line with (4).

7.2. Characteristics of shifted CGS

The shifted CGS method (Section 6.2) uses a combination with a lower degree Bi-CG polynomial. In the examples to be discussed next, the parameter μ was taken as the inverse of the real part of the largest eigenvalue estimate, delivered by a few steps of Arnoldi's algorithm. Along the x-axis the number of matrix-vector multiplications is given and the y-axis represents the scaled true residual norm $\|b - Ax_k\|_2/\|r_0\|_2$.

7.2.1. Example 1

We start with a system with a symmetric positive–definite matrix. For such a system the polynomial behavior of the error reduction with CGS is more easily interpreted, and because of the

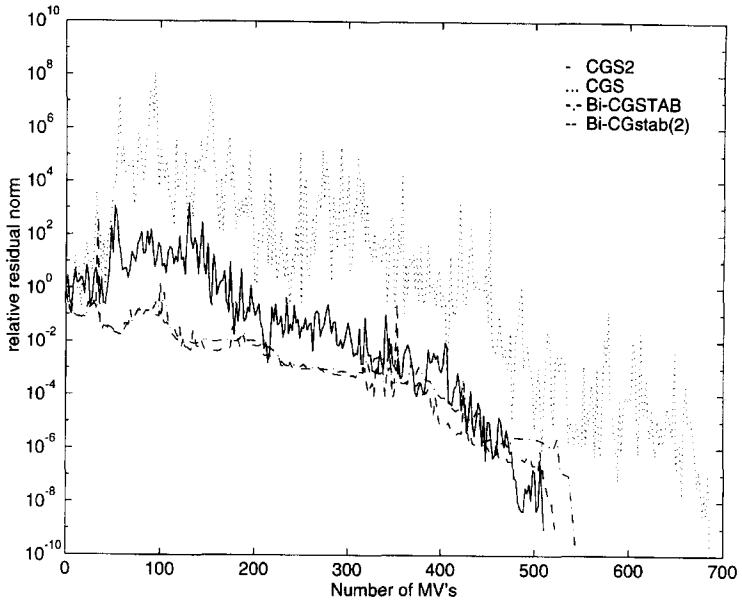

Fig. 2. Convergence history for device mct70c, 1969203 nonzeros (ILU(0) preconditioning).

orthogonal eigensystem the situation is not further confused with arguments in angles between subspaces. The linear system stems from a (82×83) finite volume discretization over the unit square, with Dirichlet boundary conditions along $y = 0$ and Neumann conditions along the other parts of the boundary, of

$$-(Du_x)_x - (Du_y)_y = 1,$$

where the function D is defined as

$$D = 1000 \quad \text{for } 0.1 \leqslant x, y \leqslant 0.9, \text{ and } D = 1 \text{ elsewhere.}$$

Symmetric ILU(0) preconditioning [14] was used. Fig. 3 confirms our heuristic arguments that for a symmetric system a combination with a lower degree Bi-CG polynomial can make the convergence somewhat smoother. The three peaks in the convergence history for CGS are not found in the history for our shifted CGS variant. Notice that full accuracy of the approximate solution is attained for both methods, as may be explained by the fact that no residual norm is larger than the initial residual norm (cf. (4)).

7.2.2. Example 2

This example is taken from [3]. The linear system stems from a (42×42) finite volume discretization over the unit square with Dirichlet boundary conditions, of

$$-\Delta u + 2\exp(2(x^2 + y^2))u_x - 100u = F,$$

where the right–hand side is taken such that the vector with all ones is the solution. ILU(0) preconditioning was used.

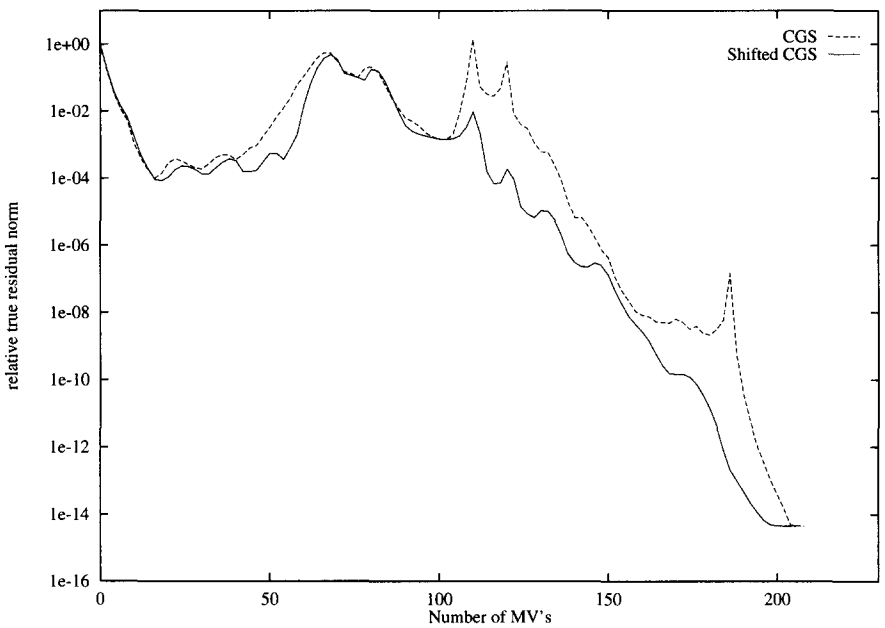

Fig. 3. Convergence history for example 1, 33292 nonzeros (ILU(0) preconditioning).

The convergence behavior of both methods reflects the fact that the matrix is now nonsymmetric. Both methods converge less smoothly. The improvement is not impressive, but on the average it seems that all residual norms of shifted CGS stay well below those with CGS. As a result, the accuracy of shifted CGS is two orders of magnitude better compared to the accuracy with CGS: the maximum intermediate residual for CGS is $\approx 3 \times 10^2$ times larger than the maximum one of shifted CGS (cf. Fig. 4 and (4)).

7.3. CGS2 as linear solver in a Newton scheme

In this subsection we discuss numerical results of nonlinear device simulations in which the linear system of equations that appear in Newton's method are solved by CGS2, CGS, and Bi-CGSTAB. The numerical data for the figures was obtained from device simulations with the package CURRY [17], written in FORTRAN, developed at Philips in Eindhoven. In this package the Jacobian matrices are explicitly formed. The computations were done on an HP workstation in double precision. With this package the evolution of the solution of the device CAPO1T with respect to Time (in seconds), and of the device DIODE with respect to voltage (in volt), is followed by a continuation method [8].

In each continuation step the value of the relevant parameter (volt or second) is increased by a certain step size and the next solution is computed by a Newton process in which the solution of the previous step is used as an initial guess. The irregularities in the convergence history in the figures are caused by convergence failures of the Newton process. In case Newton's method fails to converge for a particular step size, CURRY tries to solve this subproblem by using the continuation method with the step size halved. If Newton's method again fails to

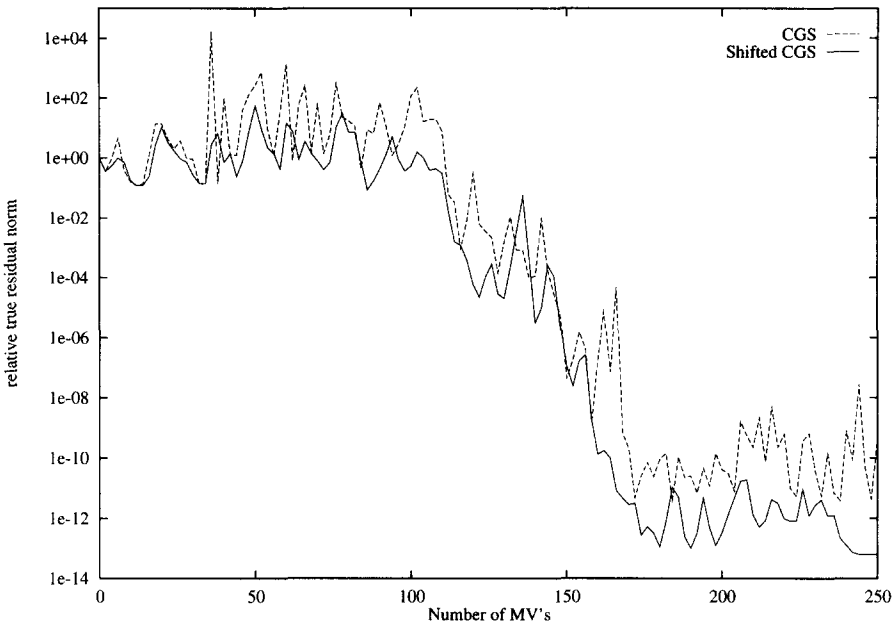

Fig. 4. Convergence history for example 2, 7840 nonzeros, (ILU(0) preconditioning).

converge for this subproblem, this strategy is repeated up to 5 times (after which CURRY gives up).

Figs. 5 and 6 show the convergence behavior of CURRY in combination with CGS2, CGS, or Bi-CGSTAB (all with ILU(0) preconditioning), as a linear solver in the consecutive Newton steps. Actually, for reasons explained in the introduction of this paper, CURRY, in the case of Bi-CGSTAB, switches to CGS in the last step of the Newton process for a particular continuation step. This improves the overall convergence of the continuation method significantly.

The figures should be understood as follows. The vertical axis shows the value of the continuation parameter, dictated by the continuation strategy in CURRY. The horizontal axis shows the cumulative number of matrix multiplications (MVs) used by the linear solver. The simulation is a success when the bottom of the plot is reached. The execution time is proportional to the number of MVs, so the fewer MVs the better the performance.

Fig. 5 shows the convergence behavior of a transient phase simulation (from 3 to 4 V) in Time for the device CAPO1T. With all three choices of the linear solvers CGS2, CGS, and Bi-CGSTAB+ (the "+" indicating the switch to CGS), the package CURRY manages to compute the solution of a full simulation. Clearly, CGS2 is the method of choice in this example. Observe the long stagnation phase between 10^{-9} and 10^{-8} s. Typically, in other transient simulations (not shown here) with similar stagnation phases, CGS2 improved the overall performance of CURRY significantly.

Fig. 6 shows the convergence behavior of a simulation for the voltage (from 0 to 40 V) of the device DIODE. The plot shows a typical convergence behavior of CURRY for simulation of "difficult" devices for which the combination of CURRY and CGS fails, e.g., in this case almost immediately (and therefore hard to see in the plot). The combination with Bi-CGSTAB+ converges initially

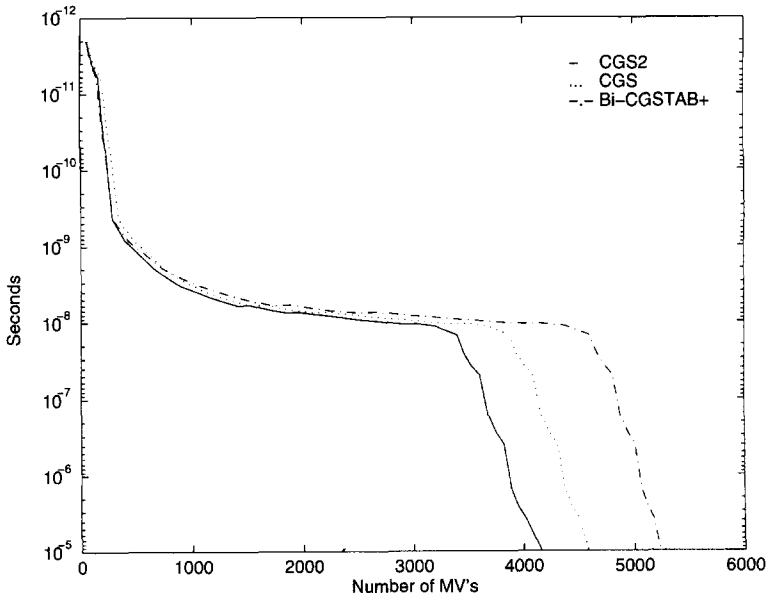

Fig. 5. CAPO1T: Convergence behavior of CURRY in combination with CGS2, CGS, and Bi-CGSTAB+.

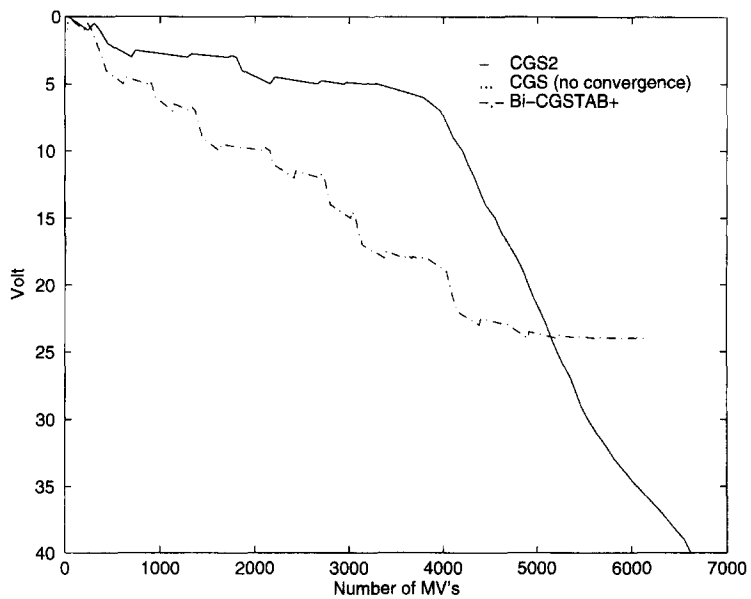

Fig. 6. DIODE: Convergence behavior of CURRY in combination with CGS2, CGS, and Bi-CGSTAB+.

much better than the combination with CGS2, but stalls at voltage level 23 and fails. CURRY with CGS2, on the other hand, after some difficulties in the initial phase, converges quite rapidly. In this example, and also in others, only the combination of CURRY with CGS2 is able to compute the solution of a full simulation.

7.4. Shifted CGS as linear solver in a Newton scheme

Here we present a comparison of shifted CGS and other linear solvers in a Newton scheme for solving the classical *driven cavity* problem from incompressible fluid flow. We compare shifted CGS, CGS2, CGS, Bi-CGSTAB, and BiCGstab(2). In step k of the Newton process, the linear system (involving the exact Jacobian), is solved to a relative residual norm reduction of 2^{-k} (see [4]), subject to a maximum of 100 matrix-vector multiplications. The correction vector obtained in this way was then used in a linesearch procedure [5] to get the new approximation. If the relative change of the (Newton) residual was less than 10^{-6} the iterations were stopped.

Following closely the presentations in [8,2] the driven cavity problem in stream function-vorticity formulation is described by these equations:

$$v\Delta\omega + (\psi_{x_2}\omega_{x_1} - \psi_{x_1}\omega_{x_2}) = 0 \quad \text{in } \Omega,$$

$$-\Delta\psi = \omega \quad \text{in } \Omega,$$

$$\psi = 0 \quad \text{on } \partial\Omega,$$

$$\frac{\partial\psi}{\partial n}(x_1,x_2)_{|\partial\Omega} = \begin{cases} 1 & \text{if } x_2 = 1, \\ 0 & \text{if } 0 \leqslant x_2 < 1, \end{cases}$$

where Ω is the unit square and the viscosity v is the reciprocal of the Reynolds number Re. In terms of ψ alone this can be written as

$$v\Delta^2\psi + (\psi_{x_2}(\Delta\psi)_{x_1} - \psi_{x_1}(\Delta\psi)_{x_2}) = 0 \quad \text{in } \Omega,$$

subject to the same boundary conditions. This equation was discretized with central differences on a 41×41 regular grid. As preconditioner we used the Modified ILU(2) decomposition [9] of the biharmonic operator Δ^2. As initial guess we took $\psi = 0$.

In Fig. 7 we show a plot of the convergence of Newton for Re=1000. The marks indicate a Newton step. Also in this example, the parameter μ in Shifted CGS was taken as the inverse of the real part of the largest eigenvalue estimate, delivered by a few steps of Arnoldi's algorithm.

As can be seen clearly, only the combination of Newton with Shifted CGS and CGS2 is successful in this example. The combination with CGS can keep up in the beginning but then CGS has trouble solving the linear system, which causes the stagnation. The combination with Bi-CGSTAB stagnates all together. This could be attributed to the fact that Bi-CGSTAB is not able to solve the linear systems, because they are very nonsymmetric [20]. BiCGstab(2) on the other hand is able to solve the linear systems (in the beginning) but apparently delivers a correction that is not of much use to Newton.

Note that in this case the combination of Newton with shifted CGS is preferable, because it is more efficient: shifted CGS uses two inner products less than CGS2.

Additional note. In [25] it was observed that replacing the residual $\boldsymbol{r}_k$ in CGS by the true residual $b - A\boldsymbol{x}_k$ has a negative effect on the iteration process. Recently, it came to our attention that

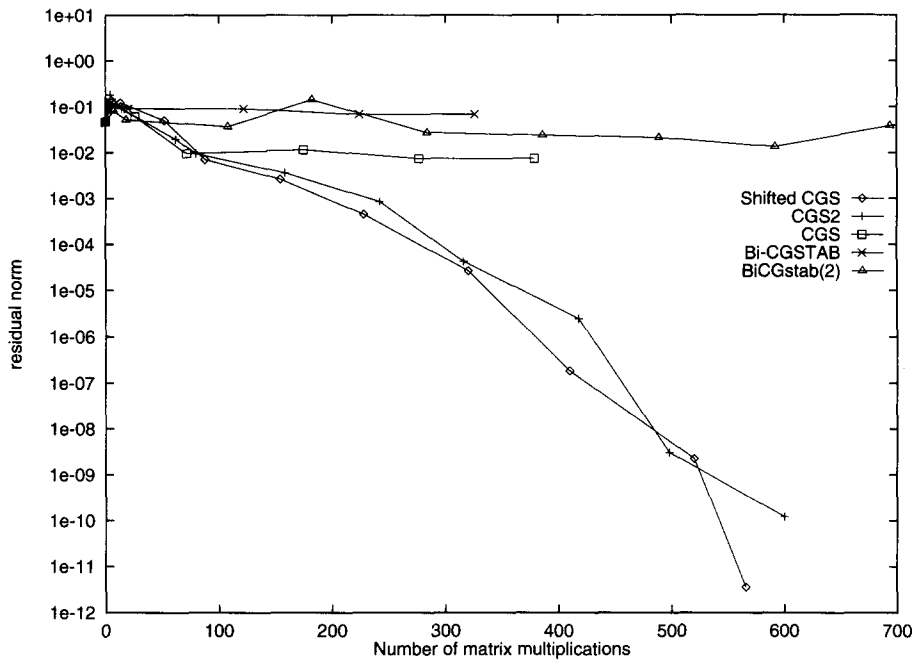

Fig. 7. Driven Cavity: Convergence behavior of Newton in combination with different linear solvers.

Neumaier [16] reports good results with a different strategy that does include the use of the true residual. His strategy can be summarized as follows:

Add the line "$x_{\text{best}} = 0$" after the first line in CGS
and replace the last line with
$r_{k+1} = b - Ax_{k+1}$
if $\|r_{k+1}\|_2 \leq \|r_{\text{best}}\|_2$ then
$b = r_{\text{best}} = r_{k+1}$
$x_{\text{best}} = x_{\text{best}} + x_{k+1}$
$x_{k+1} = 0$
endif

We have tested this approach on several problems and for those problems we confirm the observation that indeed this modification to CGS has no adverse influence on the convergence behavior and that accuracy is reached within machine precision (i.e., $|\,\|r_m\|_2 - \|b - Ax_m\|_2| \lesssim \bar{\xi}\Gamma\|r_0\|_2$ with Γ as in (4)). For an explanation and other related strategies, see [22].

8. Conclusions

We have shown how the CGS algorithm can be generalized to a method that uses the product of two nearby Bi-CG polynomials as a reduction operator on the initial residual. Two methods

are suggested to improve the accuracy and the speed of convergence, without losing the quadratic reduction of errors in converged eigenvector directions. This is important, since the Newton process seems to benefit from this property. Several numerical examples are given that confirm our heuristic arguments.

Acknowledgements

We appreciated the help of Marjan Driessen at Philips Research Laboratories (Eindhoven). She provided the numerical data for the examples in Section 7.3.

References

[1] H.G. Brachtendorf, Simulation des eingeschwungenen Verhaltens elektronischer Schaltungen, Ph.D. Thesis, Universität Bremen, 1994.
[2] P.N. Brown and Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, *SIAM J. Sci. Statist. Comput.* **11** (1990) 450–481.
[3] T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto and C.H. Tong, A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems, *SIAM J. Sci. Comput.* **15** (1994) 338–347.
[4] R.S. Dembo, S.C. Eisenstat and T. Steihaug, Inexact Newton methods, *SIAM J. Numer. Anal.* **19** (1982) 400–408.
[5] J.E. Dennis, Jr. and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1983).
[6] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G.A. Watson, Ed., *Numerical Analysis Dundee 1975*, Lecture Notes in Mathematics, Vol. 506 (Springer, Berlin, 1976) 73–89.
[7] R. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, *SIAM J. Sci. Comput.* **14** (1993) 470–482.
[8] R. Glowinski, H.B. Keller and L. Reinhart, Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems, *SIAM J. Sci. Statist. Comput.* **6** (1985) 793–832.
[9] I. Gustafsson, A class of first order factorizations methods, *BIT* **18** (1978) 142–156.
[10] M.H. Gutknecht, Variants of BiCGStab for matrices with complex spectrum, *SIAM J. Sci. Comput.* **14** (1993) 1020–1033.
[11] Y. Huang and H.A. Van der Vorst, Some observations on the convergence behaviour of GMRES, Tech. Report 89-09, Delft University of Technology, Faculty of Tech. Math., 1989.
[12] C. Lanczos, Solution of systems of linear equations by minimized iteration, *J. Res. Nat. Bur. Standards* **49** (1952) 33–53.
[13] A. Liegmann, Efficient solution of large sparse linear systems, Ph.D. Thesis, ETH Zürich, 1995.
[14] 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) 148–162.
[15] N.M. Nachtigal, S.C. Reddy and L.N. Trefethen, How fast are nonsymmetric matrix iterations?, *SIAM J. Matrix Anal. Appl.* **13** (1992) 778–795.
[16] A. Neumaier. Oral presentation at the Oberwolfach meeting, April 1994.
[17] S.J. Polak, C. den Heijer, W.H.A. Schilders and P. Markowich, Semiconductor device modelling from the numerical point of view, *Internat. J. Numer. Methods Engng.* **24** (1987) 763–838.
[18] C. Pommerell and W. Fichtner, PILS: An iterative linear solver package for ill-conditioned systems, in: *Supercomputing '91* (Albuquerque, NM, November 1991) (IEEE Computer Society Press, Los Alamitos, CA) 588–599.
[19] A. Ruhe, Rational Krylov algorithms for nonsymmetric eigenvalue problems II. Matrix Pairs, *Linear Algebra and its Appl.* **197/198** (1994) 283–295.
[20] G.L.G. Sleijpen and D.R. Fokkema, BiCGstab(ℓ) for linear equations involving matrices with complex spectrum, *Electron. Trans. Numer. Anal.* **1** (1993) 11–32.

[21] G.L.G. Sleijpen and H.A. Van der Vorst, Maintaining convergence properties of BiCGstab methods in finite precision arithmetic, *Numer. Algorithms* **10** (1995) 203–223.
[22] G.L.G. Sleijpen and H.A. Van der Vorst, Reliable updated residuals in hybrid Bi-CG methods, *Computing* **56** (2) (1996), to appear.
[23] G.L.G. Sleijpen, H.A. Van der Vorst and D.R. Fokkema, BiCGstab(ℓ) and other hybrid Bi-CG methods, *Numer. Algorithms* **7** (1994) 75–109.
[24] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **10** (1989) 36–52.
[25] H.A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13** (1992) 631–644.
[26] L. Zhou and H.F. Walker, Residual smoothing techniques for iterative methods, *SIAM J. Sci. Comput.* **15** (1994) 297–312.