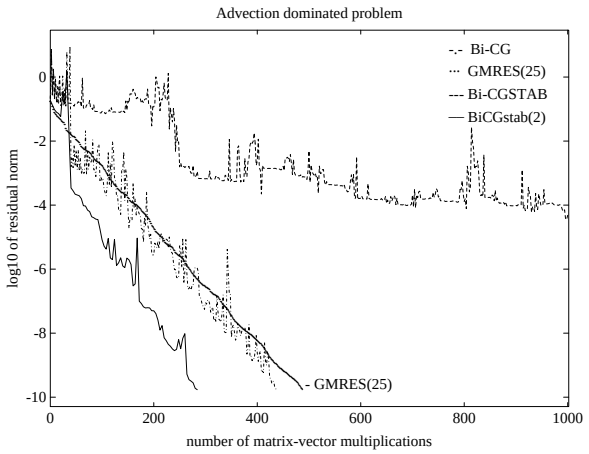

Universiteit Utrecht

Department
of
Mathematics

Hybrid Bi-Conjugate Gradient Methods for CFD Problems

by

Gerard L.G. Sleijpen and Henk A. Van der Vorst

Preprint

nr. 902

February, 1995

HYBRID BI-CONJUGATE GRADIENT METHODS FOR CFD PROBLEMS

Gerard L.G. Sleijpen* and **Henk A. Van der Vorst***

Abstract: In recent years Krylov subspac iteration methods have become very popular for the solution of discretized PDE-systems. Among these methods are CG, GMRES, Bi-CG, and CGS. More recently we see hybrid methods that can be regarded as combinations of standard Krylov subspace methods such as Bi-CG and GMRES. One of the first hybrid schemes of this type is Bi-CGSTAB (a combination of Bi-CG and GMRES(1)), other examples are Hybrid GMRES (polynomial preconditioned GMRES), the nested GMRESR method (GMRES preconditioned by itself or other schemes), and BiCGstab(ℓ). These methods have been successful in solving relevant sparse nonsymmetric linear systems, but there is still a need for further improvements. In this paper we will concentrate on hybrid Bi-CG methods and we will highlight some of the recent advancements in the search for effective iterative solvers, specially in view of CFD applications.

Key words: Iterative solvers, Bi-CG, QMR, Bi-CGSTAB, CGS, GMRES, BiCGstab(ℓ)

1 INTRODUCTION

Discretization of partial differential systems, by finite difference, or finite element methods, usually leads to very large sparse linear systems. In advection-diffusion problems these systems are unsymmetric. Especially in 3-dimensional problems it is quite unattractive to solve such systems with direct methods, because of the huge CPU-times and the large demands on memory space. As an alternative one can resort to iterative solution methods. The problem, however, is that it requires skill and expertise to select and tune the proper iterative method.
The most popular methods at the moment are the so-called Krylov subspace methods, which include methods like GMRES [1], CG [2], Bi-CG [3], CGS [4], QMR [5], and Bi-CGSTAB [6]. We will discuss these methods briefly, and then our attention will be focussed on variants of Bi-Conjugate Gradients (Bi-CG) methods in combination with low degree GMRES methods. Bi-CG is attractive because of short recursions, but it does not minimize the residual in a reasonable norm, while GMRES minimizes the residual over the search space, at the cost of long (expensive) recursions. A good example of a hybrid Bi-CG method is the Bi-CGSTAB method [6]. This latter method can be viewed as a combination of Bi-CG and repeated GMRES(1).

There are two different conditions under which Bi-CG can break down, one of which can be removed by a look-ahead strategy, the other one can be removed by solving the reduced system in a different way, as in QMR [5, 7]. A weak point in Bi-CGSTAB is that it introduces one more breakdown possibility on top of these, namely when the GMRES(1) part, part, of the algorithm stagnates. This may happen, for instance in advection dominated PDE–problems. The obvious way to overcome this problem is to combine Bi-CG with GMRES(2): BiCGstab(2) [8] (for a different implementation, see [9]). In this way a very competitive Bi-CGSTAB variant is obtained, which is easy to implement. This is a particular instance of the broader family of BiCGstab(ℓ) methods [8]. There are several choices for the implementation of these methods [10], and the differences in these implementations become practically important for larger values of ℓ, say $\ell > 4$. The additional (near) breakdown situation in Bi-CGSTAB methods has been studied in depth by Sleijpen et al [11], and a simple strategy is proposed to avoid such situations. In many relevant cases this has the effect of improving the convergence properties of the method.

Finally, we will pay some attention to the problem of accurately updating the residual vectors. It appears in many practical problems that the up-

*Mathematical Institute, Utrecht University, P.O.Box 80.010, 3508 TA UTRECHT, the Netherlands, E-mail: sleijpen@math.ruu.nl, vorst@math.ruu.nl

dated residual vector may be very different from the residual that one computes for the obtained approximation for the solution. The obvious way to control this, namely by computing the actual residual in each iteration step, is very expensive since it requires one more matrix vector operation (which is usually the CPU-dominant part of an iteration). We will suggest simple strategies for the accurate updating of the approximations for the solution as well as for the updating of the corresponding residual.

The tuning of iterative methods is usually known under the name of preconditioning. We will touch this aspect only briefly, and refer to literature for more information.

2 KRYLOV SUBSPACE METHODS

A very basic idea, that leads to many effective iterative solvers, is to to split the matrix of a given linear system in the sum of two matrices, one of which a matrix that would have led to a system that can easily be solved. The most simple splitting we can think of is $A = I-(I-A)$. Given the linear system $Ax = b$, this splitting leads to the well-known Richardson iteration:

$$x_{i+1} = b + (I - A)x_i = x_i + r_i.$$

Multiplication by $-A$ and adding b gives

$$b - Ax_{i+1} = b - Ax_i - Ar_i$$

or

$$r_{i+1} = (I - A)r_i = (I - A)^{i+1}r_0 = P_{i+1}(A)r_0,$$

or, in terms of the error

$$A(x - x_{i+1}) = P_{i+1}(A)A(x - x_0)$$

$$\Rightarrow \quad x - x_{i+1} = P_{i+1}(A)(x - x_0).$$

In these expressions P_{i+1} is a (special) polynomial of degree $i + 1$. Note that $P_{i+1}(0) = 1$.
Results obtained for the standard splitting can be easily generalized to other splittings, since the more general splitting $A = M-N = M-(M-A)$ can be rewritten as the standard splitting $B = I-(I-B)$ for the preconditioned matrix $B = M^{-1}A$. The theory of matrix splittings, and the analysis of the convergence of the corresponding iterative methods, is treated in depth in [12]. We will not discuss this aspect here, since it is not relevant at this stage. Instead of studying the basic iterative methods we will show how other more powerful iteration methods can be constructed as accelerated versions of the basic iteration methods. In the context of these accelerated methods, the matrix splittings become important in another way, since the matrix M of the splitting is often used to *precondition* the given system. That is, the iterative method is applied to, e.g., $M^{-1}Ax = M^{-1}b$. We will come back to this later.

From now on we will assume that $x_0 = 0$. This too does not mean a loss of generality, for the situation $x_0 \neq 0$ can through a simple linear transformation $z = x - x_0$ be transformed to the system

$$Az = b - Ax_0 = \widetilde{b}$$

for which obviously $z_0 = 0$.

For the simple Richardson iteration it follows that

$$x_{i+1} = r_0 + r_1 + r_2 + \cdots + r_i = \sum_{j=0}^{i}(I - A)^j r_0$$

$$\in \text{span}(r_0, Ar_0, \ldots, A^i r_0) = \mathcal{K}_{i+1}(A; r_0).$$

The subspace $\mathcal{K}_{i+1}(A; r_0)$ is called the *Krylov subspace* generated by A and r_0 of dimension $i + 1$. Apparently, the Richardson iteration delivers elements of increasing Krylov subspaces. Including local iteration parameters in the iteration would lead to other elements of the same Krylov subspaces. Let us write such an element still as x_{i+1}. Since $x_{i+1} \in \mathcal{K}_{i+1}(A; r_0)$, we have that

$$x_{i+1} = Q_{i+1}(A)r_0,$$

with Q_{i+1} an arbitrary polynomial of degree $i+1$. It follows that

$$\begin{aligned} r_{i+1} &= b - Ax_{i+1} = (I - AQ_{i+1}(A))r_0 \\ &= \widetilde{P}_{i+1}(A)r_0, \end{aligned} \tag{2.1a}$$

and, just as in the standard Richardson iteration, $\widetilde{P}_{i+1}(0) = 1$.
The Richardson iteration can be characterized by the polynomial $P_{i+1}(A) = (I - A)^{i+1}$.

The modern Krylov subspace methods all generate (implicitly) more optimal polynomials (e.g., orthogonal polynomials). These polynomials, although never explicitly formed, help us to understand the behavior of especially the hybrid Bi-CG methods.

2.1 Nonsymmetric Problems:

There are essentially three different ways to solve unsymmetric linear systems with Krylov subspace methods, while maintaining some kind of orthogonality between the residuals:

1. Solve the normal equations $A^T Ax = A^T b$ with conjugate gradients. We will not discuss this approach, since it often leads to unacceptably slow convergence in CFD situations. The reason for this is that the convergence of Conjugate Gradients depends on the condition number: the number of iterations to obtain a prescribed accuracy is roughly proportional to the square root of the condition number. Note, however, that the condition number of $A^T A$ is the square of the condition number of A, and this helps to explain the slow convergence. In some other applications it is a very attractive method though. For a discussion and for stable algorithms see Paige and Saunders [13].

2. Create explicitly an orthogonal basis for the Krylov subspace. This can be done by orthogonalizing the residual vectors, which leads to FOM [14] (or GENCG, ORTHORES, etc.). The more popular approach is to construct approximations in the Krylov subspace for which the norm of the residual is minimal, which leads to GMRES [1] (or GCR, GENCR, ORTHOMIN, ORTHODIR, etc.), which comes down to finding an orthogonal basis for a suitable Krylov subspace. The construction of a complete orthogonal basis is increasingly expensive in terms of computational costs and storage. We will discuss this approach further in Sect. 3.

3. Construct a basis for the Krylov subspace by a 3-term bi-orthogonality relation for $\mathcal{K}_i(A; r_0)$ and $\mathcal{K}_i(A^T; \widehat{r}_0)$, which is attractive since it leads to only moderate overhead in terms of computation and memory. This approach leads to the Bi-CG methods, to be discussed in Section 4. It turns out that these methods allow for further improvement, which can make them very competitive. However, this further improvement has to be done with great care. The main part of this paper (Sections 5–8) will be devoted to this subject.

3 GMRES AND FOM

For the determination of suitable approximations in the Krylov subspace it is attractive to have an orthogonal basis for that subspace. If A is symmetric then this can be done with an inexpensive 3-term recurrence relation (it suffices to make a new basis vector orthogonal to the last two basis vectors), but in the unsymmetric case a new basis vector has to be made explicitly orthonormal with respect to all the previous vectors:

$$v_1 = \frac{1}{\|r_0\|} r_0,$$

$$h_{i+1,i} v_{i+1} = Av_i - \sum_{j=1}^{i} h_{j,i} v_j .$$

The $h_{j,i}$ follow from the orthogonality conditions, and $h_{i+1,i}$ is determined so that $\|v_{i+1}\| = 1$ (here, as elsewhere, $\|\cdot\|$ is the Euclidean norm).
Note that this orthogonalization becomes increasingly expensive per iteration step, since the computation of each $h_{j,i}$ requires an inner product.
The orthogonal basis can be exploited in two different ways. The orthogonality relation can be rewritten in matrix notation as

$$AV_i = V_i H_{i,i} + h_{i+1,i} v_{i+1} e_i^T , \tag{3.1a}$$

with $V_i = [\, v_1 \,|\, v_2 | \ldots | \, v_i]$ (the matrix of which the j-th column is v_j), and $H_{i,i}$ an i by i upper Hessenberg matrix.
If we want to have the element x_i from the current Krylov subspace, spanned by $v_1, \ldots, v_i$, for which the residual $b - Ax_i$ is orthogonal to that subspace then we should have that

$$V_i^T (b - Ax_i) = 0. \tag{3.1b}$$

Note that x_i can be written as $x_i = V_i y$, for a vector y of length i. If we insert this expression for x_i in (3.1b), then together with (3.1a) we get

$$V_i^T (b - V_i H_{i,i} y) = 0. \tag{3.1c}$$

Since $V_i^T V_i$ is the unit matrix of dimension i, it follows that y is the solution of

$$H_{i,i} y = V_i^T b,$$

and this is a small system that can be solved relatively easily (if not singular). The methods FOM [14], ORTHORES [15], and GENCG [16], are implementations of this approach.

Another and more robust approach is to rewrite the orthogonality relations as

$$AV_i = V_{i+1} H_{i+1,i}, \tag{3.1d}$$

in which $H_{i+1,i}$ is an $i+1$ by i upper Hessenberg matrix. Now we try to find the x_i in the Krylov

subspace spanned by $v_1, \ldots, v_i$, for which $\|b - Ax_i\|$ is minimal.
This problem is equivalent with finding the vector z, of length i, for which

$$\|V_{i+1}^T b - H_{i+1,i} z\| \text{ is minimal.}$$

This is a standard linear least squares problem that can be solved efficiently, specially if one exploits the upper Hessenberg structure of $H_{i+1,i}$. This approach is the basis for GMRES [1]; other, less robust, implementations are known under names like ORTHODIR [17], GCR [18], ORTHOMIN [19], etc.

In order to avoid excessive storage requirements and computational costs for the orthogonalization, GMRES is usually restarted after each m iteration steps. This algorithm is referred to as GMRES(m).

There is an interesting and simple relation between the two different Krylov subspace projection approaches (3.1a), the "FOM" approach, and (3.1d), the "GMRES" approach. The projected system matrix $H_{i+1,i}$ is transformed by a Givens rotations to an upper triangular matrix (with last row equal to zero). So, in fact, the major difference between FOM and GMRES is that in FOM the last $(i+1)$-th row is simply discarded, while in GMRES this row is rotated to a zero vector. Let us characterize the Givens rotation, acting on rows i and $i+1$, in order to zero the element in position $(i+1, i)$, by the sine s_i and the cosine c_i. Let us further denote the residuals for FOM with an superscript F and those for GMRES with superscript G. Then the above observations lead to the following results for FOM and GMRES [1, 20].

1. The reduction for successive GMRES residuals is given by

$$\frac{\|r_i^{\mathrm{G}}\|}{\|r_{i-1}^{\mathrm{G}}\|} = |s_i| \ . \tag{3.1e}$$

   (see Saad and Schultz [1]: p. 862, Proposition 1)

2. If $c_i \neq 0$ then the FOM and the GMRES residuals are related by

$$\|r_i^{\mathrm{G}}\| = |c_i| \ \|r_i^{\mathrm{F}}\| \tag{3.1f}$$

   (see Brown [20]: theorem 5.1)

From these relations we see that when GMRES has a local significant reduction in the norm of the residual (i.e., s_i is small), then FOM gives about the same result as GMRES (since $c_i^2 = 1 - s_i^2$). On the other hand when FOM has a breakdown ($c_i = 0$), then the GMRES does not lead to an improvement in the same iteration step. This observation will help us to improve convergence properties in hybrid Bi-CG methods that will be described later.

4 Bi-CG AND QMR

This class of methods arises from the attempt to construct a suitable set of basis vectors for the Krylov subspace by a three-term recurrence relation:

$$\alpha_{j+1} r_{j+1} = A r_j - \beta_j r_j - \gamma_j r_{j-1}. \tag{4.1a}$$

If A is symmetric then the coefficients in this recurrence relation can be determined so that the basis vectors are orthogonal. In the nonsymmetric case this is in general not the case (for a proof of this see Faber and Manteuffel [21]).
However, it can easily be shown that for nonsymmetric A (4.1a) can be used to generate a basis r_0, $\ldots$, r_{i-1} for $\mathcal{K}_i(A; r_0)$, such that $r_j - \mathcal{K}_j(A^T; \widehat{r}_0)$, where $\widehat{r}_0$ is an arbitrary vector.
If we let the basis vectors $\widehat{r}_j$ for $\mathcal{K}_i(A^T; \widehat{r}_0)$ satisfy the same recurrence relation as the vectors r_j, i.e., with identical recurrence coefficients, then we see that

$$(r_i, \widehat{r}_j) = 0 \quad \text{for} \quad i \neq j$$

(by a simple symmetry argument).
Hence, the sets $\{r_j\}$ and $\{\widehat{r}_j\}$ satisfy a *bi-orthogonality* relation.

If we consider the r_j as being the j-th column of the matrix

$$R_i = [\, r_0 \,|\, \ldots \,|\, r_{i-1} \,]$$

then the recurrence relation says that A applied to a column of R_i results in the combination of three successive columns, or

$$AR_i = R_i \begin{bmatrix} \ddots & \ddots & & & \\ \ddots & \ddots & \gamma_j & & \\ & \ddots & \beta_j & \ddots & \\ & & \alpha_{j+1} & \ddots & \ddots \\ & & & \ddots & \ddots \end{bmatrix} + \alpha_i \begin{bmatrix} 0 \,|\, 0 \,|\, \ldots \,|\, r_i \end{bmatrix}$$

or

$$AR_i = R_i T_{i,i} + \alpha_i r_i e_i^T, \qquad (4.1b)$$

in which $T_{i,i}$ is an i by i tridiagonal matrix and e_i is the ith unit vector in $\mathbb{R}^i$.

Now we use the matrix $\widehat{R}_i = [\widehat{r}_0 \,|\, \widehat{r}_1 \,|\, ... \,|\, \widehat{r}_{i-1}]$ for the projection of the system

$$\widehat{R}_i^T (Ax_i - b) = 0,$$

or

$$\widehat{R}_i^T A R_i y_i - \widehat{R}_i^T b = 0$$

(the so-called Galerkin conditions).
Using (4.1b) we find that y_i satisfies

$$\widehat{R}_i^T R_i T_{i,i} y_i = (r_0, \widehat{r}_0) e_1 .$$

Since $\widehat{R}_i^T R_i$ is a diagonal matrix with diagonal elements $(r_j, \widehat{r}_j)$, we find, if all these diagonal elements are nonzero, that

$$T_{i,i} y_i = e_1 \quad \Rightarrow \quad x_i = R_i y_i. \qquad (4.1c)$$

This method is known as the Bi-Lanczos method [22].
Obviously that we have problems when a diagonal element of $\widehat{R}_i^T R_i$ becomes (near) zero, this is referred to in literature as a serious (near) breakdown. The way to get around this difficulty is the so-called look-ahead strategy, which comes down to taking a number of successive basis vectors for the Krylov subspace together and to make them block-wise bi-orthogonal. This has been worked out in detail by many authors [5, 23, 24, 25, 26, 27]. Another way to avoid breakdown is to restart as soon as a diagonal element gets small. Of course, this strategy looks surprisingly simple, but one should realize that at a restart the Krylov subspace, that has been built up so far, is thrown away, which destroys possibilities for faster (i.e., superlinear) convergence. Furthermore, there is still a chance that also after restart a (near) breakdown situation arises.

The LU decomposition of the tridiagonal system (4.1c) can be updated from iteration to iteration and this leads to a recursive update of the solution vector. This avoids to save all intermediate r and $\widehat{r}$ vectors. This variant of Bi-Lanczos is usually called Bi-Conjugate Gradients, or shortly Bi-CG [3].
Of course one can in general not be certain that an LU decomposition (without pivoting) of the tridiagonal matrix $T_{i,i}$ exists, and this may lead also to breakdown of the Bi-CG algorithm. Note that this breakdown can be avoided in the Bi-Lanczos formulation of this iterative solution scheme. For an approach to avoid this second type of breakdown, see Bank and Chan [7], or Chan and Szeto [28]. It is also avoided in the QMR approach, that we will discuss later.
For symmetric matrices the Bi-Lanczos method generates the same solution as the Lanczos method, provided that $\widehat{r}_0 = r_0$, and under the same condition Bi-CG delivers the same iterates as CG for positive definite matrices. However, the bi-orthogonal variants do so at the cost of two matrix vector operations per iteration step, namely one with A for the construction of the 'regular' Krylov subspace, and one with A^T for the construction of the shadow subspace that is necessary for the bi-orthogonality conditions.

The QMR method [5] relates to Bi-CG in a similar way as GMRES relates to FOM. We can also rewrite the recurrence relations in

$$AR_i = R_{i+1} T_{i+1,i},$$

with

$$T_{i+1,i} = \begin{pmatrix} \ddots & \ddots & & & \\ \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \\ & & & \ddots & \end{pmatrix} \begin{matrix} \\ \uparrow \\ i+1 \\ \downarrow \\ \\ \end{matrix} \quad \begin{matrix} \leftarrow \; i \; \rightarrow \end{matrix}$$

Similar as for GMRES we would like to construct the x_i, with

$$x_i \in \mathrm{span}(r_0, Ar_0, \ldots, A^{i-1} r_0), \quad x_i = R_i z$$

for which

$$\begin{aligned} \|Ax_i - b\| &= \|AR_i z - b\| \\ &= \|R_{i+1} T_{i+1,i} z - b\| \\ &= \|R_{i+1} \left(T_{i+1,i} z - \|r_0\| e_1 \right) \| \end{aligned}$$

is minimal. However, in this case that would be quite an amount of work since the columns of R_{i+1} are not necessarily orthogonal. Freund and Nachtigal [5] suggest to solve the minimum norm least squares problem

$$\min_{z \in \mathbb{R}^i} \|T_{i+1,i} z - \|r_0\| e_1\|. \qquad (4.1d)$$

Here also, storage of all intermediate r and $\widehat{r}$ can be avoided. This leads to the simplest form of the QMR method. A more general form arises if the least squares problem (4.1d) is replaced by a weighted least squares problem. No strategies are yet known for optimal weights, however.
In full glory the QMR method is carried out on top of a look-ahead variant of the bi-orthogonal Lanczos method, which makes the method more robust [5]. Experiments suggest that QMR has a much smoother convergence behavior than Bi-CG, but it is not essentially faster than Bi-CG.

5 CGS

The residuals $r_j = b - Ax_j$ and $\widehat{r}_j$ in the Bi-Conjugate Gradient method can be written formally as $r_j = P_j(A)r_0$ and $\widehat{r}_j = P_j(A^T)\widehat{r}_0$, where P_j is a polynomial of degree j, and because of the bi-orthogonality relation we have that

$$(r_j, \widehat{r}_i) = (P_j(A)r_0, P_i(A^T)\widehat{r}_0)$$

$$= (P_i(A)P_j(A)r_0, \widehat{r}_0) = 0,$$

for $i < j$.
The iteration parameters for Bi-CG are computed from inner products like the above. Sonneveld [4] observed that we can also construct the vectors $\widetilde{r}_j = P_j^2(A)r_0$, using only the latter form of the inner product for recovering the Bi-CG parameters (which implicitly define the polynomial P_j). By doing so, it can be avoided that the vectors $\widehat{r}_j$ have to be formed, nor is there any multiplication with the matrix A^T.
The resulting CGS [4] method works in general very well for many unsymmetric linear problems. It converges often much faster than Bi-CG (about twice as fast in some cases) and does not have the disadvantage of having to store an increasing number of vectors like in GMRES. These three methods have been compared in many studies [29, 30, 31, 32].
A problem with CGS is that it usually shows a very irregular convergence behavior. This behavior can even lead to cancellation and a spoiled solution [6]. We will address this issue in Section 6.

The preconditioned CGS algorithm for solving the linear system $Ax = b$, with preconditioning M reads as in Algorithm 1; see also, Dongarra et al [33] or Barrett et al [34].

The matrix M in this scheme represents the preconditioning matrix and the way of preconditioning. The above scheme in fact carries out the CGS procedure for the explicitly postconditioned linear system

$$AM^{-1}y = b,$$

but the vectors y_i and the residual have been back transformed to the vectors x_i and r_i corresponding to the original system $Ax = b$.

For a transpose-free form of QMR, namely TF-QMR, which comes down to a quasi-minimum residual approach for CGS, see Freund [35].

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\widehat{r}_0$ is an arbitrary vector, such that
$(r_0, \widehat{r}_0) \neq 0$, e.g., $\widehat{r}_0 = r_0$;
$\rho = 1$; $u = p = 0$;
for $i = 0, 1, 2, \ldots$
$\rho' = -\rho$;
$\rho = (r_i, \widehat{r}_0)$; $\beta = \rho/\rho'$;
$s = r_i - \beta p$;
$u = s - \beta(p - \beta u)$;
Solve $\widehat{p}$ from $M\widehat{p} = u$; $v = A\widehat{p}$;
$\gamma = (v, \widehat{r}_0)$; $\alpha = \rho/\gamma$;
$p = s - \alpha v$;
Solve $\widehat{p}$ from $M\widehat{p} = p + s$;
$x_{i+1} = x_i + \alpha\widehat{p}$;
$v = A\widehat{p}$; $r_{i+1} = r_i - \alpha v$;
if x_{i+1} is accurate enough then quit;
endfor

Alg.1: The CGS algorithm.

6 HOW SERIOUS IS IRREGULAR CONVERGENCE?

By very irregular convergence we refer to the situation where successive residual vectors in the iterative process differ in orders of magnitude in norm, and some of these residuals may be even much bigger in norm than the initial residual. In rounded arithmetic, where the actual computational work is done, this is a point of concern even if eventually the (updated) residual satisfies a given tolerance. We will try sketch the reason for this (for more details we refer to Sleijpen et al [11, 10, 36]).

Evaluation errors are introduced in any of the computational steps. These errors may affect

(i) the speed of convergence and
(ii) the numerical accuracy.

We will first discuss these effects and then, in a later section, we will propose iteration methods

that suffer less from these effects and which will have better convergence properties.

6.1 Speed of Convergence:

Bi-CG and, also, hybrid methods as CGS that rely on Bi-CG, are based on bi-orthogonality: as explained in Section 4, the Bi-CG residuals r_j are orthogonal to $\widehat{r}_i$, for $i < j$, in exact arithmetic. In rounded arithmetic, global bi-orthogonality (that is, $r_j \perp \widehat{r}_i$, also for i much smaller than j) can only be maintained at relatively high costs by selective re-orthogonalization. Surprisingly, as experiments indicate, errors do not seem to affect the convergence too much as long as the residuals are *locally* almost bi-orthogonal: we may expect good convergence if r_j and $\widehat{r}_{j-1}$ are almost orthogonal [11, 24, 37, 38]. However, if successive residuals differ considerably in magnitude, even local bi-orthogonality can be lost: although, the recurrence relation for the computation of r_j will be perturbed by an error that is relatively small with respect to r_{j-1} if $\|r_{j-1}\| \gg \|r_j\|$, the error perturbs r_j and will be relatively large with respect to this residual. Consequently, r_j will not be (almost) orthogonal to $\widehat{r}_{j-1}$. Of course, such a perturbation may be harmless if r_j is close to the required tolerance, but in other cases, it deteriorates the convergence, and it may even lead to stagnation.

6.2 Numerical Accuracy:

In exact arithmetic the updated r_j would be equal to $b - Ax_j$, but in rounded arithmetic it is unavoidable that differences between r_j and $b - Ax_j$ arise. We will say that an algorithm is *accurate* for a certain problem if the *updated residual* r_j and the *true residual* $b - Ax_j$ are of comparable size for the j's of interest.

The best we can hope for is that for each j the error in the residual is only the result of applying A to the update w_{j+1} for x_j in finite precision arithmetic:

$$\begin{aligned} &r_{j+1} = r_j - Aw_{j+1} - \Delta_A w_{j+1} \\ &\quad \text{if} \quad x_{j+1} = x_j + w_{j+1}, \quad \text{for each} \quad j, \end{aligned} \tag{6.2a}$$

where Δ_A is an $n \times n$ matrix for which $|\Delta_A| \leq n_A \overline{\xi}\,|A|$: n_A is the maximum number of non-zero matrix entries per row of A, $\overline{\xi}$ is the relative machine precision, the inequality $\leq$ and the absolute value $|\cdot|$ are to be taken element-wise. In the Bi-CG type methods that we consider, we compute explicitly the update Aw_j for the residual r_j from the update w_j for the approximation x_j by matrix multiplication: for this part, (6.2a) describes well the local deviations caused by evaluation errors.

In the "ideal" case (i.e. situation (6.2a) whenever we update the approximation) we have that

$$\begin{aligned} r_i - (b - Ax_i) &= \sum_{j=1}^{i} \Delta_A w_j \\ &= \sum_{j=1}^{i} \Delta_A (f_{j-1} - f_j), \end{aligned} \tag{6.2b}$$

where the perturbation matrix Δ_A may depend on j and f_j is the approximation error in the jth approximation: $f_j \equiv x - x_j$. Hence,

$$\begin{aligned} &\big|\|r_i\| - \|b - Ax_i\|\big| \\ &\quad \leq \quad 2\, n_A\, \overline{\xi}\, \big\||A|\big\| \sum_{j=0}^{i} \|f_j\| \qquad (6.2c) \\ &\quad \leq \quad 2\, \Gamma\, \overline{\xi} \sum_{j \leq i} \|r_j\| \qquad (6.2d) \\ &\quad \text{where} \quad \Gamma \equiv n_A \big\||A|\big\| \big\|A^{-1}\big\|. \end{aligned}$$

The first upper-bound (6.2c) and, except for the factor Γ, the second upper-bound (6.2d) appear to be rather sharp. In practise, a value for Γ of order 1 seems to be more appropriate (for a possible explanation of this, see [36]). We see that approximations with large approximation errors may ultimately lead to an inaccurate result. Such large local approximation errors are typical for CGS (cf. [6]). If there are a number of approximations with comparable large approximation errors, then their multiplicity times the norm of this error may replace the sum, otherwise it will be only the largest approximation error that makes up virtually the bound for the deviation.

Example. Figure 1 illustrates nicely the loss of accuracy as described above; for other examples, cf. [6]. The convergence history of the updated residuals (the 'circles': $\circ\circ$) and the true residuals (the solid curve: ——) of CGS is given for the matrix SHERMAN4 from the Harwell-Boeing set of test matrices. Here, as in other figures, the norm of the residuals, on $\log_{10}$-scale, is plotted (along the vertical axis) against the number of matrix-vector multiplications (along the horizontal axis). The dotted curve ($\cdots$) represents the estimated inaccuracy: $2\,\overline{\xi} \sum_{j \leq i} \|r_j\|$ (here with $\Gamma = 1$; cf. (6.2d)).

We will discuss two approaches that lead to a smoother convergence.

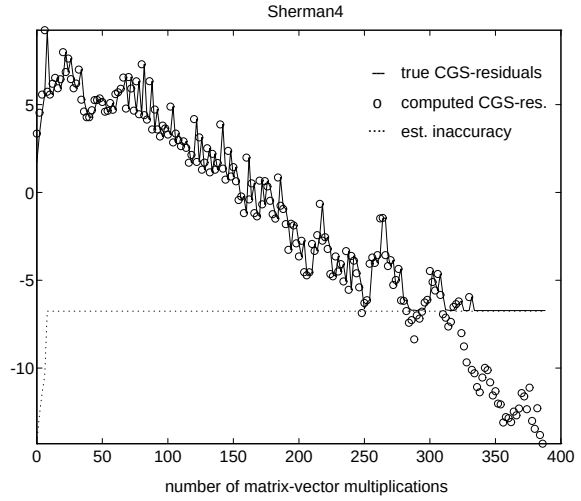

Fig.1: Convergence plot CGS for the true residuals and the updated residuals.

— Approaches to obtain the smoothing effect by adding a few lines to existing codes leave the speed of convergence essentially unchanged. One of these approaches leads to optimal accurate approximations [36] and will be discussed in Section 8. For other ones, we refer to the literature (e.g., [39]).
— In the next section, we concentrate on techniques that really change the convergence: they smooth down and speed up the convergence, and lead to more accurate approximations, all at the same time.

7 Bi-CGSTAB AND BiCGstab(ℓ)

Bi-CGSTAB [6] and other related methods are based on the following observation. Instead of squaring the Bi-CG polynomial, as in CGS, we can construct other iteration methods, by which x_i are generated so that $r_i = \widetilde{P}_i(A)P_i(A)r_0$ with other i^{th} degree polynomials $\widetilde{P}_i$ that have value 1 at 0. In this way, we hope to find methods with better convergence properties.

7.1 Bi-CGSTAB:

An obvious possibility is to take for $\widetilde{P}_i$ a polynomial of the form

$$Q_i(t) = (1-\omega_1 t)(1-\omega_2 t)\cdots(1-\omega_i t), \quad (7.1a)$$

and to select suitable constants ω_j. This expression leads to an almost trivial recurrence relation for the Q_i.

In Bi-CGSTAB ω_j in the j^{th} iteration step is chosen as to minimize r_j, with respect to ω_j, for residuals that can be written as $r_j = Q_j(A)P_j(A)r_0$. The preconditioned Bi-CGSTAB algorithm for solving the linear system $Ax = b$, with preconditioning M reads as in Algorithm 2. Compared

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\widehat{r}_0$ is an arbitrary vector, such that
$(r_0, \widehat{r}_0) \neq 0$, e.g., $\widehat{r}_0 = r_0$;
$\rho = \alpha = \omega = 1$; $v = p = 0$;
for $i = 0, 1, 2, \ldots$
$\rho' = -\omega\rho$;
$\rho = (r_i, \widehat{r}_0)$; $\beta = \alpha\rho/\rho'$;
$p = r_i - \beta(p - \omega v)$;
Solve $\widehat{p}$ from $M\widehat{p} = p$; $v = A\widehat{p}$;
$\gamma = (v, \widehat{r}_0)$; $\alpha = \rho/\gamma$;
$s = r_i - \alpha v$;
Solve $\widehat{s}$ from $M\widehat{s} = s$; $t = A\widehat{s}$;
$\omega = (s, t)/(t, t)$;
$x_{i+1} = x_i + \alpha\widehat{p} + \omega\widehat{s}$;
$r_{i+1} = s - \omega t$;
if x_{i+1} is accurate enough then quit;
endfor

Alg.2: The Bi-CGSTAB algorithm.

with CGS two extra inner products need to be calculated.

In the Bi-CGSTAB algorithm, Algorithm 2, two iteration coefficients α and β are computed in each sweep. We will discuss the effects of rounding errors on these values, in Section 7.3; see also [10, 11].

Bi-CGSTAB can be regarded as the product of Bi-CG and a repeated steepest descent method (GCR(1)). Of course, other product methods can be formulated as well. Gutknecht [9] has proposed BiCGStab2, which is constructed as the product of Bi-CG and GCR(2) (i.e. two steps of the Generalized Conjugate Residual method; cf. Section 7.2). It is also possibly to combine QMR with a steepest descent method, which leads to methods like QMRCGSTAB [40].

7.2 BiCGstab(ℓ):

A weak point in Bi-CGSTAB is that we have breakdown if an ω_j is equal to zero. One may also expect negative effects when ω_j is small. In fact, Bi-CGSTAB can be viewed as the combined effect of Bi-CG and GCR(1), or GMRES(1), steps. As

soon as the GCR(1) part of the algorithm (nearly) stagnates, then the Bi-CG part in the next iteration step cannot (or only poorly) be constructed. Another aspect of Bi-CGSTAB is that the factor Q_i has only real roots by construction. However, it is well-known that optimal reduction polynomials for matrices with complex eigenvalues may have complex roots as well. If, for instance, the matrix A is real skew-symmetric, then GCR(1) stagnates forever, whereas a method like GCR(2) (or GMRES(2)), in which we minimize over two combined successive search directions, may lead to convergence, and this is mainly due to the fact that then complex eigenvalue components in the error can be effectively reduced as well. For skew-symmetric matrices BiCGStab2 [9] will also break down, since the GCR(2) steps are constructed afterwards from two GCR(1) steps, one of which already failed to further expand the subspace.
Sleijpen and Fokkema [8] suggested how to combine Bi-CG with GMRES(ℓ), for $\ell \geq 1$. In this approach the polynomial Q is constructed right away as a product of ℓ degree factors, without ever constructing lower degree factors. As a result the new method BiCGstab(ℓ) leads only to meaningful residuals at each ℓ-th step, and the other steps do not necessarily lead to useful approximations. The main idea is that ℓ successive Bi-CG steps are carried out, where for the sake of an A^T-free construction the already available part of Q is expanded by simple powers of A. This means that after the Bi-CG part of the algorithm vectors from the Krylov subspace s, As, A^2s, ..., $A^\ell s$, with $s = Q_{i-\ell}(A)P_i(A)r_0$ are available, and then it is relatively easy to minimize the residual over that particular Krylov subspace. There are variants of this approach in which more stable bases for the Krylov subspaces are generated [10], but for low values of ℓ a standard basis satisfies, together with a minimum norm solution obtained through solving the associated normal equations (which requires the solution of an ℓ by ℓ system). In most cases BiCGstab(2) already will give nice results for problems where Bi-CGSTAB or BiCGStab2 may fail. Note, however, that, in exact arithmetic, if no breakdown situation occurs, BiCGStab2 would produce exactly the same results as BiCGstab(2) at the even-numbered steps. For $\ell = 1$ we obtain BiCGstab(1), which is the same as Bi-CGSTAB.

BiCGstab(2) can be represented by Algorithm 3.

For more general BiCGstab(ℓ) schemes, as well as discussions on their implementations, see [8, 10].

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\widehat{r}_0$ is an arbitrary vector, such that
$(r_0, \widehat{r}_0) \neq 0$, e.g., $\widehat{r}_0 = r_0$
$\rho = \alpha = \omega_1 = \omega_2 = 1$; $w = v = p = 0$;
for $i = 0, 2, 4, 6, ...$
$\rho' = -\omega_2 \rho$;
even Bi-CG step:
$\rho = (r_i, \widehat{r}_0)$; $\beta = \alpha\rho/\rho'$; $\rho' = \rho$;
$p = r_i - \beta(p - \omega_1 v - \omega_2 w)$;
$v = Ap$;
$\gamma = (v, \widehat{r}_0)$; $\alpha = \rho/\gamma$;
$r = r_i - \alpha v$;
$s = Ar$;
$x = x_i + \alpha p$;
odd Bi-CG step:
$\rho = (s, \widehat{r}_0)$; $\beta = \alpha\rho/\rho'$; $\rho' = \rho$;
$v = s - \beta v$;
$w = Av$;
$\gamma = (w, \widehat{r}_0)$; $\alpha = \rho/\gamma$;
$p = r - \beta p$;
$r = r - \alpha v$;
$s = s - \alpha w$;
$t = As$;
GMRES(2)-part:
$\omega_1 = (r, s)$; $\mu = (s, s)$; $\nu = (s, t)$;
$\tau = (t, t)$; $\omega_2 = (r, t)$;
$\tau = \tau - \nu^2/\mu$; $\omega_2 = (\omega_2 - \nu\omega_1/\mu)/\tau$;
$\omega_1 = (\omega_1 - \nu\omega_2)/\mu$;
$x_{i+2} = x + \alpha p + \omega_1 r + \omega_2 s$;
$r_{i+2} = r - \omega_1 s - \omega_2 t$;
if x_{i+2} accurate enough then quit;
endfor

Alg.3: The BiCGstab(2) algorithm.

Our next numerical example illustrates quite nicely the difference in convergence behavior of some of the methods that we have discussed.

Example. We consider an advection dominated 2nd order PDE, with Dirichlet boundary conditions, on the unit cube (this equation was taken from [41]):

$$-u_{xx} - u_{yy} - u_{zz} + 1000\, u_x = f. \qquad (7.2a)$$

The function f is defined by the solution

$$u(x, y, z) = xyz(1 - x)(1 - y)(1 - z).$$

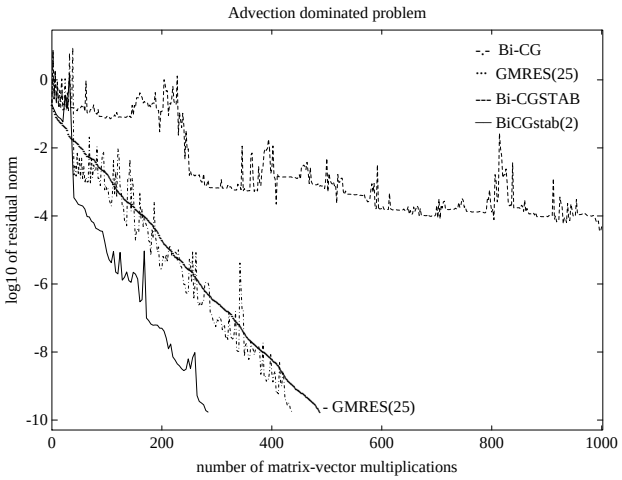

Fig.2: Convergence plot.

This equation was discretized using $22 \times 22 \times 22$ volumes, resulting in a seven-diagonal linear system of order 10648. In order to make differences between iterative methods more visible, we have here and in our other examples not use any form of preconditioning.

In Figure 2 we see a plot of the convergence history. Bi-CGSTAB almost stagnates, as might be anticipated from the fact that this linear system has eigenvalues with relatively large imaginary parts. Surprisingly, Bi-CGSTAB does even worse than Bi-CG. For this type of matrices this behavior of Bi-CGSTAB is not uncommon and, as we will see in the next subsection, this can be explained by the poor recovery of the Bi-CG iteration coefficients α_k and β_k. BiCGstab(2) converges quite nicely and almost twice as fast as Bi-CG. GMRES(25) is about as fast as Bi-CG. Since the GMRES steps are much more expensive, Bi-CGstab(2) is the most efficient method here.

7.3 Maintaining Convergence:

The BiCGstab methods are designed for smooth convergence, with the purpose to avoid loss of local bi-orthogonality in the underlying Bi-CG process. This is important, since then, as has been observed in Section 6.1, the convergence of the Bi-CG part is exploited as much as possible. However, local bi-orthogonality may also be disturbed by, for instance, inaccuracies in the Bi-CG coefficients α and β. They are the quotients of scalars $\rho = (r_i, \widehat{r}_0)$ and $\gamma = (Ap, \widehat{r}_0)$ (cf. Algorithm 2 and 3) and they will be inaccurate if ρ or γ is relatively small (see (7.3b)). The question is, when does this occur and how can it be avoidedΓ Here, we will concentrate on ρ only, but similar arguments apply to γ as well.

As in the introduction of this section, r_i is the residual $r_i = \widetilde{P}_i(A)P_i(A)r_0$ where $\widetilde{P}_i$ is an appropriate polynomial of degree i with $\widetilde{P}_i(0) = 1$. Now, ρ is given by

$$\rho \equiv \rho_i \equiv (\widetilde{P}_i(A)P_i(A)r_0, \widehat{r}_0). \tag{7.3a}$$

The scalar ρ_i can be small if the underlying Bi-Lanczos process nearly breaks down (i.e. $(\widetilde{P}_i(A)P_i(A)r_0, \widehat{r}_0) \approx 0$ relatively, for any polynomial $\widetilde{P}_i$ of degree i). Also an 'unlucky' choice of $\widetilde{P}_i$ may lead to a small ρ_i (which occurs in Bi-CGSTAB if the GCR(1) part stagnates). Here, we will concentrate on typical Bi-CGSTAB situations. Therefore, we assume that the Bi-Lanczos process itself (and the LU decomposition) does not (nearly) break down.

The relative rounding error ϵ_i in ρ_i can relatively and sharply be bounded by

$$|\epsilon_i| \leq n\,\overline{\xi}\frac{(|r_i|, |\widehat{r}_0|)}{|(r_i, \widehat{r}_0)|} \leq n\,\overline{\xi}\frac{\|r_i\|\,\|\widehat{r}_0\|}{|(r_i, \widehat{r}_0)|}. \tag{7.3b}$$

For a small relative error we want to have the expression at the right-hand side as small as possible.

Since the Bi-CG residual $P_i(A)r_0$, here to be denoted by s_i, is orthogonal to $\mathcal{K}_i(A^T; \widehat{r}_0)$ it follows that

$$(r_i, \widehat{r}_0) = \theta_i(A^i s_i, \widehat{r}_0)$$

if

$$\widetilde{P}_i(A) = \theta_i A^i + \theta_{i-1}^{(i)} A^{i-1} + \ldots.$$

Therefore, since $\|\widehat{r}_0\|/|(A^i s_i, \widehat{r}_0)|$ does not depend on $\widetilde{P}_i$, minimizing the right-hand side of (7.3b) is equivalent to minimizing

$$\frac{\|\widetilde{P}_i(A)s_i\|}{|\theta_i|} \tag{7.3c}$$

with respect to all polynomials $\widetilde{P}_i$ of exact degree i with $\widetilde{P}_i(0) = 1$. This minimization problem is solved by the FOM polynomial P_i^{F}, here associated with the initial residual s_i: P_i^{F} is the i^{th} degree polynomial for which $r_i^{\mathrm{F}} = P_i^{\mathrm{F}}(A)s_i$ (cf. Section 3). This polynomial is characterized by (cf. (3.1b))

$$P_i^{\mathrm{F}}(A)s_i - \mathcal{K}_i(A; s_i) \text{ and } P_i^{\mathrm{F}}(0) = 1.$$

For optimally accurate coefficients, we should select FOM polynomials for our polynomials $\widetilde{P}_i$.

However, since the hybrid Bi-CG methods are designed to avoid all the work for the construction of an orthogonal basis, the selection of complete FOM polynomials is out of the question.

For efficiency reasons, we have used products of first degree polynomials in Bi-CGSTAB and products of degree ℓ polynomials in BiCGstab(ℓ). Of course, our arguments can also be applied to such low degree factors. Therefore, suppose that $s = Q_{i-\ell}(A)P_i(A)r_0$ (as BiCGstab(ℓ)) has been computed and that the vectors s, As, ..., $A^\ell s$ are available. The suggestion for BiCGstab(ℓ) to minimize the residual over this particular Krylov subspace is equivalent to selecting a polynomial factor q_i ($Q_i = q_i Q_{i-\ell}$) of exact degree ℓ with $q_i(0) = 1$ such that

$$\|q_i(A)s\| \tag{7.3d}$$

is minimal, while in this situation, for optimal accurate coefficients, we rather would like to minimize

$$\|q_i(A)s\| \tag{7.3e}$$

where, with θ_i such that $q_i(A) = \theta_i A^\ell + \ldots$,

$$\|q_i(A)s\| \equiv \frac{\|q_i(A)s\|}{|\theta_i|}. \tag{7.3f}$$

The GMRES polynomial q_i^{G} of degree ℓ solves (7.3d), the FOM polynomial q_i^{F} solves (7.3e). For small residuals, the FOM polynomial is not optimal:

$$\|q_i^{\mathrm{G}}(A)s\| = |c_i|\,\|q_i^{\mathrm{F}}(A)s\|$$

with c_i as in (3.1f). Similarly, for accurate coefficients, the GMRES polynomial is not optimal [11]:

$$\|q_i^{\mathrm{F}}(A)s\| = |c_i|\,\|q_i^{\mathrm{G}}(A)s\|$$

with the same scalar c_i. For degree 1 factors, as in Bi-CGSTAB, (assuming no preconditioning)

$$c_i = \frac{(s, As)}{\|s\|\,\|As\|}, \tag{7.3g}$$

and c_i is the cosine of the angle between s and As (in, Algorithm 2, t represents As).

Clearly, for extremely small $|c_i|$, say $|c_i| \leq \sqrt{\xi}$ (in the $\ell = 1$ case, this means that s and As are almost orthogonal), taking GMRES polynomials for the degree ℓ factors will lead to inaccurate coefficients ρ_i, α and β, while FOM polynomials on the other hand will lead to large residuals. In both situations, the speed of convergence will seriously be deteriorated. The same phenomena can be observed when in a consecutive number of sweeps $|c_i|$ is small, but not necessarily extremely small (say, it takes k sweeps before $|c_{i-k}c_{i-k+1}\cdots c_i| \leq \sqrt{\xi}$). In other words, the inaccuracies seem to accumulate. This seems to occur quite often in practise. E.g., for linear equation stemming from PDEs with large advection terms, Bi-CGSTAB often stagnates, although all c_i may be larger than, say .1, and none of the ω_i can considered to be relatively small ($\omega_i = c_i\|s\|/\|As\|$).

Both Bi-CGSTAB and BiCGstab(ℓ) are built on top of the same Bi-CG process. At roughly the same computational costs, one sweep of BiCGstab(ℓ) covers the same Bi-CG track as ℓ sweeps of Bi-CGSTAB. In one sweep of BiCGstab(ℓ), GMRES(ℓ) is applied once, in ℓ sweeps of BiCGSTAB, GMRES(1) is applied ℓ times. For two reasons it pays off to use GMRES(ℓ) instead of $\ell\times$GMRES(1):

1. Due the super-linear convergence, one sweep of GMRES(ℓ) may be expected to give a better residual reduction than ℓ times GMRES(1).
2. In ℓ steps of GMRES(1), ℓ small c_i's may contribute to inaccuracies in the coefficients α and β, where GMRES(ℓ) contributes ato this only once.

BiCGstab(ℓ) profits from GMRES(ℓ) by a better residual reduction in the GMRES part and by the faster convergence of a better recovered Bi-CG due to the more stable computations. However, we do not recommend to take ℓ large; $\ell = 2$ or $\ell = 4$ will usually lead already to almost optimal speed of convergence. The computational costs increase slightly by increasing ℓ (i.e. $2\ell+10$ vector updates and $\ell+7$ inner products per 4 matrix multiplications), and more vectors have to be stored ($2\ell+5$ vectors). Moreover, the method is less accurate for larger ℓ due to the fact that intermediate residuals (as r and $r-\omega_1 s$ in Algorithm 3) can be large, with similar negative effects as in Section 6.2.

For Bi-CGSTAB there is a simple strategy that relaxes the danger of error amplification in consecutive sweeps with small $|c_i|$: replace in the Bi-CGSTAB algorithm 2 the line

'$\omega = (s,t)/(t,t)$'

by the piece of code in Algorithm 4. In this way we limit the size of $|c|$. The constant .7 is rather arbitrarily and may be replaced by any other fixed non-small constant less than 1. Since GMRES(1) reduces well only if $|c_i| \approx 1$ (see (3.1e)), this strat-

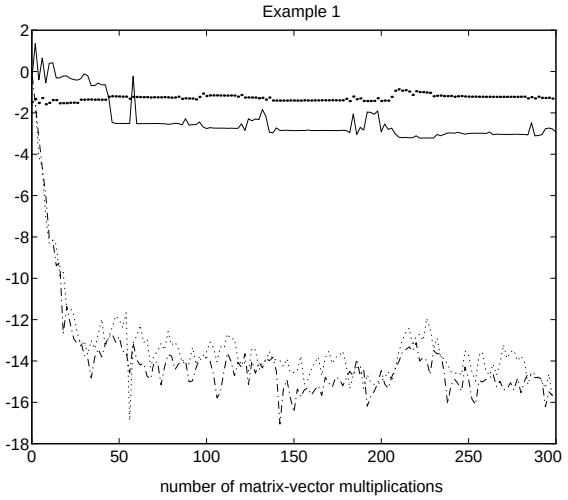

Fig.3: Convergence Bi-CGSTAB.

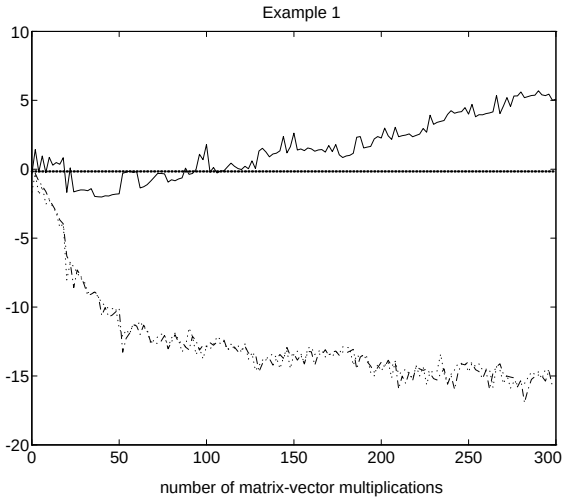

Fig.4: Convergence stabilized Bi-CGSTAB.

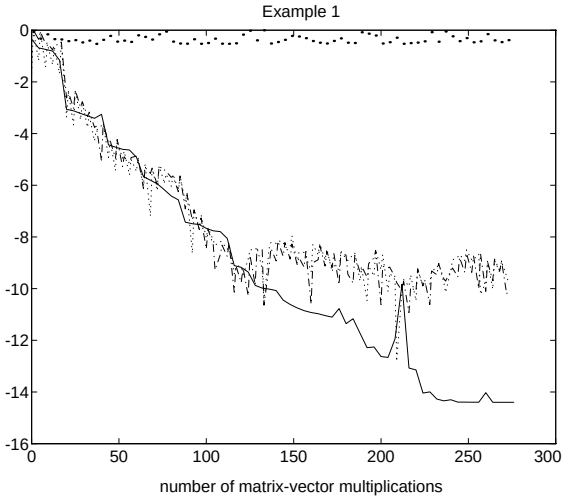

Fig.5: Convergence BiCGstab(2).

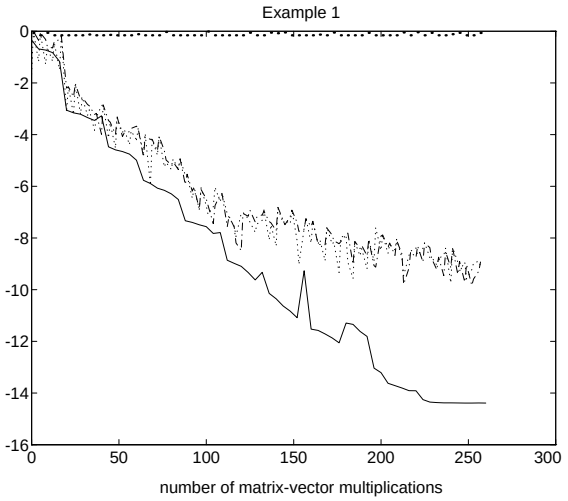

Fig.6: Convergence stab. BiCGstab(2).

egy still profits from a possible good reduction by GMRES(1). A similar strategy that is equally cheap and easy to implement can be applied to BiCGstab(ℓ); see [11] for details.

$$
\begin{aligned}
&\vdots\\
&c = (s,t)/(\|s\|\,\|t\|);\\
&\omega = \mathrm{sign}(c)\,\max(|c|, 0.7)\,\|s\|/\|t\|;\\
&\vdots
\end{aligned}
$$

Alg.4: Limiting the size of $|c|$.

We give a few numerical examples that demonstrate the cumulative effects of small $|c|$'s and that illustrate the effects of limiting its sizes.

Examples. The figures for the examples display, all on $\log_{10}$-scale, the values for each iteration step of
– the residual-norms $\|r\|$, by solid curves (——);
– the scaled ρ, $\widehat{\rho} \equiv |(r, \widehat{r}_0)|/(\|r\|\,\|\widehat{r}_0\|)$ (cf. (7.3b)), by dashed-dotted curves (- · - ·);
– the scaled γ: $\widehat{\gamma} \equiv |(Ap, \widehat{r}_0)|/(\|Ap\|\,\|\widehat{r}_0\|)$, by dotted curves (· · · · ·);
– $|c|$, resp. $\max(|c|, 0.7)$, by bullets (●●●).

Before describing the examples, we will discuss part of the results.
In the figures 3–14, we see that the scaled ρ and the scaled γ behave similarly (the dashed-dotted – and dotted curves coincide more or less). Further,

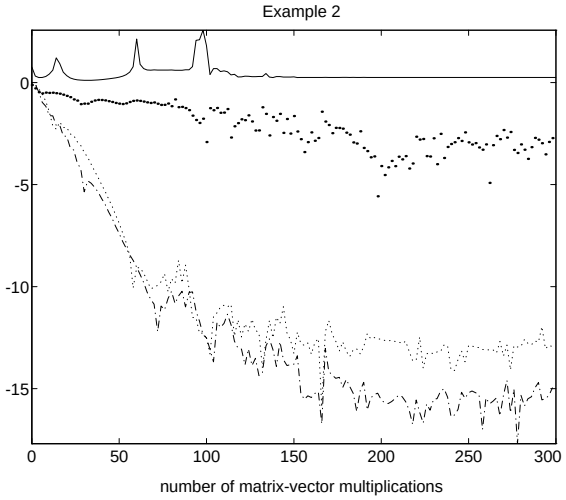

Fig.7: Convergence Bi-CGSTAB.

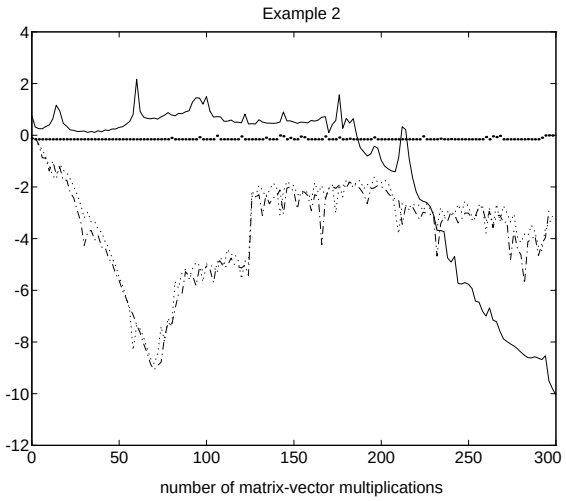

Fig.8: Convergence stabilized Bi-CGSTAB.

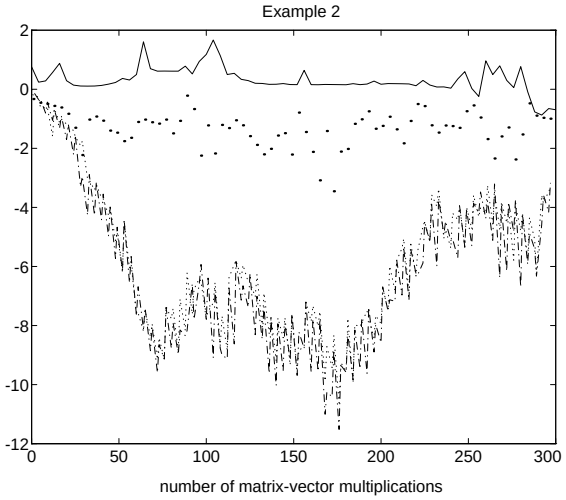

Fig.9: Convergence BiCGstab(2).

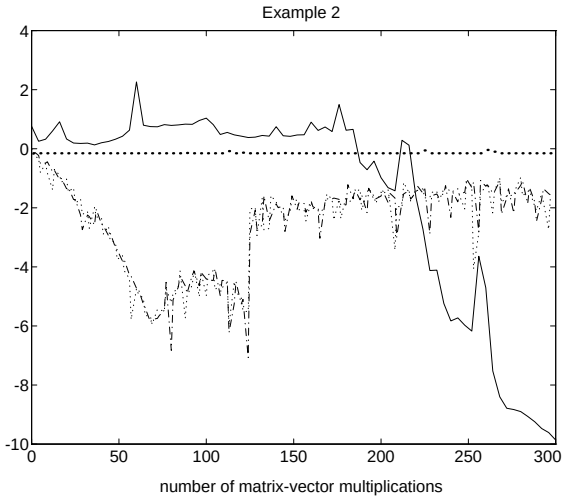

Fig.10: Convergence stab. BiCGstab(2).

none of the $|c|$ is extremely small even not in cases where the $\widehat{\rho}$ and $\widehat{\gamma}$ are. The decrease of $\widehat{\rho}$ for values of $\widehat{\rho}$ not in the range of the machine precision ($\geq 10^{-12}$) seems to be proportional to the product of previous $|c|$'s. In all the examples, the method stagnates if $\widehat{\rho}$'s or $\widehat{\gamma}$'s become extremely small, say less than 10^{-12}. In these cases, almost all significance of the Bi-CG coefficients α and β will be lost. Limiting the size of $|c|$ (Algorithm 4) slows down the decrease of $\widehat{\rho}$ and $\widehat{\gamma}$. In the caption of the figures, we used the adjective 'stabilized' to indicate that we used the limiting strategy. Often 'stabilizing' is enough to overcome the stagnation phase, and to lead to a converging process.

Example 1 (Figures 3–6). BiCGstab(2) converges. Although stabilizing Bi-CGSTAB leads to more accurate Bi-CG coefficients in the initial phase of the process, this is apparently not enough to restore full convergence.

Example 2 (Figures 7–10). Increasing ℓ to $\ell = 2$ leads to a slowly converging BiCGstab(2) process (many more than 300 matrix vector multiplications are needed; not shown in the graph). Our simple stabilizing strategy works well here.

Example 3 (Figures 11–14). The combined improvements, stabilizing and increasing ℓ to $\ell = 2$, are necessary for convergence.

For the first example, we have taken the PDE of (7.2a). The righ-hand side f is defined by the solution

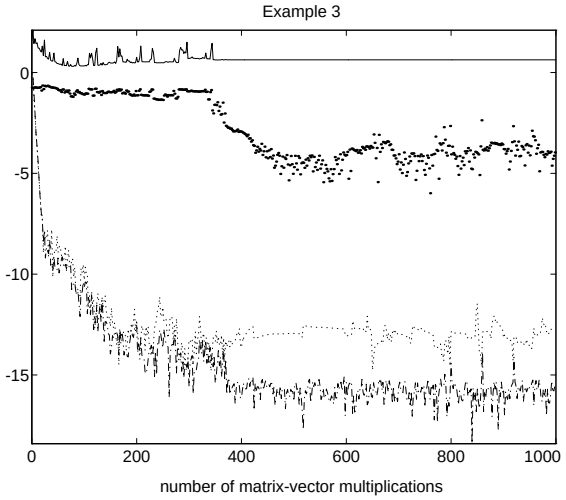

Fig.11: Convergence Bi-CGSTAB.

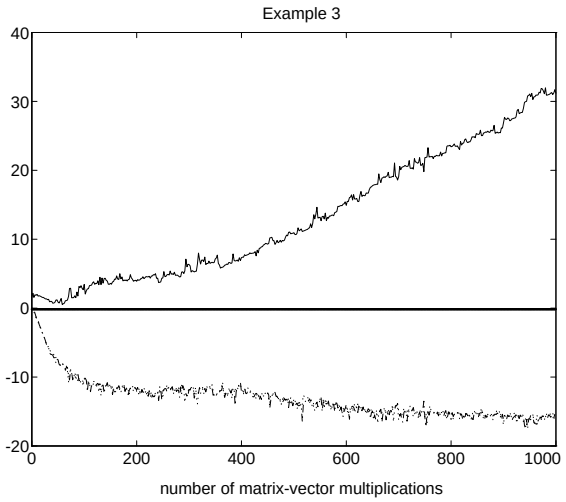

Fig.12: Convergence stabilized Bi-CGSTAB.

$$u(x,y,z) = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z).$$

The discretization is with $10 \times 10 \times 10$ finite volumes (no preconditioner has been used).

In the second and third example [35, 42], we have discretized

$$-u_{xx} - u_{yy} + a(xu_x + yu_y) + bu = f$$

on the unit-square with Dirichlet boundary conditions, with 63×63 finite volumes, taking $a = 100$ and $b = -200$, respectively 66×66, $a = 1000$ and $b = 10$ (no preconditioner has been used). The function f is such that the discrete solution is constant 1 (on the grid).

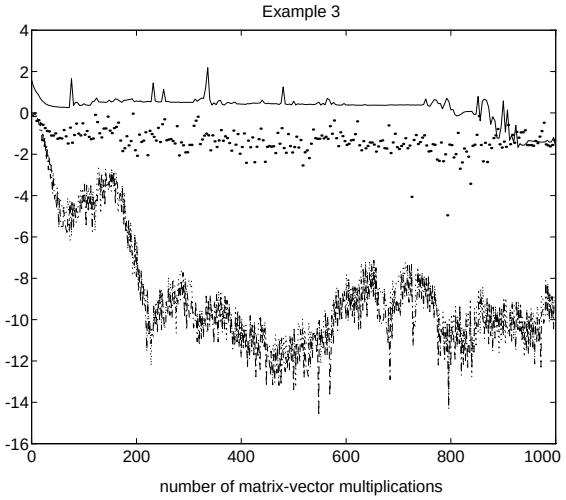

Fig.13: Convergence BiCGstab(2).

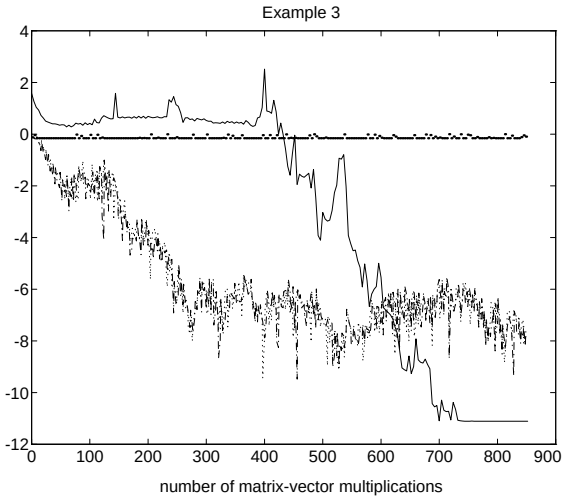

Fig.14: Convergence stab. BiCGstab(2).

7.4 Generalized CGS:

We have now discussed in some detail the family of BiCGstab(ℓ) methods, but one should not deduce from this that these methods are to be preferred over CGS in all circumstances. We have had very good experiences with CGS in the context of solving nonlinear problems with Newton's method. It turns out that we can exploit some of the presented ideas also to improve on CGS itself.

In the Newton method one has to solve a Jacobian system for the correction. This can be done by any method of choice, e.g., CGS or Bi-CGstab(ℓ). Often fewer Newton steps are required to solve a non-linear problem accurately when using CGS. Although the BiCGstab methods tend

to solve each of the linear systems (defined by the Jacobi matrices) faster, the computational gain in these inner loops does not always compensate for the loss in the outer loop because of more Newton steps.

This phenomenon can be understood as follows. For eigenvalues λ that are extremal in the convex hull of the set of all eigenvalues of A (the Jacobian matrix), the values $P_i(\lambda)$ of the Bi-CG polynomials P_i tend to converge more rapidly towards zero than for eigenvalues λ in the interior. Since CGS squares the Bi-CG polynomials, CGS may be expected to reduce extremely well the components of the initial residual r_0 in the direction of the eigenvectors associated with extremal eigenvalues λ: with reduction factor $P_i(\lambda)^2$. Of course, the value $P_i(\lambda)$ can also be large, specifically for interior eigenvalues and in an initial stage of the process. CGS amplifies the associated components, (which also explains the typical irregular convergence behavior of CGS). The BiCGstab polynomial Q_i does not have this tendency of favoring the extremal eigenvalues. Therefore, the BiCGstab methods tend to reduce all eigenvector components equally well: on average, the "interior components" of a BiCGstab residual r_i are smaller than the corresponding components of a CGS residual $\widetilde{r}_i$, while, with respect to the exterior components the situation is the other way around. However, the non-linearity of a non-linear problem seems often to be represented rather well by the space spanned by the "extremal eigenvectors". With respect to this space, and hence with respect to the complete space, Newtons scheme with CGS behaves like an exact Newton scheme.

We would like to preserve this property when constructing iterative schemes for Newton iterations. Fokkema et al [43] suggest polynomials $\widetilde{P}_i$ that lead to efficient algorithms (small modifications of the CGS algorithm) with a convergence that is slightly smoother, faster, and more accurate, than for CGS, but that still has the property of reducing extremal components quadratically. As a linear solver for isolated linear problems these "generalized CGS" schemes do not seem to have much advantage over BiCGstab(ℓ), but as a linear solver in a Newton scheme for non-linear problems, they often do rather well.

8 RELIABLE UPDATING

In all the Bi-CG related methods we see that the approximation for x and the residual vector r are updated by different vectors, and that the value for x does not influence the further iteration process, whereas the value for r does. In exact arithmetic the updated r is equal to the true residual $b-Ax$, but in rounded arithmetic it is unavoidable that differences between r and $b-Ax$ arise. This means that we may be misled for our stopping criteria, which are usually based upon knowledge of the updated r (and that we may have iterated too far in vain).

In this section we will discuss some techniques that have been proposed recently for the improvement of the updating steps. It turns out that this can be settled by relatively easy means.

Although the techniques in the previous section led to smoother and faster convergence and more accurate approximations the approximation may still not as accurate as possible. Here, we strive for optimal accuracy, i.e the updated r_i should be very close to the values of $b-Ax_i$, while leaving the convergence of the updated r intact.

First, we observe that even if x_m is the exact solution then the residual, computed in rounded arithmetic as $b-Ax_m$, may not be expected to be zero: using the notation of Section 6.2,

$$\begin{aligned} \|b-Ax_m\| &\leq \overline{\xi}\left(\|b\| + n_A \left\||A|\right\| \|x_m\|\right) \\ &\leq 2\,\Gamma\,\overline{\xi}\,\|b\|. \end{aligned} \tag{8.1a}$$

Therefore, the best we can strive for is an approximation x_m for which the true residual and the updated one differ in order of magnitude by the initial residual times the relative machine precision ($\mathcal{O}(\overline{\xi}\,\|r_0\|)$; recall that we assumed $x_0 = 0$, and hence $r_0 = b$).

Now it becomes also obvious why it is a bad idea to replace the updated residual in each step by the true one. Except from the fact that this would cost an additional matrix vector multiplication in each step, it also introduces errors in the recursions for the residuals. Although these errors may be expected to be small relatively to r_0, they will be large relatively to r_i if $\|r_i\| \ll \|r_0\|$. This perturbs the local bi-orthogonality of the underlying Bi-CG process and it may significantly slow down the speed of convergence. This observation suggests to replace the updated residual by the true one only if the updated residual has the same order of magnitude as the initial residual. However, meanwhile x_i and r_i may have drifted apart, and replacing r_i by $b-Ax_i$ brings in the "error of x_i" in the recursion (bounded as in (6.2d)), and

again the speed of convergence may be affected. Although it is a good idea to use true residuals at strategic places, the approximation x_i should first be 'tied' more closely to the updated residual r_i. We can achieve this by updating x_i cumulatively: if $x_i = x_0 + w_1 + \ldots + w_i$ (cf. (6.2a)) then we actually compute x_i in groups as

$$x_i = x_0 + x'_1 + x'_2 + \ldots \qquad (8.1b)$$

where, for some decreasing sequence of indices $\pi(1) = 1$, $\pi(2)$, ..., x'_j represents the sum of a group;

$$x'_j = w_{\pi(j)} + w_{\pi(j)+1} + \ldots + w_{\pi(j+1)-1}, \text{ etc..}$$

Simultaneously, we compute r_i as

$$r_i = r_0 - Ax'_1 - Ax'_2 - \ldots. \qquad (8.1c)$$

In this way we can control the size of the updates for x_i and r_i, and we avoid large errors (cf. (6.2d)): for a proper choice of the $\pi(j)$, the x'_j will be small even if some of the w_j are large.

In the modification of the algorithms that we will propose in Algorithm 5, we kept in mind that we only may allow errors which
(a) are small with respect to the initial residual r_0 (otherwise accuracy will be disturbed) and
(b) are small with respect to the present updated residual r_i (otherwise local bi-orthogonality may be jeopardized).

In Section 2, we have explained that it is no restriction to take $x_0 = 0$, arguing that this situation can be forced simply by a shift: shift x by x_0, and b by Ax_0. This shift can be made explicit in the hybrid Bi-CG algorithms by making three changes:
(i) adding as a last line to the initialization phase

$$x = x_0; \quad x' = 0; \quad b' = r_0;$$

(ii) adding as a last line in the algorithms (just after 'end')

$$x = x + x';$$

(iii) replacing all x_i (and x) by x' (skipping the index i).
Even in rounded arithmetic, this modification will not change the value of any of the vectors and scalars in the computational scheme, except for the x's. Since $x + x'$ is the approximation that we are interested in, one also may want to change the termination criterion. We propose to replace the line

if x is accurate enough then quit;

by

if $\|r_{i+1}\|$ is small enough then quit;

To allow for a more accurate way of updating of the residual and the approximation, we suggest to add another few lines just before 'end' in the algorithm, as is shown in Algorithm 5. We

```
    .
    .
x = x_0;   x' = 0;   b' = r_0;
for i = 0, 1, 2, ...
        .
        .   Replace all x_i and x by x'.
        .
    if r_{i+1} is small enough then quit;
    set 'compute_res' and 'update_app';
    if 'compute_res' is true
        r_{i+1} = b' - Ax';
        if 'update_app' is true
            x = x + x'; x' = 0; b' = r_{i+1};
        endif
    endif
endfor
x = x + x';
```

Alg.5: For accurate approximations.

suggest to replace the updated residual by the true one on strategically chosen steps (we have to explain when the value of the boolean functions *'compute_res'* is true). However, we also suggest to shift the problem once in a while (when the boolean function *'update_app'* is true) in order to let the right-hand decrease (cf. (8.1a)). Here we use the fact that, in exact arithmetic, also these intermediate shifts do not change the iteration parameters and vectors (except for the vectors x). Observe that the updated residual r_{i+1} is replaced by the true residual $b' - Ax'$ of the shifted problem if *'compute_res'* is true.

For this we propose the following strategy.
Update x and b' only if the residual is significantly smaller than the initial residual, while an intermediate residual was larger (cf. (8.1b), (8.1c) and reminder (a)):

$$\begin{aligned} &\textit{'update\_app'} = \text{true} \\ &\text{if} \quad \|r_{i+1}\| \le \|b\|/100 \;\;\&\;\; \|b\| \le \mu \qquad (8.1d) \\ &\text{else } \textit{'update\_app'} = \text{false}, \end{aligned}$$

where $\mu \equiv \max\|r_i\|$ and the maximum is taken over all residuals since the previous update of x and b' (since the previous *'update_app'* is true).
The bound in (8.1a) suggests that the norm $\|b\|$ of the initial residual should be used as criterion for shifting the problem (*'update_app'* is true if $\|r_{i+1}\| \leq \|b\|$ & $\|r_i\| \geq \|b\|$). However, if the process converges irregularly this would lead to many shifts. The relaxed version in (8.1d) turns out to work equally well at less costs.
Compute a true residual whenever *'compute_res'* is true and if a previous residual is larger than the initial residual and significantly larger than the present updated residual:

$$\begin{aligned} &\textit{'compute\_res'} = \mathsf{true} \\ &\mathsf{if} \quad \|r_{i+1}\| \leq M/100 \ \ \& \ \ \|b\| \leq M \\ &\quad\ \ \mathsf{or} \ \textit{'update\_app'} \ \mathsf{is\ true} \\ &\mathsf{else} \ \textit{'compute\_res'} = \mathsf{false}, \end{aligned} \tag{8.1e}$$

where $M \equiv \max\|r_i\|$ and the maximum is taken over all residuals since the last computation of the true residual.
Replacing the updated residual by the true one perturbs the recursion for the residuals. If the residual decreases too much since the previous replacement, the perturbation may become large relatively to the present residual (reminder (b)). Therefore, *'compute_res'* may be true more often than *'update_app'*.

We suggest to add the above strategy to an existing code. That means that an additional matrix-vector multiplication has to be performed whenever a true residual has to be computed. The conditions (8.1d) and (8.1e) are chosen as to minimize the number of these additional computations. One also may try to skip a matrix-vector multiplication in one of the preceding lines of the algorithm, which requires some additional care for BiCGstab(ℓ), but which easily can be accomplished for CGS (cf. Algorithm 1).

If CGS is modified as suggested, then the new lines do not require additional matrix vector multiplications, and there is no need to restrict the number of computations of true local residuals. For this CGS variant, Neumaier [44] suggested places where the x and b' can be updated for accurate approximations: update x and b' whenever the residual decreases with respect to the previous 'best residual',

$$\begin{aligned} &\textit{'update\_app'} = \mathsf{true} \\ &\mathsf{if} \quad \|r_{i+1}\| \leq \|b'\| \\ &\mathsf{else} \ \textit{'compute\_res'} = \mathsf{false}. \end{aligned} \tag{8.1f}$$

The modifications according to Neumaier's approach are given in Algorithm 6. Observe that

```
      :
x = x_0;  x' = 0;  b' = r_0;  μ' = ||b'||;
for i = 0, 1, 2, ...
          :
          :  Replace all x_i and x by x'.
          :
Skip the CGS update for r
together with the MV involved
in this update. Compute instead
    r_{i+1} = b' − Ax';  μ = ||r_{i+1}||;
    if μ is small enough then quit;
    if μ ≤ μ'
        x = x + x';  x' = 0;
        b' = r_{i+1};   μ' = μ;
    endif
endfor
x = x + x';
```

Alg.6: Neumaier's strategy for CGS.

the norm of the b' (the residuals with respect to the x) strictly decrease: the Neumaier trick also smoothes convergence (without improving its speed!).

Below, we discuss the effects of our strategies in practise. We illustrate our observations by a simple numerical example.

Example. Figure 15 shows the convergence history of the *true residuals* as produced by standard CGS, and by the modified versions of CGS as suggested above, applied to the SHERMAN4 matrix of the Harwell-Boeing collection (as in the example of Section 6.2). The dotted curve ($\cdots$) represents the results for standard CGS. We also applied modified CGS as in Algorithm 5, using the update criterions (8.1d) and (8.1e). The solid curve (——) represents the results for this *simple strategy*, while the dashed-dotted curve (- · - ·) represents the results for Neumaier's strategy in Algorithm 6. On $\log_{10}$-scale, the norm of the

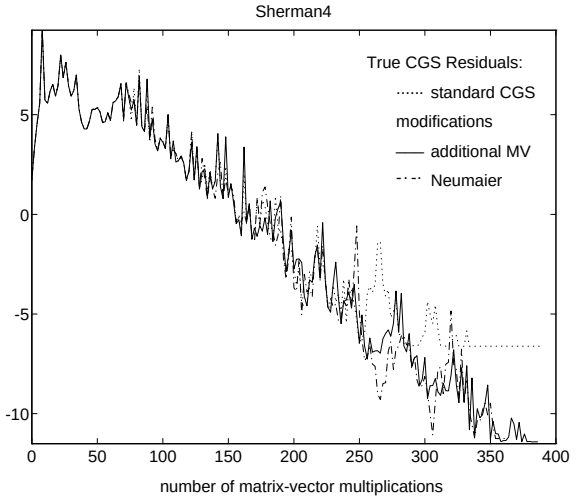

Fig.15: Reliable updates.

true residuals $\|b - Ax_i\|$, $\|b - A(x + x')\|$, respectively, is plotted against the number of matrix-vector multiplications. Neumaier's strategy as well our's lead to approximations that are accurate (cf. (8.1a)): comparing $\|r_0\|$ with the norm of the smallest true residual, we see that a reduction is obtained by a factor $\approx 10^{-14}$ ($\overline{\xi} = 2.2\,10^{-16}$). Standard CGS does not produce true residuals smaller than $\approx 10^{-9}\|r_0\|$, which is approximately $\overline{\xi} \cdot \max\|r_i\| \approx 2.2\,10^{-16} \cdot 10^{7}\|r_0\|$; cf. (6.2d). Observe that, though the convergence histories do not coincide for residuals less than $\approx 10^{1}$, the speed of convergence is not affected: the modified versions exhibit a rate of convergence that is very similar to the one of the updated residuals in standard CGS as shown in Figure 1.

Experiments for other examples and with other iterative schemes, as Bi-CGSTAB and BiCGstab(ℓ), led to similar conclusions. Although, two observations should be made.
— Quite often the improvements are much more spectacular than for this SHERMAN4 example: CGS may produce intermediate residuals as large as $\|r_0\|/\overline{\xi}$ and none of the digits in the finial approximation of standard CGS will be correct.
— There are some differences between CGS and the BiCGstab methods: (i) as observed above, Neumaier's strategy only works well for CGS, while the simple strategy of Algorithm 5 can always be applied. (ii) Especially for the BiCGstab methods, the simple strategy of Algorithm 5 with update criterions (8.1d) and (8.1e) does not lead to much additional work. The additional computation of a true residual takes place after the process encounters residuals that are (much) larger than the initial residual. Since BiCGstab(ℓ) tends to show much smoother convergence behavior than CGS, for small ℓ, the additional work in these methods is usually much less than for CGS. In the SHERMAN4 example, our strategy for CGS requires 7 additional matrix-vector multiplications (*'compute_res'* is true 7 times) and one special update of the approximation (*'update_app'* is true only once). For BiCGstab(ℓ), $\ell \leq 6$, only 1 additional matrix-vector multiplication was needed. Neumaier's strategy for CGS does not require additional matrix-vector multiplications (but 364 additional updates for the approximation were needed).

9 A NOTE ON PRECONDITIONING

We have argued that the described composite methods can be viewed as basic methods preconditioned with other methods. This should not be confused with the more familiar preconditioning that one uses in order to improve spectral properties of the given operator A, in the hope to reduce the number of iteration steps.
It should be noted that we have not made any special assumption for the operator A in our algorithms. This implies that if we have a suitable preconditioner M available for the given system $Ax = b$, then we have the choice between applying any of the described algorithms to one of

$$M^{-1}Ax = M^{-1}b, \tag{9.1a}$$
$$AM^{-1}y = b \quad \text{with} \quad x = M^{-1}y, \tag{9.1b}$$
$$L^{-1}AU^{-1}z = L^{-1}b \quad \text{with} \quad M = LU, \quad x = U^{-1}y. \tag{9.1c}$$

The operations with A in the algorithms should then read as operations with $M^{-1}A$, AM^{-1}, or $L^{-1}AU^{-1}$, respectively. Of course, the vector b in the algorithms should read as $M^{-1}b$, b, or $L^{-1}b$, respectively.
It is well-known that for these operations it is almost never advantageous to invert the operators M, L, and U, explicitly. Instead, a statement like, for instance, $w = M^{-1}Av$ is almost always handled more economically by solving w from $Mw = Av$, since in general preconditioners are constructed to make this solution process inexpensive. A suitable class of such preconditioners are the well-known incomplete LU factorizations [45]. For remarks with respect to vector

computing and parallel computing, see Dongarra et al [33].
Recently it has been shown [46] that a reordering strategy in combination with limited fill-in can greatly improve the iteration reducing effect of incomplete LU factorizations for many relevant problems.

For an overview on preconditioning techniques that are relevant for CFD-problems, see Chan and Van der Vorst (1994).

REFERENCES

[1] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[2] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.*, 49:409–436, 1952.

[3] R. Fletcher. Conjugate gradient methods for indefinite systems, volume 506 of *Lecture Notes Math.*, pages 73–89. Springer-Verlag, Berlin–Heidelberg–New York, 1976.

[4] P. Sonneveld. CGS: a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10:36–52, 1989.

[5] R.W. Freund and N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Num. Math.*, 60:315–339, 1991.

[6] H.A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13:631–644, 1992.

[7] R.E. Bank and T.F. Chan. An analysis of the composite step biconjugate gradient method. *Numer. Math.*, 66:259–319, 1993.

[8] G.L.G. Sleijpen and D.R. Fokkema. Bi-CGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum. *Electronic Transactions on Numer. Anal. (ETNA)*, 1:11–32, 1993.

[9] M.H. Gutknecht. Variants of BiCGStab for matrices with complex spectrum. *SIAM J. Sci. Comput.*, 14:1020–1033, 1993.

[10] G.L.G. Sleijpen, H.A. Van der Vorst, and D.R. Fokkema. BiCGstab(ℓ) and other hybrid Bi-CG methods. *Numerical Algorithms*, 7:75–109, 1994.

[11] G.L.G. Sleijpen and H.A. Van der Vorst. Maintaining convergence properties of Bi-CGstab methods in finite precision arithmetic. *Preprint Nr. 861*, Dept. Math., University Utrecht, 1994. To appear in *Numerical Algorithms*.

[12] R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs N.J., 1962.

[13] C.C. Paige and M.A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Soft.*, 8:43–71, 1982.

[14] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comput.*, 37:105-126, 1981.

[15] D.M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York–San Francisco–London, 1971.

[16] O. Axelsson. Conjugate gradient type of methods for unsymmetric and inconsistent systems of linear equations. *Linear Algebra Appl.*, 29:1–16, 1980

[17] D.M. Young and K.C. Jea. Generalized conjugategradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra Appl.*, 34:159–194, 1980.

[18] S.C. Eisenstat, H.C. Elman and M.H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20:345–357, 1983.

[19] P.K.W. Vinsome. ORTHOMIN, an iterative method for solving sparse sets of simultaneous linear equations. Paper SPE 5729, *4th Symposium of Numerical Simulation of Reservoir Performance of the Society of Petroleum Engineers of the AIME*, 1976.

[20] P.N. Brown. A theoretical comparison of the Arnoldi and GMRES algorithms. *SIAM J. Sci. Statist. Comput.*, 12:58–78, 1991.

[21] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:315–339, 1984.

[22] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand.*, 49:33–53, 1952.

[23] B.N. Parlett, D.R. Taylor, and Z.A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.*, 44:105–124, 1985.

[24] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comput.*, 14:137–158, 1993.

[25] R.W. Freund and N.M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, part 2. *Technical Report 90.46*, RIACS, NASA Ames Research Center, 1990.

[26] C. Brezinski and M. Redivo-Zaglia. Treatment of near breakdown in the CGS algorithm. *Numerical Algorithms*, 7:33–74, 1994.

[27] A.T. Chronopoulos and S.K. Kim. s-Step Orthomin and GMRES implemented on parallel computers. *Technical Report 90/43R*, UMSI, Minneapolis, 1990.

[28] T.F. Chan and T. Szeto. A composite step conjugate gradient squared algorithm for solving nonsymmetric linear systems. *Numerical Algorithms*, 7:12–32, 1994.

[29] G. Radicati di Brozolo and Y. Robert. Parallel conjugate gradient-like algorithms for solving sparse non-symmetric systems on a vector multiprocessor. *Parallel Computing*, 11:223–239, 1989.

[30] G. Brussino and V. Sonnad. A comparison of direct and preconditioned iterative techniques for sparse unsymmetric systems of linear equations. *Int. J. for Num. Methods in Eng.*, 28:801–815, 1989.

[31] C. Pommerell and W. Fichtner. PILS: An iterative linear solver package for ill-conditioned systems. In *Supercomputing '91*, ACM-IEEE, pp. 588–599, Albuquerque, NM, Nov., 1991.

[32] N.M. Nachtigal, S.C. Reddy, and L.N. Trefethen. How fast are nonsymmetric matrix iterationsΓ *SIAM J. Matrix Anal. Appl.* 13:778–795, 1992.

[33] J.J. Dongarra, I.S. Duff, D.C. Sorensen, and H.A. Van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM, Philadelphia, PA, 1991.

[34] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Dunato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, PA, 1993.

[35] R.W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14:470–482, 1993.

[36] G.L.G. Sleijpen and H.A. Van der Vorst. Reliable updated residuals in hybrid Bi-CG methods. *Preprint Nr. 886*, Dept. Math., University Utrecht, 1994.

[37] C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra Appl.*, 34:235–258, 1980.

[38] A. Greenbaum. Behavior of slightly perturbed Lanczos and conjugate gradient recurrences. *Linear Algebra Appl.*, 113:7–63, 1989.

[39] L. Zhou and H.F. Walker. Residual smoothing techniques for iterative methods. *SIAM J. Sci. Comput.*, 15:297–312, 1994.

[40] 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.*, 15:338–347, 1994.

[41] U. Meier Yang. Preconditioned Conjugate Gradient-Like Methods for Nonsymmetric Linear Systems. *Preprint*, Center for Research and Development, University of Illinois at UrbanaChampaign, 1992.

[42] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14:461–469, 1993.

[43] D.R. Fokkema, G.L.G. Sleijpen and H.A. Van der Vorst. Generalized Conjugate Gradient Squared. *Preprint 851*, Dept. Math., University Utrecht, 1994.

[44] A. Neumaier. Oral presentation at the Oberwolfach meeting: Numerical Linear Algebra, Oberwolfach, 1994.

[45] 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.*, 31:148–162, 1977.

[46] A. Van der Ploeg, E.F.F. Botta, and F.W. Wubs. Grid-independent convergence based on preconditioning techniques. *Technical Report W-9310*, University of Groningen, Groningen, 1993.