Computing 56, 141–163 (1996)

Computing

Reliable Updated Residuals in Hybrid Bi-CG Methods

G. L. G. Sleijpen and **H. A. van der Vorst**, Utrecht

Received February 1, 1995; revised May 29, 1995

Abstract — Zusammenfassung

Reliable Updated Residuals in Hybrid Bi-CG Methods. Many iterative methods for solving linear equations $Ax = b$ aim for accurate approximations to x, and they do so by updating residuals iteratively. In finite precision arithmetic, these computed residuals may be inaccurate, that is, they may differ significantly from the (true) residuals that correspond to the computed approximations. In this paper we will propose variants on Neumaier's strategy, originally proposed for CGS, and explain its success. In particular, we will propose a more restrictive strategy for accumulating groups of updates for updating the residual and the approximation, and we will show that this may improve the accuracy significantly, while maintaining speed of convergence. This approach avoids restarts and allows for more reliable stopping criteria. We will discuss updating conditions and strategies that are efficient, lead to accurate residuals, and are easy to implement. For CGS and Bi-CG these strategies are particularly attractive, but they may also be used to improve Bi-CGSTAB, BiCGstab(l), as well as other methods.

AMS Subject Classification: 65F10

Key words: Non-symmetric linear systems, iterative solver, CGS, Bi-CGSTAB, BiCGstab(l).

Zuverläßlich berechnete Residuen in hybriden Bi-CG Verfahren. Viele iterative Methoden zur Lösung linearer Gleichungssysteme berechnen die Iterierten über aufdatierte Residuen. In endlicher Arithmetik können diese Residuen sehr ungenau sein, d.h., sie können sich erheblich von den tatsächlichen unterscheiden. In dieser Arbeit stellen wir Varianten der Neumaier Strategie vor, die ursprünglich für das CGS-Verfahren vorgeschlagen wurde, und erklären deren Erfolge. Insbesondere werden wir eine Variante vorschlagen, bei der mehrere Aufdatierungsschritte zusammengefaßt werden. Wir zeigen, daß sich die Genauigkeit der berechneten Residuen dadurch erheblich verbessern läßt, ohne daß die Konvergenzgeschwindigkeit beeinträchtigt wird. Dieser Ansatz vermeidet Neustarts und ermöglicht zuverlässigere Abbruchkriterien. Wir diskutieren Aufdatierungsbedingungen und Strategien, die effizient und leicht zu implementieren sind. Diese Strategien führen zu genaueren Residuen und sind insbesondere für CGS und Bi-CG–aber auch für Bi-CGSTAB, BiCGstab(l) und andere Verfahren–sehr attraktiv.

1. Introduction

We will focus on the iterative solution of linear systems

$$Ax = b \tag{1}$$

in which A is a non-singular $n \times n$ matrix and b a given n-vector. Typically n is large and A is sparse. To simplify our presentation, we will assume A and b to be real. The class of iterative methods is characterized by the fact that the

update for the residual vector is computed independently from the current approximation to the solution. This class includes Krylov-type methods, like Bi-CG, CGS, and Bi-CGSTAB.

Of course, one is interested in an accurate approximation for the solution x. Since the exact solution is not known, the residual is usually used (in stopping criteria) to judge the quality of the approximation. Many iterative methods have recursively *updated residuals* available. These are usually exploited in stopping criteria, which is attractive, since the computation of a *true* residual $b - A\mathbf{x}_k$ for an approximation $\mathbf{x}_k$ usually requires an additional matrix-vector (MV) product. Unfortunately, the true and the updated residual drift apart during the iteration process, and this leads to misleading impressions of the actual errors. When the method produces a small updated residual $\mathbf{r}_k$ (as is the "purpose" of the method), then the true residual $b - A\mathbf{x}_k$ may not be small at all. In order to obtain small true residuals, we may have to restart the process, which is less attractive, since, by restarting, we may loose for instance the speed that has been accelerated by superlinear convergence. It may also be the case that we have done too many iteration steps, since the actual errors tend to stagnate in many cases where the updated residuals still further decrease beyond a certain value.

We will discuss relatively simple strategies that lead to more *accurate updated residuals* (that is, to small $\|b - A\mathbf{x}_k - \mathbf{r}_k\|$), without giving up any speed of convergence.

Our strategies and theoretical observations will be illuminated by applications to Bi-CG and to hybrid Bi-CG methods, like CGS, Bi-CGSTAB, and BiCGstab(l). However, they seem to be applicable to many other iterative methods as well.

In Section 2 we will briefly recall some relevant properties of these Bi-CG type of methods. In Section 3, we will explain the critical points in the computation of accurate residuals, in particular we will discuss the accuracy that is required to maintain speed of convergence. Next, in Section 4, we translate our observations into practical strategies that we illustrate in our Section 5 on numerical experiments.

2. Hybrid Bi-CG Methods

Starting with an initial guess x_0 for the solution x and a "shadow" residual $\tilde{r}_0$, Bi-CG [2,9] produces iteratively sequences of approximations x_k, residuals r_k, and search directions u_k by

$$u_k = r_k - \beta_k u_{k-1}, \qquad x_{k+1} = x_k + \alpha_k u_k, \qquad r_{k+1} = r_k - \alpha_k A u_k, \tag{2}$$

where the Bi-CG coefficients α_k and β_k are such that r_k and Au_k are orthogonal to the shadow Krylov subspace $\mathcal{K}_k(A^T; \tilde{r}_0)$. For any sequence of polynomials ψ_k of degree k with a non-trivial leading coefficient θ_k, the vectors

$\psi_0(A^T)\tilde{r}_0, \ldots, \psi_{k-1}(A^T)\tilde{r}_0$ form a basis of $\mathcal{K}_k(A^T; \tilde{r}_0)$ and we have (see [19] or [16]):

$$\beta_k = \frac{\theta_{k-1}}{\theta_k}\frac{\rho_k}{\sigma_{k-1}} \quad \text{and} \quad \alpha_k = \frac{\rho_k}{\sigma_k} \quad \text{where} \begin{cases} \rho_k := (r_k, \psi_k(A^T)\tilde{r}_0), \\ \sigma_k := (Au_k, \psi_k(A^T)\tilde{r}_0). \end{cases} \tag{3}$$

We can rewrite the inner products so as to avoid the operations with A^T, e.g.,

$$\rho_k = (r_k, \psi_k(A^T)\tilde{r}_0) = (\mathbf{r}_k, \tilde{r}_0) \qquad \text{where } \mathbf{r}_k := \psi_k(A)r_k, \tag{4}$$

and generate recursions for the vectors $\mathbf{r}_k$, hoping that the operator $\psi_k(A)$ would lead to a further reduction of the Bi-CG residual. The corresponding search directions for the corresponding approximation $\mathbf{x}_k$ can be constructed in a similar way. In this approach the Bi-CG residuals and search directions are not computed explicitly.

Sonneveld [19] suggested to take $\psi_k = \phi_k$, with ϕ_k such that $r_k = \phi_k(A)r_0$, which led to the CGS method: $\mathbf{r}_k = \phi_k^2(A)r_0$. The matrix polynomial $\phi_k(A)$ reduces the initial residual r_0 (in case of convergence) and one may hope that this polynomial reduces $\phi_k(A)r_0$ as well. Unfortunately, especially in the early phases of the process, there is often an amplification in Bi-CG that is squared in CGS, leading to large intermediate residuals which may affect the accuracy of the final residual (cf. Section 3 and [18], Section 2.1). A more subtle approach, in which ψ_k was taken as the Bi-CG polynomial for a 'nearby' process, was followed in [3] for the construction of GCGS.

Bi-CGSTAB [20] in Algorithm 1 attempts to avoid large intermediate residuals by selecting ψ_k as a product of linear factors that minimize residuals locally. Unfortunately, products of locally minimizing degree one polynomials may lead to inaccurate Bi-CG coefficients [17], and this may lead to poor convergence. By taking products of locally minimal degree l polynomials as in BiCGstab(l) [16, 18], one can often maintain the speed of convergence of the underlying Bi-CG process and avoid large intermediate residuals at the same time. The m-th sweep of BiCGstab(l) starts with increasing the index k of $\psi_j(A)r_k$ in l steps from $k = ml$ to $k = ml + l$, using the relations in (2), and then the index j is increased from $j = ml$ to $j = ml + l$, by l steps of, say, GMRES [15].

3. Improving the Accuracy

3.1. Errors in the Residuals

For Bi-CG methods, Bi-CGSTAB and CGS, one expects (and often observes indeed) that they inherit the attractive convergence properties (as super-linear convergence) from Bi-CG.

However, in finite precision arithmetic, evaluation errors can not be avoided. The best one can hope for is that these errors introduced in one iteration step

are relatively small, i.e. the new errors in steps as in (2) are small with respect to $\|u_k\|$ and $\|r_{k+1}\|$, say less than $\sqrt{(\bar{\xi})}\|u_k\|$ and $\sqrt{(\bar{\xi})}\|r_{k+1}\|$, respectively, where $\bar{\xi}$ is the relative machine floating point precision. Surprisingly, as we have learned from actual computations (see, for instance, [5, 17]) and from theoretical results for symmetric matrices A [7, 12], this is also sufficient for exhibiting the Bi-CG type of convergence behavior (see also [1]). This makes the methods attractive even when possibly relatively large global errors occur: *For convergence we need local errors that are locally relatively small.*

However, for an accurate updated residual it is not enough to have relatively small local errors: the accumulation of these local errors should be small as well [18]. For our discussion we recall arguments in [18], Section 3, that led to estimate (6). We restrict our analysis to the effect of rounding errors from the matrix-vector multiplications. For the accuracy results, other sources of rounding errors are of lesser importance and will not affect essentially the conclusions. For a more rigorous analysis, we refer to [8].

In the methods that we are interested in, the updates for $\mathbf{x}_k$ and $\mathbf{r}_k$ are related as follows

$$\mathbf{x}_{k+1} = \mathbf{x}_k + p_k, \qquad \text{and} \qquad \mathbf{r}_{k+1} = \mathbf{r}_k - Ap_k$$

(in exact arithmetic), and the update Ap_k for $\mathbf{r}_k$ is computed from the update p_k of $\mathbf{x}_k$ by explicit matrix multiplication. In finite precision arithmetic, for some matrix Δ_A that may depend on k, we actually have that

$$\mathbf{r}_{k+1} = \mathbf{r}_k - Ap_k - \Delta_A p_k \qquad \text{with } \|\Delta_A\| \le n_A \bar{\xi} \| |A| \|, \tag{5}$$

where n_A is the maximum number of non-zero elements per row of A and the $|\cdot|$ refers to the element-wise absolute value. Therefore,

$$\begin{aligned} \|b - A\mathbf{x}_k - \mathbf{r}_k\| &\le \bar{\xi} n_A \| |A| \| \sum_{j<k} \|p_k\| \le \bar{\xi} \Gamma \sum_{j<k} \|Ap_j\| \\ &\le \bar{\xi}\, 2\Gamma \sum_{j\le k} \|\mathbf{r}_j\| \qquad \text{with } \Gamma := n_A \| |A| \| \|A^{-1}\|. \end{aligned} \tag{6}$$

We have skipped higher order terms in $\bar{\xi}$.

Except for the factor Γ this estimate seems to be sharp in practice. In order to understand why we do not see the factor Γ in actual computations, we slightly rewrite (5) as

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \mathrm{fl}(Ap_k) = \mathbf{r}_k - Ap_k - f_k.$$

We have that

$$|f_k| \le 1.01 \bar{\xi} n_A |A| |p_k|.$$

Since we are interested in the dominant parts of the error, we wonder how large the largest elements $(|A||p_k|)_j$ may be with respect to $(|Ap_k|)_j$.

In particular, we wonder whether it may happen that

$$\| |A| |p_k| \|_\infty \gg \|Ap_k\|_\infty.$$

We may expect significant influence of rounding errors if

$$\|Ap_k\|_\infty \ll \| \, |A| \, |p_k| \, \|_\infty \leq \| \, |A| \, \|_\infty \| \, |p_k| \, \|_\infty = \|A\|_\infty \|p_k\|_\infty,$$

or

$$\frac{\|Ap_k\|_\infty}{\|p_k\|_\infty} \ll \|A\|_\infty \approx |\lambda_{\max}|.$$

Now note that $\mathbf{r}_k = \psi_k(A)\phi_k(A)r_0$, and that the zeros of ϕ_k are the Ritz values (for Bi-CG: $\psi_k(t) = 1$, for CGS: $\phi_k = \psi_k$). In the 'normal' case the Ritz values tend to converge most rapidly to the extremal eigenvalues. That means that already after modest numbers of iteration steps we will see that ϕ_k and ϕ_{k+1} have nearby zeros, close to the extremal eigenvalues of A (extremal in the sense of close to the convex hull).

This implies that the major contributions of Ap_k are with respect to eigenvectors inside the hull (at least they may not be expected to be small).

Therefore,

$$\frac{\|Ap_k\|_\infty}{\|p_k\|_\infty} \approx \lambda_{\text{average}}$$

and in the context of discretized PDEs, this average λ will be not far from $\frac{1}{2}\|A\|_\infty$ (in a relative sense).

This means that the largest elements of the vector Ap_k are not much smaller than the largest elements of $\|A\| \, \|p_k\|$, and therefore the norm of f_k will be in the order of $2n_A \bar{\xi} \|Ap_k\|$, so that the factor "condition number" of A in the formula (6) for Γ will not show up (except possibly in rather special situations, for instance when there is not much reduction in the residual (Ap_k is small), but at the same time considerable variation in $\mathbf{x}_k$ (p_n not small)).

For the BiCGstab(l) methods we note that ψ_k is a product of low degree polynomials of which the zeros can be interpreted as harmonic Ritz values [13]. Also these harmonic Ritz values tend to converge most rapidly to extremal eigenvalues, so that the same way of reasoning as above carries over.

One may encounter sometimes convergence histories without large intermediate residuals and nevertheless still an inaccurate final residual, which seems to be in contradiction with our theory. This can happen for instance with BiCGstab(l), for larger values of l. The explanation for this is the following. For some methods, it is natural not to show all intermediate residuals; one may show only residuals at the end of a sweep that consists of more than one residual update. This is the case for BiCGstab(l), where the residuals are usually only reported at the end of a sweep of l internal updates. The shown residuals may then be small while the hidden intermediate ones may be large and this may explain the observed discrepancies.

Estimate (6) suggests a stopping criterion that may help to avoid superfluous steps (see also [18], Section 2.1):

$$\text{stop if } \|\mathbf{r}_k\| < tol \qquad \text{or} \qquad \|\mathbf{r}_k\| \leq \bar{\xi}\kappa \sum_{j<k} \|\mathbf{r}_j\|. \tag{7}$$

for some modest constant $\kappa \geq 1$. Throughout this paper κ wil always represent such a constant of modest size.

The local errors $\Delta_A p_j$ are relatively small ($\|\Delta_A p_j\| \leq \bar{\xi}\Gamma(\|\mathbf{r}_j\| + \|\mathbf{r}_{j+1}\|)$, but the global ones $b - A\mathbf{x}_k - \mathbf{r}_k$ can still be absolutely large with respect to, say, $\|\mathbf{r}_0\|$ (if $\|\mathbf{r}_j\| \gg \|\mathbf{r}_0\|$ for some $j \leq k$): we should avoid large local errors and, of lesser importance, an accumulation of local errors. *For an accurate residual we should have local errors that are globally absolutely* small (say[1], $\leq \kappa\bar{\xi}\|\mathbf{r}_0\|$).

We will suggest some modifications to existing algorithms that lead to locally relatively and globally absolutely accurate computations.

We proceed in two steps:

(a) In Section 3.2, we will explain how the approximations and the residuals can be updated by a process in which groups of updates are taken together so that the actual updating of the approximations takes place only every once in a while, such that the updated residuals reflect accurately the group-wise updated approximations.
(b) For Bi-CG type algorithms, it is tempting to replace the recursively computed residuals by the actual residuals. This would introduce relatively large local errors in the recursions, but, as we will show, this can be avoided by replacing occasionally the recursively computed residuals by "true" residuals. We will identify the proper stages in the iteration process where this can be done safely (Section 3.4).

Remark. We can not simply replace the recursively computed residuals by the "true" ones, that is, by those computed as $b - A\mathbf{x}_k$, because, if $\|\mathbf{r}_k\| \ll \|\mathbf{r}_0\|$, this approach would introduce relatively non-small errors (of order $\bar{\xi}\Gamma\|\mathbf{r}_0\|$) and may decelerate the speed of convergence.

3.2. Improving Accuracy by Group-Wise Updating

As an alternative for the usual updating process, we may compute the $\mathbf{x}_k$ and $\mathbf{r}_k$ as follows.

---

[1] Even if x is the exact solution of (1) then the residual computed from $b - Ax$ may be $\approx \kappa\bar{\xi}\|b\|$. For ease of presentation, we will assume that $\mathbf{x}_0 = 0$, $\mathbf{r}_0 = b$.

We select some increasing sequence $(\pi(j))$ in $\mathbb{N}$, $\pi(0) = 0$, and, for $k = \pi(j+1)$, compute $\mathbf{x}_k$ and $\mathbf{r}_k$ by a cumulative or group-wise update:

$$\begin{cases} \mathbf{x}_k = \mathbf{x}_{\pi(j)} + q_j & \text{where } q_j := x_k^{(j)} = p_{\pi(j)} + p_{\pi(j)+1} + \ldots + p_{k-1} \\ \mathbf{r}_k = \mathbf{r}_{\pi(j)} - Aq_j. \end{cases} \tag{8}$$

The $x_i^{(j)}$ can be computed iteratively: $x_{\pi(j)}^{(j)} = 0$ and $x_{i+1}^{(j)} = x_i^{(j)} + p_i$ if $\pi(j) \le i < \pi(j+1)$ (the intermediate approximations for x need not be computed).

Then, for $k = \pi(j+1)$,

$$\|b - A\mathbf{x}_k - \mathbf{r}_k\| \le \bar{\xi}\Theta \sum_{j,\, \pi(j)<k} \|Aq_j\|, \tag{9}$$

where $\Theta \le \Gamma$ (in relevant situations, often $\Theta = \mathcal{O}(1)$, see Section 3.1).

If we take $\pi(j)$ such that the accumulated values of q_j are not too large in a global sense (say, such that $\|Aq_j\| \le \kappa\|\mathbf{r}_0\|$), then we may expect an accurate residual. Of course, the intermediate $p_{\pi(j)+i}$ should not be too large either, since then the rounding errors in the addition process may accumulate and spoil the result q_j. We will return to this in Section 3.3.

However, for good convergence, q_j should be *locally* small too (say, $\|Aq_j\| \le \kappa\|\mathbf{r}_{\pi(j+1)}\|$). These requirements can be conflicting. For instance, if $\pi(1) = k$ and $\|\mathbf{r}_k\| \ll \|\mathbf{r}_0\|$ then $\|Aq_j\| = \|\mathbf{r}_0 - \mathbf{r}_k\| \approx \|\mathbf{r}_0\| \not\le \kappa\|\mathbf{r}_k\|$.

Neumaier [11] suggested a compromise for CGS (see also [3], Section 4), but it also seems to work well for other methods. He suggested to select the $\pi(j)$ such that

$$\|\mathbf{r}_{\pi(j+1)}\| < \|\mathbf{r}_{\pi(j)}\| \le \|\mathbf{r}_k\| \qquad \text{if } \pi(j) < k < \pi(j+1), \tag{10}$$

that is, the approximation $\mathbf{x}_{\pi(j)}$ is updated by the accumulation q_j of the local updates p_i as soon as the residual norm becomes smaller than $\mathbf{r}_{\pi(j)}$. Then the q_j can be expected to be small with respect to both the local and the global residuals. Similar effects can be accomplished with a sequence $\pi(j)$ such that

$$\|\mathbf{r}_{\pi(j+1)}\| < \|\mathbf{r}_0\| \qquad \text{and} \qquad \|\mathbf{r}_{\pi(j+1)-1}\| \ge \|\mathbf{r}_0\| \tag{11}$$

or any other strategy that updates by accumulated local updates when the norm of the residual $\mathbf{r}_{\pi(j+1}$ decreases with respect to $\mathbf{r}_0$ (for globally absolutely small errors) but also the norm of $\mathbf{r}_{\pi(j+1)-1}$ is not too small with respect to $\mathbf{r}_{\pi(j)}$ (for locally relatively small errors).

3.3. Considerations on the Norm of the Updates

In this section we will try to give an impression of the size of Ap_k in some realistic cases. This gives a motivation to take measures to prevent cancellation

effects to spoil the updated solution. Of course, it is necessary that the intermediate $p_{\pi(j)+i}$ be not too large either, for instance, if any of these vectors is much larger than, say, $\|r_0\|/\bar{\xi}$, then q_j will be large and $\|Aq_j\|$ will be large. We will now argue that this is normally not the case.

The residuals of Bi-CG, $r_i^{\text{Bi-CG}}$, satisfy the following relation [6]:

$$\|r_i^{\text{Bi-CG}}\| = \frac{1}{|c_i|}\|r_0\|\,|s_1 \cdot s_2 \cdots s_i|,$$

where c_i and s_i are the cosine and sine of the rotation that eliminates the bottom element in the tridiagonal Lanczos matrix at step k. Note, that when r_i is significantly smaller than r_{i-1} then $c_i \approx 1$. The sines describe essentially the reducation of the residual. We also have from [6] that

$$\|r_i^{\text{Bi-CG}}\| \leq \frac{1}{|c_i|}\sqrt{i+1}\,\|r_0\|.$$

This means that, unless $|c_i|$ is very small, $r_i^{\text{Bi-CG}}$ cannot be very large, and hence Ap_i cannot be very large (except when $r_i^{\text{Bi-CG}}$ is very small w.r.t. $r_{i-1}^{\text{Bi-CG}}$). The case where $|c_i|$ is very small is known as a near break-down situation and it should be cured appropriately, for instance with look-ahead techniques [5, 6].

In the Bi-CGSTAB algorithms Bi-CG is combined with low degree GMRES processes, which can only further decrease the norm of r_i, so that, apart from near break-down situations, the Ap_i cannot be very large either.

CGS has a bad reputation for its high peaks in its residual plots, and we wonder how high these peaks may be. Suppose that the v_j form a complete set of normalized eigenvectors of A, with associated eigenvalues λ_j (we assume A not to be nearly defect). Write $r_0 = \sum \gamma_j v_j$. Then with

$$r_i^{\text{Bi-CG}} = \phi_i(A)r_0$$

we have that $r_i^{\text{Bi-CG}} = \sum \phi_i(\lambda_j)\gamma_j v_j$, and hence

$$|\phi_i(\lambda_j)\gamma_j| \lesssim \|r_i^{\text{Bi-CG}}\|.$$

Therefore, $|\phi_i(\lambda_j)| \lesssim \|r_i^{\text{Bi-CG}}\|/\gamma_j$.

The worst case will be $\gamma_j = \bar{\xi}\|r_0\|$ (of course γ_j may be smaller in theory, but in the presence of rounding errors ours is a realistic worst case scenario). This leads to

$$|\phi_i(\lambda_j)| \lesssim \frac{1}{\bar{\xi}}\frac{\|r_i^{\text{Bi-CG}}\|}{\|r_0^{\text{Bi-CG}}\|},$$

so that for the residual components of CGS we have that

$$|\phi_i^2(\lambda_j)\gamma_j| \lesssim \frac{1}{\bar{\xi}^2}\left(\frac{\|r^{\text{Bi-CG}}\|}{\|r_0^{\text{Bi-CG}}\|}\right)^2 \bar{\xi}\,\|r_0^{\text{Bi-CG}}\|.$$

This means that

$$\|r_i^{\text{CGS}}\| \lesssim \frac{1}{\bar{\xi}} \left(\frac{\|r_i^{\text{Bi-CG}}\|}{\|r_0^{\text{Bi-CG}}\|} \right)^2 \|r_0^{\text{Bi-CG}}\|.$$

In practice, i can not be small if we encounter a large value for $\phi_i(\lambda_j)$. This means that the factor $\|r_i^{\text{Bi-CG}}\|/\|r_0^{\text{Bi-CG}}\|$ will not be very large typically. Nevertheless, apart from near break-down situations $\|r_i^{\text{CGS}}\|$ can be of the order of $\|r_0\|/\bar{\xi}$ at most. Hence, even in worst case situations we may expect that the group-wise update approach leaves just enough accuracy for accurate residuals.

3.4. Maintaining the Speed of Convergence

We have seen how to compute locally and globally accurate residuals for $k = \pi(j)$ from approximations and intermediate "search directions". However, it is not clear whether we may simply replace the residuals as computed by the algorithm (the *standard updated residuals*) by the residuals computed in this new way (the *group-wise updated residuals*). Indeed, by doing so, we introduce local errors in the recurrence relations: for maintaining convergence, the standard updated and the group-wise updated residuals should be locally relatively close together.

If we compute the residuals as $\mathbf{r}_k = \mathbf{r}_{\pi(j)} - Ax_k^{(j)}$ also for $k > \pi(j)$ and $k < \pi(j+1)$, and the $\pi(j)$ have been chosen strategically, then the local errors can be relatively small. Unfortunately, this approach may require many additional MVs (see also Section 4.6). On the other hand the use of the recurrence relations for $k \neq \pi(j)$ and of the relation (8) for $k = \pi(j+1)$ may introduce locally relatively large errors in the recurrence relations for the residuals (see Fig. 1). As a compromise we may compute $\mathbf{r}_k$ as the *true local residual* $\mathbf{r}_i = \mathbf{r}_{\pi(j)} - Ax_k^{(j)}$ for a few strategically chosen values of k, between $\pi(j)$ and $\pi(j+1)$, say, $k = k_0, \ldots, k_m$. The problem of how to choose these values is similar to the one we had before: using the recurrences, we update $\mathbf{x}_k$ and $\mathbf{r}_k$ (in exact arithmetic) as $\mathbf{x}_{k+1} = \mathbf{x}_k + p_k$, $\mathbf{r}_{k+1} = \mathbf{r}_k + Ap_k$. But now, $\mathbf{r}_{k_i}$ is the true local residual and the recursively computed "$\mathbf{r}_{k_{i+1}}$" should be relatively close to the "exact" one $\mathbf{r}_{\pi(j)} - Ax_{k_{i+1}}^{(j)}$ (the true local residual): "$\mathbf{r}_{k_{i+1}}$" is computed by the recurrence in a few steps from the "exact" $\mathbf{r}_{k_i}$.

Now, there are two sources of errors that are of interest: (i) as argued before, the updating steps can introduce relatively non-small errors and, (ii) the true local residual $\mathbf{r}_{k_{i+1}}$ that will replace the recursively computed residual can have a relatively non-small error. As we know, for locally relatively small errors (Sub. (i)), the size of the updates p_k, for $k = k_i, \ldots, k_{i+1} - 1$, should be relatively non-large with respect to $\|\mathbf{r}_{k_{i+1}}\|$. For a good true local residual $\mathbf{r}_{k_{i+1}}$ (Sub. (ii)), $\|\mathbf{r}_{\pi(j)}\|$ should be non-large as compared with $\|\mathbf{r}_{k_{i+1}}\|$. Hence, the sizes $\|\mathbf{r}_k\|$, for $k = k_i, \ldots, k_{i+1}$, *and* $\|\mathbf{r}_{\pi(j)}\|$ should be non-large as compared with $\|\mathbf{r}_{k_{i+1}}\|$. In particular, this means that it may harm to use true local residuals for $k >$

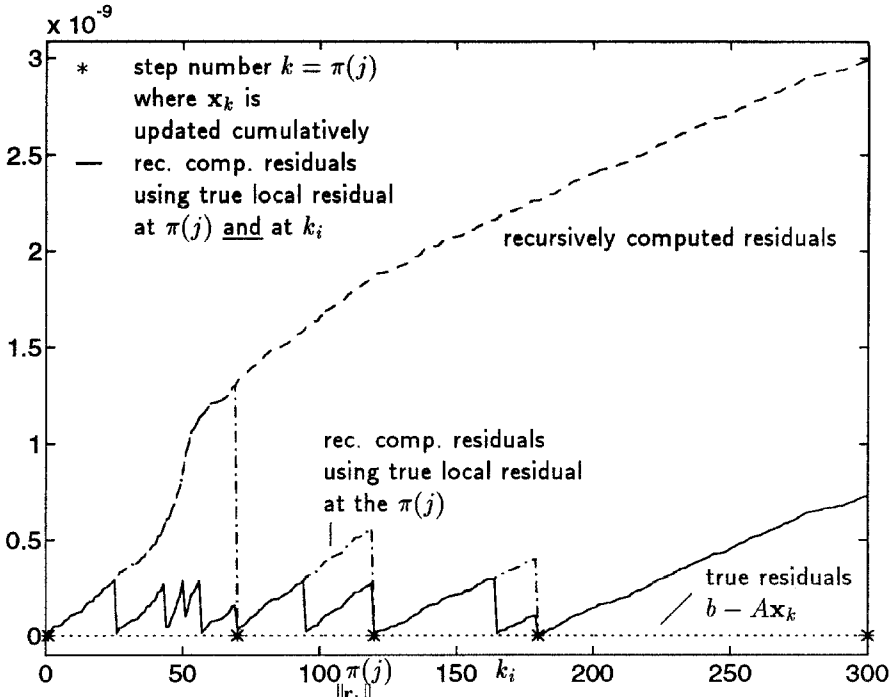

Figure 1. Idealized graphs of $\frac{\|\mathbf{r}_k\|}{\|b - A\mathbf{x}_k\|} - 1$, with residuals $\mathbf{r}_k$ computed in different ways

$\max_j \pi(j)$. We expect to maintain the speed of convergence, for instance, by computing and using the true local residual $\mathbf{r}_k = \mathbf{r}_{\pi(j)} - Ax_k^{(j)}$ for those

$$k \in (\pi(j), \pi(j+1)] \qquad \text{for which} \qquad \|\mathbf{r}_0\| \le \|\mathbf{r}_k\| < \|r_{k-1}\| \tag{12}$$

if the $\pi(j)$ are such that (11) holds.

Algorithm 1. The Bi-CGSTAB algorithm
Choose $\mathbf{x}_0$ **and** $\tilde{r}_0$. **Set** $\mathbf{x} = \mathbf{x}_0$.
Compute $\mathbf{b} = b - A\mathbf{x}_0$
$\mathbf{y} = 0, \quad \mathbf{r} = \mathbf{b}$
$\mathbf{u} = 0, \quad \omega = \sigma = 1$
While $\|\mathbf{r}\| > tol$ **do**
$\rho = (\mathbf{r}, \tilde{r}_0), \quad \beta = (-\rho)/(\omega\sigma)$
$\mathbf{u} = \mathbf{r} - \beta\mathbf{u}, \quad \mathbf{c} = A\mathbf{u}$
$\sigma = (\mathbf{c}, \tilde{r}_0), \quad \alpha = \rho/\sigma$
$\mathbf{y} = \mathbf{y} + \alpha\mathbf{u}$
$\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}, \quad \mathbf{s} = A\mathbf{r}$
$\omega = (\mathbf{r}, \mathbf{s})/(\mathbf{s}, \mathbf{s})$
$\mathbf{u} = \mathbf{u} - \omega\mathbf{c}$
$\mathbf{y} = \mathbf{y} + \omega\mathbf{r}$
$\mathbf{r} = \mathbf{r} - \omega\mathbf{s}$
end while
$\mathbf{x} = \mathbf{x} + \mathbf{y}$

3.5. Towards Smooth Convergence

The updating condition in (10) leads quite naturally to a smooth convergence history, since only the best results $\|\mathbf{r}_{\pi(j)}\|$ have to be plotted (the intermediate results now no longer affect the solution with accumulated rounding errors and they are not very relevant anymore) as well as to more accurate residuals. Other residual smoothing strategies (as in [21]), based on norm-minimization of convex combinations of residuals, do not always lead to more accurate residuals, because the intermediate updates may already have polluted the process. However, these strategies do not require additional MVs and a comparison would only be fair if the strategies in (10) and (11) can be performed efficiently too. We will address this issue in Section 4.6.

4. Efficient Methods for Improving the Accuracy

We will illustrate our strategies and conditions for Bi-CGSTAB (cf. Algorithm 1), but they can be applied to other Bi-CG based methods as well (cf. Section 2; see also [8]). As a matter of fact, they usually are more helpful for other methods, since Bi-CGSTAB produces most often already rather accurate residuals by itself.

Algorithm 2. A simple update strategy
set '*compute_res*' **and** '*flying_restart*'
if '*compute_res*' = '**true**'
    $\mathbf{r} = \mathbf{b} - A\mathbf{y}$,
    if '*flying_restart*' = '**true**'
        $\mathbf{x} = \mathbf{x} + \mathbf{y}$, $\mathbf{y} = 0$, $\mathbf{b} = \mathbf{r}$
    end if
end if
'*compute_res*' = '*flying_restart*' = '**false**'

4.1. Shifting the Problem

In Algorithm 1, we actually give the Bi-CGSTAB algorithm for a shifted problem, allowing us to assume that 0 is the initial guess: if $\mathbf{x}_k$ are the Bi-CGSTAB approximations with initial approximation $\mathbf{x}_0$ for the original problem "$Ax = b$", and $\mathbf{y}_k$ are the Bi-CGSTAB approximations with initial approximation $\mathbf{y}_0 = 0$ for the *shifted problem* $Ay = \mathbf{b} := b - A\mathbf{x}_0$, then

$$x = \mathbf{x}_0 + y, \qquad \mathbf{x}_k = \mathbf{x}_0 + \mathbf{y}_k \qquad \text{and} \qquad \mathbf{r}_k = b - A\mathbf{x}_k = \mathbf{b} - A\mathbf{y}_k.$$

The form in Algorithm 1 simplifies the explanation of our approach. In the first two lines of Algorithm 1 we shift the problem, in the third line we "select" our initial approximation $\mathbf{y}_0 = 0$ and "compute" the initial residual $\mathbf{r}_0 = \mathbf{b}$. In the last line, we "undo" the shift. We skipped the indices, since the new values are allowed to overwrite the old ones.

4.2. *Flying Restarts*

If a restart strategy is used, one also has to deal with group-wise updated approximations (cf. the last line in Algorithm 1). The convergence of the methods of interest tends to accelerate during the iteration (super-linear convergence). This speed-up may be lost when restarting in the obvious way (i.e., by resetting the vector $\mathbf{u}$ and the scalars ω and σ to trivial values). The vectors $\mathbf{r}$, $\mathbf{u}$ and the scalars ω, σ determine the speed of convergence (of $\mathbf{r}$ towards 0). At least in exact arithmetic, these quantities do not change if, by restart, the auxiliary quantities $\mathbf{u}$, ω and σ are *not* reset (or, equivalently, if the fourth line is skipped after restart). Therefore, after such a "*flying restart*", the same accelerated speed may be expected with continued acceleration as before the restart. Our strategy of group-wise updated approximations and residuals, as discussed in Section 3.2, can be viewed as a strategy for flying restarts.

4.3. *The Basic Strategy*

Now, our updating strategy is characterized by the following modification in Algorithm 1:

- At the end of the iteration (just before the 'end while') we update $\mathbf{x}$ and recompute $\mathbf{r}$ as in Algorithm 2, where '*compute_res*' and '*flying_restart*' are Boolean valued functions that depend on the norm of the residuals computed so far. '*compute_res*' has to be '**true**' when '*flying_restart*' is '**true**'. These conditions will be discussed in Section 4.4.

With "$\mathbf{x} = \mathbf{x} + \mathbf{y}$" the previous shift is undone, with "$\mathbf{b} = \mathbf{r}$" $(= \mathbf{b} - A\mathbf{y})$ the problem is shifted and "$\mathbf{y} = 0$" marks the restart. Since the auxiliary quantities $\mathbf{u}$, ω and σ are not reset, the new start is "flying". In exact arithmetic $\mathbf{b} - A\mathbf{y} = b - A\mathbf{x}$, but in finite precision arithmetic the differences can be very significant (if $\|\mathbf{b}\| \ll \|b\|$; cf. Section 3.2).

Eventually we are interested in the approximation $\mathbf{x}$. Shifting the problem (and, hence, our modification in Algorithm 2) requires two additional vector arrays, and, besides the additonal MV if '*compute_res*' is '**true**', it also requires three additional vector updates whenever '*flying_restart*' is '**true**'. Note that each flying restart requires the computation of a true local residual.

If the cumulative updating or flying restart has been performed j times ('*flying_restart*' has been '**true**' j times) then, the $\mathbf{y}$, $\mathbf{r}$, $\mathbf{x}$ and $\mathbf{b}$ are related to the quantities in Section 3 as follows:

$$\mathbf{x} = \mathbf{x}_{\pi(j)} \qquad \text{and} \qquad \mathbf{b} = b - A\mathbf{x} = \mathbf{r}_{\pi(j)},$$

and after k iteration steps:

$$k > \pi(j), \mathbf{y} = x_{k+1}^{(j)} = p_{\pi(j)} + \ldots + p_k, \qquad \text{and} \qquad \mathbf{r} = \mathbf{r}_{k+1},$$

where $p_i = \mathbf{x}_{i+1} - \mathbf{x}_i (= \mathbf{y}_{i+1} - \mathbf{y}_i)$ is the difference between two consecutive regular Bi-CGSTAB approximations $\mathbf{x}_i$ (or $\mathbf{y}_i$, in the ith step).

It would not be wise to update as in Algorithm 2 in each iteration (i.e., '*flying_restart*' = '*compute_res*' = '**true**'), since, as we have argued in Section 3, this does not improve the accuracy of the computed residual and it also requires an additional (expensive) MV in each step.

Our basic strategy with '*flying_restart*' = '**true**' at steps $\pi(j)$ as defined by condition (10) or (11) helps to improve the accuracy, while, in combination with a permanent condition '*compute_res*' = '**true**' or with '*compute_res*' = '**true**' in steps k as defined by (12), it may be expected to maintain the speed of convergence (although this may still be expensive).

We may try to limit the additional computational costs by a restrictive number of computations of true local residuals (see Section 3.4 and Section 4.5) or by a reformulation of the algorithm in order to gain one MV whenever a true local residual is desirable (see Section 4.6). However, in order to restrict the number of true local residuals we also have to restrict the number of flying restarts (as explained in Section 3.4).

4.4. Conditions for Flying Restarts

In the restart conditions below, $\mathbf{b}$ is the residual after the latest restart and we take the maximum of the norm of the residuals since the latest restart. All restarts are assumed to be "flying".

In the notation of this section and Section 4.3, condition (10) leads to

$$\text{if} \quad \|\mathbf{r}\| < \|\mathbf{b}\| \quad \text{then '}\textit{flying\_restart}\text{'} = \text{'}\mathbf{true}\text{'}, \tag{13}$$

and condition (11) leads to

$$\text{if} \quad \|\mathbf{r}\| < \|b\| \;\&\; \|b\| \leq \max\|\mathbf{r}_j\| \quad \text{then '}\textit{flying\_restart}\text{'} = \text{'}\mathbf{true}\text{'}. \tag{14}$$

Condition (14) will limit the number of restarts considerably as compared with condition (13), but this may still require too many additional MVs if the residuals converge very irregularly.

Therefore, we propose to perform the restart only when the norm of the residual has decreased significantly since the latest restart

$$\text{if} \quad \|\mathbf{r}\| < \delta\|\mathbf{b}\|, \quad \text{with, say } \delta = 10^{-2} \quad \text{then '}\textit{flying\_restart}\text{'} = \text{'}\mathbf{true}\text{'}, \tag{15}$$

Algorithm 3. MV-saving strategy for CGS
set '*compute_res*' **and** '*flying_restart*'
if '*compute_res*' **is 'false'**
$\mathbf{r} = \mathbf{r} - Ap$

else
  $\mathbf{r} = \mathbf{b} - A\mathbf{y}$,
  if '*flying_restart*' '**true**'
    $\mathbf{x} = \mathbf{x} + \mathbf{y}$, $\mathbf{y} = 0$, $\mathbf{b} = \mathbf{r}$
  end if
end if
'*compute_res*' = '*flying_restart*' = '**false**'

or when the norm of one of the residuals since the last restart has been larger than $\|b\|$ and when there has been a significant decrease since then,

$$\text{if} \quad \|\mathbf{r}\| < \delta\|b\| \quad \text{and} \quad \|b\| \leq \max\|\mathbf{r}_j\| \quad \text{then '}\textit{flying\_restart}\text{'} = \text{'}\mathbf{true}\text{'}. \tag{16}$$

4.5. Conditions on When to Compute the True Residual

One may compute the *true local residual* $\mathbf{r} = \mathbf{b} - A\mathbf{y}$ in each step (as Neumaier [11] has suggested for CGS):

$$\text{'}\textit{comput\_res}\text{' is always '}\mathbf{true}\text{'}. \tag{17}$$

However, in combination with our update strategy in Algorithm 2, this condition will increase the number of MVs by 50%. A way out is to impose the following condition (in combination with (16)), which may be true only a few times during a run and it still maintains local accurate computations:

$$\text{if} \quad \left\{ \begin{array}{l} \|\mathbf{r}\| < \delta \max\|\mathbf{r}_j\| \;\&\; \|b\| \leq \max\|\mathbf{r}_j\| \\ \text{or '}\textit{flying\_restart}\text{'} = \text{'}\mathbf{true}\text{'} \end{array} \right\} \text{ then '}\textit{compute\_res}\text{'} = \text{'}\mathbf{true}\text{'}, \tag{18}$$

where the maximum is taken over all residual norms since the previous computation of a true local residual. This condition slightly relaxes (12), just as (16) relaxes (14).

The function '*compute_res*', as defined by (18), depends on $\|\mathbf{r}\|$, but we may not have computed (approximately) $\mathbf{r}$ so far (as will be the case in our strategy in Section 4.6, Algorithm 3). In our experiments, we have used the norm of the previous residual. We may expect that this will work fairly well unless there is a huge difference between the norms of two consecutive residuals.

4.6. Saving Matrix-Vector Multiplications

In a cumulative update version of CGS, we may simply *replace* the line "$\mathbf{r} = \mathbf{r} - Ap$", where the residual $\mathbf{r}$ is updated by some linear combination p of two search directions, by the lines in Algorithm 3. This strategy is very efficient even when '*compute_res*' is always '**true**': the MV 'Ap' is replaced by another MV '$A\mathbf{y}$'.

Neumaier [11] suggested the strategy in Algorithm 3 for CGS, with '*compute_res*' always '**true**', and imposing (13) on '*flying_restart*'.

A similar MV-saving modification can be incorporated in other Bi-CG based type algorithms, but one has to proceed with a little more care. For instance, if we would like to replace the line "$r_{k+1} = r_k - \alpha_k A u_k$" in Bi-CG (see, (2)) by a "conditional" "$r_{k+1} = \mathbf{b} - A\mathbf{y}$", then we should realize that we need Au_k anyway, since the usual formulation of Bi-CG requires Au_k for the update of $\mathbf{y}$ via α_k and σ_k. However, note that $\sigma_k = (Au_k, \tilde{r}_k) = (u_k, A^T\tilde{u}_k)$ so that σ_k can also be computed from u_k and $A^T\tilde{u}_k$. This last vector is availabe in Bi-CG since it is required for the update of the shadow residual $\tilde{r}_k$.

In Bi-CGSTAB, two lines are candidate for replacement by a conditional update. Since we need $\mathbf{s}$ for ω (see Algorithm 1), since ω is required for updating $\mathbf{y}$, the line "$\mathbf{r} = \mathbf{r} - \omega\mathbf{s}$" does not seem to be accessible for modifications. As for the other line "$\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}$", we have to figure out how to compute σ without $\mathbf{c}$. To this end we note that because $\sigma = (\mathbf{u}, A^T\tilde{r}_0)$ we need only $\mathbf{u}$ and an additional vector $\tilde{s} := A^T\tilde{r}_0$ that has to be stored[2]. This vector needs to be computed only once in the initialization phase of the process.Once we have $\mathbf{r}$, as $\mathbf{r} = \mathbf{r}_{\text{new}} = \mathbf{b} - A\mathbf{y}$ and $\mathbf{y} = 0$, $\mathbf{b} = \mathbf{r}_{\text{new}}$, we can compute $\mathbf{c}$ at relatively low costs as $\mathbf{c} = (\mathbf{r}_{\text{old}} - \mathbf{r}_{\text{new}})/\alpha$. So, in addition to our earlier modifications in the initialization part and at the end, we propose to add $\tilde{s} = A^T\tilde{r}_0$ to the second line and to *replace* the statements "$\mathbf{c} = A\mathbf{u}$",..., "$\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}$" in the Bi-CGSTAB algorithm (In Algorithm 2) by the lines in Algorithm 4.

Algorithm 4. MV-saving strategy for BiCGstab(l)

$\sigma = (\mathbf{u}, \tilde{s})$, $\alpha = \rho/\sigma$
$\mathbf{y} = \mathbf{y} + \alpha\mathbf{u}$
set '*compute_res* **and** '*flying_restart*'
if '*compute_res*' = '**false**'
  $\mathbf{c} = A\mathbf{u}$, $\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}$,
else
  $\hat{\mathbf{r}} = \mathbf{b} - A\mathbf{y}$
  $\mathbf{c} = (\mathbf{r} - \hat{\mathbf{r}})/\alpha$, $\mathbf{r} = \hat{\mathbf{r}}$
  if '*flying_restart*' = '**true**'
    $\mathbf{x} = \mathbf{x} + \mathbf{y}$, $\mathbf{y} = 0$, $\mathbf{b} = \mathbf{r}$
  end if
end if
'*compute_res*' = '*flying_restart*' = '**false**'

BiCGstab(l) can be implemented in several ways, as explained in [18]. The strategy of conditional updating, as proposed above for Bi-CGSTAB, can be

[2] ρ can be computed from $\mathbf{r}$ and $\tilde{s}$ too. Therefore, this approach does not require additional vector storage.

used for any of these implementations (perform the conditional updating and adapted computation of $A\mathbf{u}$ for the first residual and the first $A\mathbf{u}$ that has to be computed in the Bi-CG part of the BiCGstab(l) sweep).

It works well for the orthonormal basis approach (see [18], Section 5) in the sense that it does not seem to affect the speed of convergence, while it improves the accuracy of the computed residual. For this version we even do not have to compute and store an additional vector $\tilde{s} = A^T \tilde{r}_0$, since we can use one of the available l initial shadow residuals $\tilde{r}_0, \ldots, \tilde{r}_{l-1}$ (take $\tilde{s} = \tilde{r}_1$).

For the power basis variant (see [18], Section 4 and [16]), our modification does not lead to any improvement (see the left figure in Fig. 5 in Section 5.1). This can be understood as follows. While, in the orthonormal basis variant, each $\mathbf{u}_1 = A\psi_k(A)u_{k+j}$, with $k = ml$ and $j = 0, \ldots, l-1$, is computed by explicitly multiplying $\mathbf{u} = \psi_k u_{k+j}$ by A, it is computed recursively in the power basis variant as:

$$
\begin{aligned}
&(\mathbf{u} = \psi_k(A)u_{k-1}, \mathbf{r} = \psi_k(A)r_k) \\
&\mathbf{u} = \mathbf{u} - \beta\mathbf{u}, \mathbf{u}_1 = A\mathbf{u}, \\
&\text{for } j = 1, \ldots, l-1 \\
&\quad \mathbf{u} = \mathbf{r} - \beta\mathbf{u}, \\
&\quad \mathbf{u}_1 = \mathbf{r}_1 - \beta\mathbf{u}_1,
\end{aligned}
$$

where the $\mathbf{r} = \psi_k(A)r_{k+j}$ and $\mathbf{r}_1 = A\psi_k(A)r_{k+j}$ are computed simultaneously, using similar recursive relations. Only the first $\mathbf{u}_1 (= A\psi_k(A)u_k)$ and the first $\mathbf{r}_1$ $(= \psi_k(A)r_{k+1})$ are computed by explicit matrix multiplication. This computational scheme decouples the updating of the approximation and the residual: $\mathbf{y}$ is updated by $\alpha\mathbf{u}$ and $\mathbf{r}$ by $\alpha\mathbf{u}_1$ rather than by $\alpha A\mathbf{u}$. Replacing the first $\mathbf{u}_1$ by $(\mathbf{r}_{\text{old}} - \mathbf{r}_{\text{new}})/\alpha_k$ shifts the accumulated inaccuracy in $\psi_k(A)r_k$ to $\psi_k(A)r_{k+2}$. Because, the new $\mathbf{u}_1$ contains the accumulated inaccuracies and this $\mathbf{u}_1$ and $A\mathbf{u}$ may differ significantly. Consequently, the next $\mathbf{u}_1$ and A times the next $\mathbf{u}$ differ significantly, resulting in an inaccurate next residual. Since in this version, old inaccuracies are shifted (to the search directions) rather than diminished, a further degradation in the speed of convergence may be expected.

5. Numerical Experiments

5.1. The Effect of Updating Strategies and Conditions

In this section we illustrate by numerical experiments the effects of our update strategies and conditions. We have tested them on a number of problems from the Harwell-Boeing collection (matrices A plus a right hand side vector b), and from example sections in [4, 10, 14]. Since all our experiments led to essentially similar conclusions, we will only show the results for one problem: Sherman4 from the Harwell-Boeing collection.

Of course, for some examples and some methods, the improvements are more impressive than for others. How much the improvement may be depends on how accurate the method is itself, without modification. Since, in this study, we are interested in inexpensive modifications to existing methods, we did not attempt to identify the best iterative method nor did we try to find a good preconditioner. In fact, we have not used preconditioning in these experiments. We did our experiments in MATLAB with $\bar{\xi} = 2.2\ 10^{-16}$.

The figures show the convergence histories of the *true* residuals on a logarithmic scale (that is, they show the graphs of $\log_{10}\|b - A\mathbf{x}\|$) as a function of the number of MVs (2 MVs for one CGS sweep; none of the additional MVs for computing true residuals have been taken into account). As an initial approximation we took $\mathbf{x}_0 = 0$, and for the (initial) shadow residual $\tilde{r}_0$ in the BiCG-type methods we simply took the initial residual $\tilde{r}_0 = \mathbf{r}_0 = b - A\mathbf{x}$.

Figures 2–4 illustrate the effect of different updating conditions. We present the results only for CGS. Since CGS amplifies the residuals in the initial phase of the process and exhibits an irregular convergence behavior (as is quite typical for CGS) it is well suited for demonstrating the characteristics of the updating conditions, but our conclusions hold for other methods as well (although the effects may be less pronounced). We return to this issue when discussing Fig. 5–6 below.

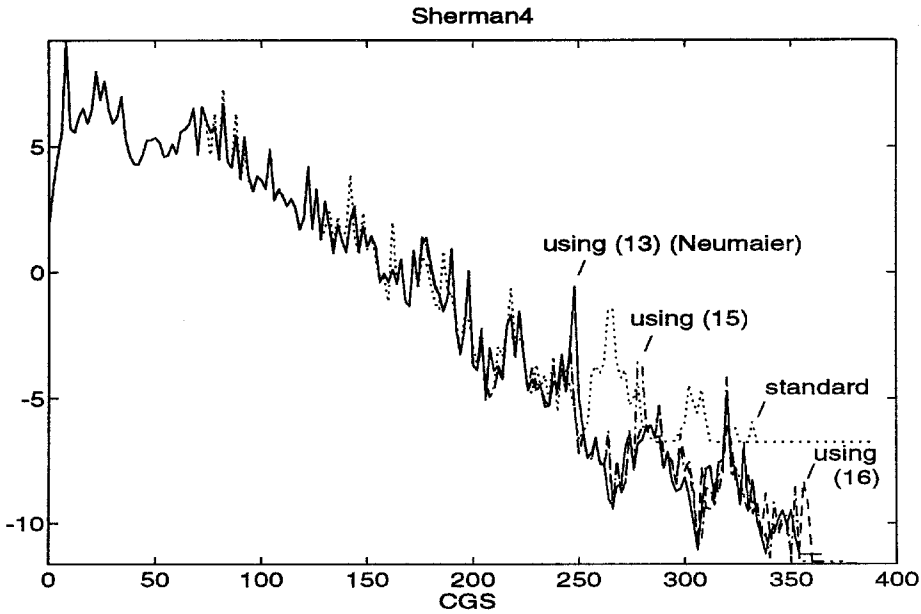

Figure 2. Convergence history of the true residuals of standard CGS and of CGS versions with true local residuals and various conditions for flying restarts

The dotted curve ('·····') in Fig. 2 shows the convergence history of the unmodified CGS process. Due to the large initial residuals (of $\approx 10^{+9}$) there is a loss of 7 digits ($\|b - A\mathbf{x}_k\| \approx 10^{-7}$ instead of $\approx \bar{\xi}\|b\| \approx 10^{-14}$). The other curves show the convergence history of CGS modified as in Algorithm 3 with '*compute_res*' is '**true**' in each step. For the condition for flying restart ('*flying_restart*' is '**true**') we took (13) (solid curve '——'; Neumaier approach), (15) (dash-dot curve '-·-·-') and (16) (dashed curve '-----'). Apparently, all restart conditions lead to the same fast convergence. Note that in the modifications that we discuss here (where $\mathbf{r} = b - A\mathbf{x}$) errors in $\mathbf{x}$ may affect the speed of convergence, in contrast to the situation for the unmodified CGS algorithm. All the updating conditions also lead to similar accurate residuals ($\|b - A\mathbf{x}_k\| \approx 10^{-12}$). However, there is some difference in efficiency: the number of flying restarts (that is, additional vector updates) is 29, 5, 1, respectively.

In Figs. 3 and 4, we report on the effects of different conditions for the replacement of the recursively computed residual by true local residuals.

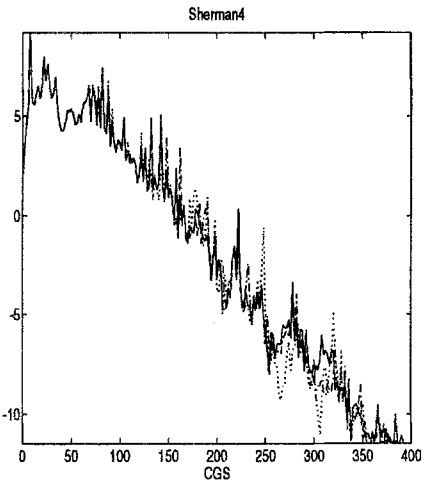

Figure 3. CGS versions with a few flying restarts and various conditions for true local residuals with simple and with the MV-saving strategy

The dotted curve ('·····') corresponds with Neumaier's approach: the MV saving modification in Algorithm 3 with conditions "'*compute_res*' = '**true**' in each step" and (13). This curve is given here as a reference.

In Fig. 3 we have used the MV-saving strategy in Algorithm 3 (the solid curve '——') and the simple strategy of Algorithm 2 (the dash-dot curve '-·-·-'). In both cases we have combined this strategy with the "most inexpensive" conditions (16) and (18). All shown approaches perform equally well as far as it concerns speed of convergence and accuracy of the residuals. However, the number of

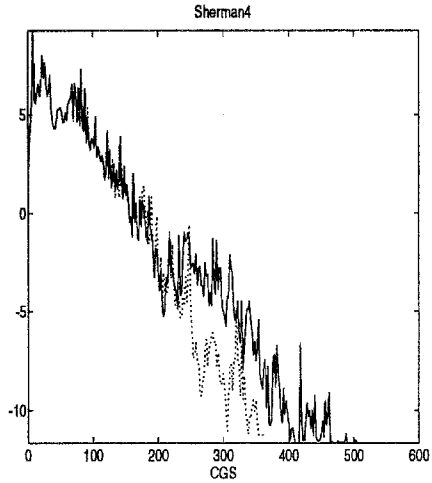

Figure 4. The CGS version with the MV saving strategy using a few flying restarts without any intermediate true local residual

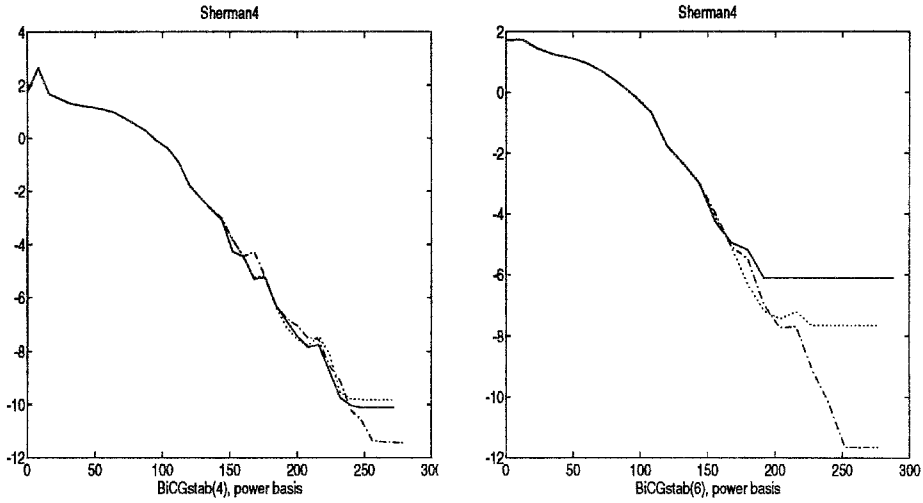

Figure 5. Flying restart with simple and with the MV-saving strategy for the power bases version of BiCGstab(4) (left) and BiCGstab(6) (right)

additional residual computations $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ dropped dramatically from 364 (if '*compute_res*' is always '**true**') to 7 (if condition (18) holds) for both strategies.

For the MV-saving strategy in Algorithm 3 there is no gain in efficiency, but for the simple strategy of Algorithm 2 the gain is considerable. Since we can apply this simple strategy also for other iterative methods, our conclusion is relevant: at the cost of a few (7 in this case) additional MVs, we may improve the accuracy of the residual considerably.

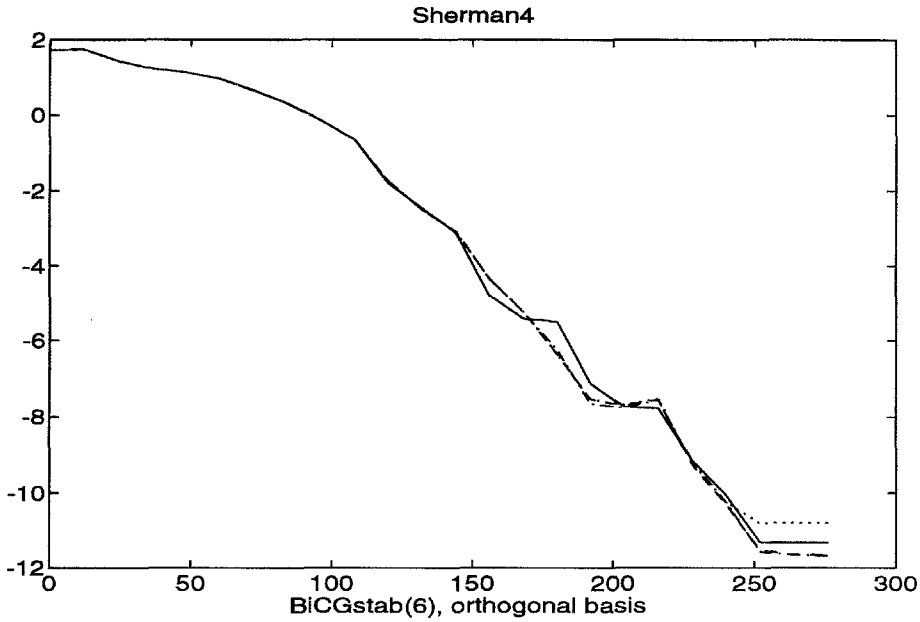

Figure 6. Flying restart with the simple and with the MV-saving strategy for the orthogonal bases version of BiCGstab(6)

In Fig. 4, we have applied the MV-saving strategy and we performed flying restarts under the condition (16). We have allowed '*compute_res*' to be '**true**' if and only if '*flying_restart*' is '**true**' (solid curve '——'). Locally non-small (in a relative sense) errors may be introduced, through the new true local residuals, in the recursions, and the speed of convergence may be affected (see the discussion on (ii) in Section 3.4). Indeed, as we can see from the figure, there is some degradation in the speed of convergence.

The convergence histories of BiCGstab(l) for Sherman4 do not show large residuals (see Figs. 5–6). However, especially for the power basis variant there may be large intermediate residuals (cf. [18], Section 3.3), and they are not shown in the figures. Computation of the norm of these residuals would have required one additional inner product per 2 MV. Therefore, it is normally not natural to present these intermediate norms. Nevertheless, these large residuals affect the accuracy and our updating strategies seem to work well even without computing the intermediate norms.

As argued in Section 4.6, the power basis variant of BiCGstab(l) is not suitable for the MV-saving strategy in Algorithm 4 (the solid curves '——' in Fig. 5), but the simple strategy of Algorithm 2 (the dash-dot curves '-·-·-' in Fig. 5) works well. In both figures in Fig. 5 the dotted curve ('·····') represents the convergence history of the unmodified BiCGstab(l) algorithm. We used the "most inexpensive" updating conditions (16) and (18). We only had to compute the

true local residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ once and we had to perform a flying restart only once for both strategies.

The additional costs, even for the simple strategy, are insignificant (1 MV and 1 vector update) but the gain in accuracy is considerable.

BiCGstab(l), in the orthogonal basis variant, is rather accurate without any modification (see [18]), but still we may gain a few digits almost for free by modifying the methods. Figure 6 shows the convergence history of BiCGstab(6) without modification (the dotted curve '·····') with the MV-saving strategy in Algorithm 4 and expensive conditions (13) and (17) (the solid curve '——'), with the MV-saving strategy and 'inexpensive' conditions (16) and (18) (the dash-dot curve '-·-·-·'), and with the simple strategy and cheap conditions (the dashed curve '-----'; here, the dash-dot curve '-·-·-' and the dashed curve '-----' coincide). The 'inexpensive' conditions were fulfilled only once during the whole run.

5.2. A Stopping Criterion

Figure 7 shows the convergence history of the true and computed residuals of CGS (unmodified), applied to the Sherman4 problem: the solid curve ('——') for the true residual, and the circles ('∘') for the computed residual. The dotted curve ('...') represents the values of $2\bar{\xi}\Sigma_{j<k}\|\mathbf{r}_j\|$ as a function of k (= 0.5# MV) on a logarithmic scale. Criterion (7) (with, say, $tol \leq 10^{-14}$) would have termi-

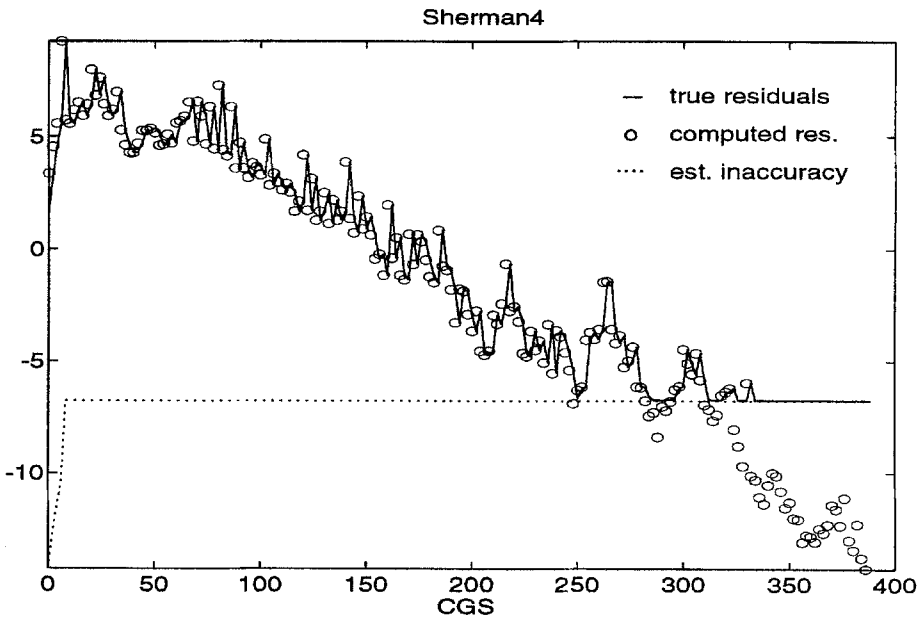

Figure 7. Estimating the inaccuracy in the computed residual of standard CGS

nated the iteration at about 250 MVs. As the figure shows, the CGS algorithm could not improve the approximation after that point.

6. Concluding Remarks

The combination of updating conditions (13) and (17) together with the updating strategy in Algorithm 3 of Neumaier is very attractive for CGS and Bi-CG. It can also successfully be incorporated in Bi-CGSTAB and in the orthonormal basis variant of BiCGstab(l).

For Bi-CGSTAB this would require a multiplication by the transpose of A, which might be a serious disadvantage. Since the BiCGstab(l) methods converge rather smoothly (in contrast to CGS and Bi-CG), the combination of updating conditions (as in (16) and (18)) may be expected to be fulfilled only a few times in one convergence history. Therefore, a simple updating strategy as in Algorithm 2, is attractive as well. The power basis variant of BiCGstab(l) may have large intermediate residuals and condition (15) in combination with (18) may do better in this case.

Of course, there is no need for special update schemes if the BiCGstab(l) methods produce accurate residuals (as is often the case).

Acknowledgement

The first author gratefully acknowledges stimulating and inspiring discussions with Jan Modersitzki.

References

[1] Bai, Z.: Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem. Math. Comp. *62*, 209–226 (1994).
[2] Fletcher, R.: Conjugate gradient methods for indefinite systems. In: Proc. of the Dundee Biennial Conference on Numerical Analysis (Watson, G., ed.), pp. 73-89. New York: Springer 1975.
[3] Fokkema, D. R., Sleijpen, G. L. G., van der Vorst, H. A.: Generalized conjugate gradient squared. Preprint 851, Dept. Math., University Utrecht (1994).
[4] Freund, R. W.: A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, SIAM J. Sci. Comput. *14*, 470–482 (1993).
[5] Freund, R. W., Gutknecht, M. H., Nachtigal, N. M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. SIAM J. Sci. Comput. *14*, 137–158 (1993).
[6] Freund, R. W., Nachtigal, N. M.: QMR: a quasi-minimal residual method for non-Hermitian linear systems. Numer. Math. *60*, 315–339 (1991).
[7] Greenbaum, A.: Behavior of slightly perturbed Lanczos and conjugate gradient recurrences. Linear Algebra Appl. *113*, 7–63 (1989).
[8] Greenbaum, A.: Estimating the attainable accuracy of recursively computed residual methods. Preprint, Courant Institute of Math. Sc., 1995.
[9] Lanczos, C.: Solution of systems of linear equations by minimized iteration. J. Res. Nat. Bur. Stand. *49*, 33–53 (1952).
[10] Meier Yang, U.: Preconditioned conjugate gradient-like methods of nonsymmetric linear systems. Preprint, Center for Research and Development, University of Illinois at Urbana-Champaign, 1992.

[11] Neumaier, A.: Oral presentation at the Oberwolfach meeting "Numerical Linear Algebra". Oberwolfach, April 1994.
[12] Paige, C. C.: Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. Linear Algebra Appl. *34*, 235–258 (1980).
[13] Paige, C. C., Parlett, B. N., Van der Vorst, H. A.: Approximate solutions and eigenvalue bounds from Krylov subspaces. Numer. Lin. Alg. Appl. *2*, 115–134 (1995).
[14] Saad, Y.: A flexible inner-outer preconditioned GMRES algorithm. SIAM J. Sci. Statist. Comput. *14*, 461–469 (1993).
[15] Saad, Y., Schultz, M. H., GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear system. SIAM J. Sci. Stat. Comp. *7*, 856–869 (1986).
[16] Sleijpen, G. L. G., Fokkema, D. R.: BiCGstab(l) for linear equations involving matrices with complex spectrum. Elect. Trans. Numer. Anal. (ETNA) *1*, 11–32 (1993).
[17] Sleijpen, G. L. G., Van der Vorst, H. A.: Maintaining convergence properties of BiCGstab methods in finite precision arithmetic. Preprint Nr. 861, Dept. Math., University Utrecht, 1994. To appear in Numerical Algorithms.
[18] Sleijpen, G. L. G., Van der Vorst, H. A., Fokkema, D. R.: BiCGstab(l) and other hybrid Bi-CG methods. Numer. Alg. *7*, 75–109 (1994).
[19] Sonneveld, P.: CGS, a fast Lanczos-type solver for nonsymmetric linear systems. SIAM J. Sci. Stat. Comp. *10*, 36–52 (1989).
[20] Van der Vorst, H. A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. *13*, 631–644 (1992).
[21] Zhou, L., Walker, H. F.: Residual smoothing techniques for iterative methods. SIAM J. Sci. Comput. *15*, 297–312 (1994).

Gerard L. G. Sleijpen
Henk A. van der Vorst
Mathematical Institute
University Utrecht
P.O. Box 80.010
NL-3508 TA Utrecht
The Netherlands
E-mail: sleijpen@math.ruu.nl vorst@math.ruu.nl