UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

A Composite Step Conjugate Gradients Squared Algorithm for Solving Nonsymmetric Linear Systems

Tony F. Chan
Tedd Szeto

July 1993
CAM Report 93-27

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

A COMPOSITE STEP CONJUGATE GRADIENTS SQUARED ALGORITHM FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS

TONY F. CHAN* AND TEDD SZETO†

Abstract. We propose a new and more stable variant of the CGS method [27] for solving nonsymmetric linear systems. The method is based on squaring the Composite Step BCG method, introduced recently by Bank and Chan [1, 2], which itself is a stabilized variant of BCG in that it skips over steps for which the BCG iterate is not defined and causes one kind of breakdown in BCG. By doing this, we obtain a method (Composite Step CGS or CSCGS) which not only handles the breakdowns described above, but does so with the advantages of CGS, namely, no multiplications by the transpose matrix and a faster convergence rate than BCG. Our strategy for deciding whether to skip a step does not involve any machine dependent parameters and is designed to skip near breakdowns as well as produce smoother iterates. Numerical experiments show that the new method does produce improved performance over CGS on practical problems.

AMS(MOS) subject classification. 65F10,65F25

Key Words. Lanczos method, conjugate gradients squared algorithm, breakdowns, composite step

1. Introduction.

Recently, there has been much activity in solving large linear systems

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

using iterative methods. Many of these are based on the classical Conjugate Gradient (CG) method due to Hestenes and Stiefel [20], which solves symmetric and positive definite systems using short recurrences, thereby minimizing work and storage, and also has an optimal convergence rate due to a minimization of the residual in a given Krylov subspace. For nonsymmetric systems, the Conjugate Gradients Squared (CGS) algorithm, developed by Sonneveld [27], has proven to be an attractive method with many advantages. Since it is based on the squaring of the residual polynomial of the Bi-conjugate Gradient (BCG) algorithm [22], it maintains simple and short recurrences but also has the added advantage of being "transpose-free" (i.e., not requiring multiplications by the transpose of matrix A). Moreover, with every two matrix-vector multiplies in the CGS algorithm, we can advance two degrees of the Krylov subspace whereas BCG advances only one degree with two matrix-vector multiplications. However, since CGS is based on BCG, it inherits some of the latter's disadvantages as well. Specifically, CGS encounters breakdowns (divisions by 0) under the same conditions as for BCG, and the squaring of the residual polynomial also amplifies the erratic residual convergence behavior in BCG causing worse numerical stability in practice [29].

Our goal is to overcome these disadvantages as best we can while retaining the good properties. The problem of breakdowns in BCG has received a lot of study in the

* Dept. of Mathematics, Univ. of Calif. at Los Angeles, Los Angeles, CA 90024. E-mail: chan@math.ucla.edu. Partially supported by the Office of Naval Research grant N00014-92-J-1890, the National Science Foundation grant ASC92-01266, and the Army Research Office grant DAAL03-91-G-150.

† Dept. of Mathematics, Univ. of Calif. at Los Angeles, Los Angeles, CA 90024. E-mail: szeto@math.ucla.edu. Supported by same grants as the first author.

literature. There are essentially two kinds of breakdowns: one due to the nonexistence of the residual polynomial as defined by the Galerkin condition in defining the BCG iterates (since this breakdown is implicitly caused by encountering a zero pivot in the factorization of the tridiagonal matrix generated in the underlying Lanczos process, we call this a *pivot breakdown*), and the other is due to a breakdown of the particular recurrence in BCG to compute this polynomial (since this kind of breakdown is directly related to the breakdown of the underlying Lanczos process, we call this a *Lanczos breakdown*). [1] Assuming no Lanczos breakdown, the pivot breakdown is easy to fix and can be removed by a 2×2 composite step in the CSBCG algorithm, recently developed by Bank and Chan [1, 2]. The unnormalized BIORES algorithm of Gutknecht's [19] also cures this breakdown using a three-term recurrence. Lanczos breakdowns are harder to fix and many look-ahead methods have been proposed to remedy them as well, see e.g. Freund, Gutknecht and Nachtigal [16], Brezinski, Redivo-Zaglia and Sadok [8, 9], Brezinski and Sadok [3], Joubert [21], and Parlett et al [25]. Although the step size needed to overcome an exact breakdown can be computed in principle, these techniques can unfortunately be quite complicated for handling near breakdowns since the sizes of the look-ahead steps are variable (indeed, the breakdown can be *incurable*) and in practice has to be determined by carefully designed heuristics often involving user specified tolerances.

In this paper, we "square" the CSBCG method to produce a new Composite Step CGS (CSCGS) algorithm. We also propose a practical adaptive stepping strategy to cure both exact and near pivot breakdowns, as well as to provide some smoothing effect to the residuals. CSCGS, then, is a transpose-free method that can overcome near pivot breakdowns in CGS while preserving the short recurrence and transpose-free properties of CGS. Existing implementations of CGS can be easily modified to incorporate CSCGS, as a maximum look-ahead step size of 2 is sufficient. We also apply the Minimal Residual Smoothing algorithm of Schönauer and Weiss [26, 30, 31] to the CSCGS iterates in order to further smooth the convergence behavior.

Several related algorithms have been recently proposed to cure breakdowns in CGS. By squaring the unnormalized BIORES algorithm, Gutknecht [19] derived an unnormalized $BIORES^2$ algorithm which overcomes exact pivot breakdowns in CGS. The MRZS method (and several variants) proposed by Brezinski and Sadok [4] goes a step further and overcomes exact Lanczos breakdowns as well (with the exception of incurable breakdown). Brezinski and Redivo-Zaglia are currently developing practical heuristics for overcoming near breakdowns in CGS [7]. Extensive testing and comparisons among these methods remains to be done.

We note that all look-ahead versions of a basic method are essentially equivalent in the sense that, given the same step size, they all compute the same uniquely defined iterate at the end of the look-ahead step. The different methods differ only in the details of the stepping strategy and the recurrences used in carrying out the step. The composite step approach seeks to achieve this with a minimal modification of the usual implementation of BCG and CGS, in the hope that the empirically observed numerical stability of these methods will be inherited. In trying to cure only one of the two possible breakdowns in CGS, we make a conscious decision in favor of having a simpler modification of CGS instead of a version which is less prone to breakdown

[1] In other literature, what we term the *pivot* and *Lanczos* breakdowns, are also known as *true* and *ghost* breakdowns [5], *Galerkin* and *serious Lanczos* breakdowns [15], *hard* and *soft* breakdowns [21].

but more complicated. Thus, the CSCGS method should be viewed as one in a spectrum of methods with varying degrees of breakdown protection and complexities of implementation.

We begin this paper by giving a brief review of the CSBCG method in section 2, followed by the description of CSCGS (section 3). Section 4 gives further details of implementation including the decision strategy for when to take a composite step. This is an important practical issue; the strategy we have chosen does not require any user specified tolerance parameters, thus further simplifying the method. In section 5, we discuss Minimal Residual Smoothing and its effect on the convergence behavior of CSCGS, and finally, results of numerical experiments are reported in section 6.

2. Composite Step BCG. Given initial guesses of x_0 and $\tilde{x}_0$ to the solutions of linear systems $Ax = b$ and $A^T\tilde{x} = \tilde{b}$, respectively, the BCG algorithm produces iterates of the form

$$x_n = x_0 + y_n; \qquad \tilde{x}_n = \tilde{x}_0 + \tilde{y}_n,$$

with residuals

$$r_n = b - Ax_n; \qquad \tilde{r}_n = \tilde{b} - A\tilde{x}_n,$$

where

$$y_n \in K_n(r_0) = span\{r_0, Ar_0, \ldots, A^{n-1}r_0\};$$

$$\tilde{y}_n \in K_n^*(\tilde{r}_0) = span\{\tilde{r}_0, A^T\tilde{r}_0, \ldots, (A^T)^{n-1}\tilde{r}_0\},$$

and such that the following Galerkin conditions are satisfied:

$$r_n \perp K_n^*; \qquad \tilde{r}_n \perp K_n.$$

One standard way to *implement* the algorithm is as follows [22, 13]:

Algorithm BCG

$$\begin{array}{lll}
 & r_0 = b - Ax_0; & \tilde{r}_0 = \tilde{b} - A^T\tilde{x}_0 \\
 & p_0 = r_0; & \tilde{p}_0 = \tilde{r}_0 \\
 & \rho_0 = \tilde{r}_0^T r_0 & \\
For & n = 0, 1, \ldots & \\
 & \sigma_n = \tilde{p}_n^T A p_n & \\
 & \alpha_n = \rho_n / \sigma_n & \\
 & r_{n+1} = r_n - \alpha_n A p_n; & \tilde{r}_{n+1} = \tilde{r}_n - \alpha_n A^T \tilde{p}_n \\
 & x_{n+1} = x_n + \alpha_n p_n; & \tilde{x}_{n+1} = \tilde{x}_n + \alpha_n \tilde{p}_n \\
 & \rho_{n+1} = \tilde{r}_{n+1}^T r_{n+1} & \\
 & \beta_{n+1} = \frac{\rho_{n+1}}{\rho_n} & \\
 & p_{n+1} = r_{n+1} + \beta_{n+1} p_n; & \tilde{p}_{n+1} = \tilde{r}_{n+1} + \beta_{n+1}\tilde{p}_n \\
End & &
\end{array}$$

We can see that there are two possible kinds of breakdowns in the above routine:

1. $\rho_n = 0$, but $r_n \neq 0$ (Lanczos breakdown)

2. $\sigma_n = 0$ (pivot breakdown)

The first kind of breakdown can be handled using look-ahead techniques, but this can be quite involved and must be done in a careful and sophisticated way, as in [3, 8, 9, 16]. The second kind of breakdown can be cured more easily using the composite step idea mentioned earlier. Since it is this composite step technique we will be using for CSCGS, we will describe in more detail how it is implemented in the CSBCG algorithm.

Suppose, then, in running the BCG algorithm, we encounter a situation where $\sigma_n = 0$ at step n. We see that the values $x_{n+1}, \tilde{x}_{n+1}, r_{n+1}, \tilde{r}_{n+1}$ are not defined. CSBCG overcomes this problem by skipping the $n+1$ update and computing the quantities in step $n+2$ by using scaled versions of r_{n+1} and $\tilde{r}_{n+1}$, which do not require divisions by σ_n. More specifically, auxiliary vectors $z_{n+1} \in K_{n+2}(r_0)$ and $\tilde{z}_{n+1} \in K^*_{n+2}(\tilde{r}_0)$ are defined:

$$z_{n+1} = \sigma_n r_{n+1} \tag{2}$$
$$= \sigma_n r_n - \rho_n A p_n, \tag{3}$$

$$\tilde{z}_{n+1} = \sigma_n \tilde{r}_{n+1}$$
$$= \sigma_n \tilde{r}_n - \rho_n A^T \tilde{p}_n.$$

These are then used in looking for the step $n+2$ iterate:

$$x_{n+2} = x_n + [p_n, z_{n+1}] f_n,$$
$$\tilde{x}_{n+2} = \tilde{x}_n + [\tilde{p}_n, \tilde{z}_{n+1}] \tilde{f}_n,$$

where $f_n, \tilde{f}_n \in R^2$.
The corresponding residuals are :

$$r_{n+2} = r_n - A[p_n, z_{n+1}] f_n, \tag{4}$$
$$\tilde{r}_{n+2} = \tilde{r}_n - A^T [\tilde{p}_n, \tilde{z}_{n+1}] \tilde{f}_n. \tag{5}$$

Similarly, the search directions in a composite step are found using

$$p_{n+2} = r_{n+2} + [p_n, z_{n+1}] g_n, \tag{6}$$
$$\tilde{p}_{n+2} = \tilde{r}_{n+2} + [\tilde{p}_n, \tilde{z}_{n+1}] \tilde{g}_n, \tag{7}$$

where $g_n, \tilde{g}_n \in R^2$.

To solve for the unknowns $f_n = \left(f_n^{(1)}, f_n^{(2)}\right)^T$ and $g_n = \left(g_n^{(1)}, g_n^{(2)}\right)^T$, we impose the Galerkin condition of BCG:

$$(\tilde{p}_n, \tilde{z}_{n+1})^T r_{n+2} = 0,$$

and the conjugacy condition:

$$(\tilde{p}_n, \tilde{z}_{n+1})^T A p_{n+2} = 0,$$

and solve the resulting two 2×2 linear systems:

$$\begin{bmatrix} \tilde{p}_n^T A p_n & \tilde{p}_n^T A z_{n+1} \\ \tilde{z}_{n+1}^T A p_n & \tilde{z}_{n+1}^T A z_{n+1} \end{bmatrix} \begin{bmatrix} f_n^{(1)} \\ f_n^{(2)} \end{bmatrix} = \begin{bmatrix} \tilde{p}_n^T r_n \\ \tilde{z}_{n+1}^T r_n \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} \tilde{p}_n^T A p_n & \tilde{p}_n^T A z_{n+1} \\ \tilde{z}_{n+1}^T A p_n & \tilde{z}_{n+1}^T A z_{n+1} \end{bmatrix} \begin{bmatrix} g_n^{(1)} \\ g_n^{(2)} \end{bmatrix} = - \begin{bmatrix} \tilde{p}_n^T A r_{n+2} \\ \tilde{z}_{n+1}^T A r_{n+2} \end{bmatrix}. \tag{9}$$

A similar process solves $\tilde{f}_n$ and $\tilde{g}_n$. It is now possible to compute x_{n+2}, $\tilde{x}_{n+2}$, r_{n+2}, $\tilde{r}_{n+2}$ and thus, advance from step n to step $n+2$. The Composite Step BCG algorithm, then, is simply the combination of the 1×1 and 2×2 steps. It can be proven [1, 2], provided the underlying Lanczos process doesn't break down, that the use of 2-steps are sufficient for CSBCG to compute exactly those iterates of BCG which are well defined.

Ideas similar to the Composite Step approach were used earlier by Luenberger [23] and Fletcher [13] in the case where A is symmetric. Note that in this case, there are no Lanczos breakdowns and mathematically, CSBCG and the methods in [14, 23] all produce precisely the same iterate x_{n+2}. However, the details of exactly how x_{n+2} is updated are different. Thus, CSBCG can be viewed as a generalization of [14, 23] to nonsymmetric A.

3. Composite Step CGS. The Conjugate Gradients Squared method is an attractive alternate to BCG for reasons mentioned earlier. However, breakdowns exist in the CGS algorithm analogous to the breakdowns encountered in BCG. The purpose of this paper is to cure the pivot breakdowns using a composite step technique similar to CSBCG (Section 2). We will assume for the rest of this paper that Lanczos breakdowns do not occur.

Suppose CSBCG were to take only 1×1 steps, then we could square CSBCG in the same way CGS squares BCG (see [27]). First, we must express the CSBCG quantities in polynomial form. Since the residual $r_n = b - Ax_n$ of any iterate $x_n \in x_0 + K_n(r_0)$ can be written in the form $r_n = \phi_n(A)r_0$, where ϕ_n is a polynomial of degree $\leq n$ and $\phi_n(0) = 1$, we can write the CSBCG residual as

$$r_n = \phi_n(A)r_0.$$

Similarly,

$$\begin{aligned} p_n &= \psi_n(A)r_0, \\ z_{n+1} &= \xi_{n+1}(A)r_0. \end{aligned}$$

We can now obtain squared CSCGS polynomials

$$r_n^{CSCGS} = \phi_n^2(A)r_0$$

$$p_n^{CSCGS} = \psi_n^2(A)r_0$$

with corresponding iterates of the form $x_n^{CSCGS} \in x_0 + K_{2n}(r_0)$.

To handle breakdowns, as in CSBCG, if we encounter $\sigma_n = 0$ in the n^{th} step of the CGS algorithm, we will need to take a 2×2 step. We formulate this first by writing the polynomial equivalents of equations (4), (6), and (3):

$$\phi_{n+2}(\vartheta) = \phi_n - \vartheta\psi_n f_n^{(1)} - \vartheta\xi_{n+1} f_n^{(2)}, \tag{10}$$

$$\psi_{n+2}(\vartheta) = \phi_{n+2} + \psi_n g_n^{(1)} + \xi_{n+1} g_n^{(2)}, \tag{11}$$

$$\xi_{n+1}(\vartheta) = \sigma_n\phi_n - \rho_n\vartheta\psi_n. \tag{12}$$

Squaring equations (10) and (11), we obtain:

$$\phi_{n+2}^2 = \phi_n^2 + \left[f_n^{(1)}\right]^2 \vartheta^2\psi_n^2 + \left[f_n^{(2)}\right]^2 \vartheta^2\xi_{n+1}^2$$

$$
\begin{aligned}
& -2f_n^{(1)}\vartheta\phi_n\psi_n - 2f_n^{(2)}\vartheta\phi_n\xi_{n+1} + 2f_n^{(1)}f_n^{(2)}\vartheta^2\psi_n\xi_{n+1}, \\
\psi_{n+2}^2 &= \phi_{n+2}^2 + \left[g_n^{(1)}\right]^2\psi_n^2 + \left[g_n^{(2)}\right]^2\xi_{n+1}^2 \\
& +2g_n^{(1)}\phi_{n+2}\psi_n + 2g_n^{(2)}\phi_{n+2}\xi_{n+1} + 2g_n^{(1)}g_n^{(2)}\psi_n\xi_{n+1}.
\end{aligned}
$$

Thus, we see that we will need relations to define the quantities $\phi_n\xi_{n+1}$, $\psi_n\xi_{n+1}$, ξ_{n+1}^2, $\phi_{n+2}\psi_n$, $\phi_{n+2}\xi_{n+1}$. From equations (12) and (10), we can write:

$$
\begin{aligned}
\phi_n\xi_{n+1} &= \sigma_n\phi_n^2 - \rho_n\vartheta\phi_n\psi_n \\
\psi_n\xi_{n+1} &= \sigma_n\phi_n\psi_n - \rho_n\vartheta\psi_n^2 \\
\xi_{n+1}^2 &= \sigma_n^2\phi_n^2 - 2\sigma_n\rho_n\vartheta\phi_n\psi_n + \rho_n^2\vartheta^2\psi_n^2 \\
\phi_{n+2}\psi_n &= \phi_n\psi_n - f_n^{(1)}\vartheta\psi_n^2 - f_n^{(2)}\vartheta\psi_n\xi_{n+1} \\
\phi_{n+2}\xi_{n+1} &= \phi_n\xi_{n+1} - f_n^{(1)}\vartheta\psi_n\xi_{n+1} - f_n^{(2)}\vartheta\xi_{n+1}^2.
\end{aligned}
$$

All of these quantities can be defined in terms of ϕ_n^2 $(= r_n^{CSCGS})$, ψ_n^2 $(= p_n^{CSCGS})$, and $\phi_n\psi_n$. What remains, then, is the updating of $\phi_n\psi_n$. If the previous step was a 1×1 step, $\phi_n\psi_n$ is defined as

$$\phi_n\psi_n = \phi_n^2 + \beta_n\phi_n\psi_{n-1},$$

where the term $\phi_n\psi_{n-1}$ can be updated (as in CGS) by the relation

$$\phi_n\psi_{n-1} = \phi_{n-1}\psi_{n-1} - \alpha_{n-1}\vartheta\psi_{n-1}^2.$$

However, if the previous step was a 2×2 step, we cannot use this because the mixed term $\phi_n\psi_{n-1}$ doesn't exist. Instead, we use the fact that

$$\psi_n = \phi_n + \psi_{n-2}g_{n-2}^{(1)} + \xi_{n-1}g_{n-2}^{(2)}.$$

Hence,

$$\phi_n\psi_n = \phi_n^2 + g_{n-2}^{(1)}\phi_n\psi_{n-2} + g_{n-2}^{(2)}\phi_n\xi_{n-1},$$

where the terms $\phi_n\psi_{n-2}$ and $\phi_n\xi_{n-1}$ can be updated by the relations $\phi_{n+2}\psi_n$ and $\phi_{n+2}\xi_{n+1}$ which have already been computed.

4. Implementation Details. One simplification we can perform is in the process of finding coefficients f_n and g_n. As shown in [1] (Lemma 5.1), the systems (8) and (9) can be simplified and the two systems can be solved explicitly for f_n and g_n:

$$f_n = (\zeta_{n+1}\rho_n, \theta_{n+1})\rho_n^2/\delta_n \tag{13}$$

$$g_n = (\rho_{n+2}/\rho_n, \sigma_n\rho_{n+2}/\theta_{n+1}), \tag{14}$$

where ρ_n and σ_n are defined as before, and

$$
\begin{aligned}
\zeta_{n+1} &= \tilde{z}_{n+1}^T A z_{n+1}, \\
&= (\tilde{r}_0, A\xi_{n+1}^2(A)r_0), \\
\theta_{n+1} &= \tilde{z}_{n+1}^T z_{n+1}, \\
&= (\tilde{r}_0, \xi_{n+1}^2(A)r_0), \\
\delta_n &= \sigma_n\zeta_{n+1}\rho_n^2 - \theta_{n+1}^2.
\end{aligned}
$$

Fortunately, all of these quantities are already computed.

As far as deciding when to take a 2×2 step, we need a practical stepping strategy that will skip over exact breakdowns. We use the same criterion as in CSBCG [1, 2]:

$$\|r_{n+1}\| > \max\{\|r_n\|, \|r_{n+2}\|\}. \tag{15}$$

If this condition were met and we performed two 1×1 steps, it would result in a "peak" in the residual convergence. By taking a 2×2 step, we skip over this and obtain a smoother, more stable method. In order to avoid unnecessary computation of $\|r_{n+2}\|$, we express condition (15) in the following algorithm:

```
If (‖r_{n+1}‖ < ‖r_n‖) then
   choose 1 × 1 step
else
   if (‖r_{n+1}‖ < ‖r_{n+2}‖) then
      choose 1 × 1 step
   else
      choose 2 × 2 step
   end
end
```

In order to implement the above test efficiently, and stably, we need to arrange the associated computations in a careful way. Our approach follows that in [1, 2], but the details are somewhat different. Since we do not have r_{n+1} yet, and do not want to compute it unnecessarily (if we decide to skip over it), we use the fact that that $\xi_{n+1} = \sigma_n \phi_{n+1}$ to obtain the relationship $s_{n+1} = \sigma_n^2 r_{n+1}$. Hence, the first condition, $\|r_{n+1}\| < \|r_n\|$, can be written:

$$\|s_{n+1}\| < \sigma_n^2 \|r_n\|.$$

To estimate $\|r_{n+2}\|$, we use the fact that we can write f_n explicitly as in (13) and rescale the r_{n+2} update by defining ν_{n+2} as follows:

$$\nu_{n+2} \equiv \delta_n^2 r_{n+2} = \delta_n^2 r_n - A[\alpha'_n(\delta_n u_n + v'_n) + \alpha'_{n+1}(\delta_n t_{n+1} + w'_{n+1})], \tag{16}$$

where $\alpha'_n = \zeta_{n+1}\rho_n^3, \alpha'_{n+1} = \theta_{n+1}\rho_n^2$, and v'_n, w'_{n+1} are scaled versions of v_n, w_{n+1} :

$$v'_n \equiv \delta_n v_n = \delta_n u_n - \alpha'_n b_n - \alpha'_{n+1} c_{n+1},$$

$$w'_{n+1} \equiv \delta_n w_{n+1} = \delta_n t_{n+1} - \alpha'_n c_{n+1} - \alpha'_{n+1} d_{n+1}.$$

Thus, the condition $\|r_{n+1}\| < \|r_{n+2}\|$ can be expressed as:

$$\delta_n^2 \|s_{n+1}\| < \sigma_n^2 \|\nu_{n+2}\|. \tag{17}$$

Unfortunately, evaluating the exact $\|\nu_{n+2}\|$ using (16) requires one more matrix-vector multiply. Hence, for practical purposes, we use the following upper bound to approximate $\|\nu_{n+2}\|$:

$$\|\nu_{n+2}\| \leq \|\hat{\delta}_n^2 r_n\| + \kappa \|\hat{\alpha}_n(\hat{\delta}_n u_n + \hat{v}_n) + \alpha'_{n+1}(\hat{\delta}_n t_{n+1} + \hat{w}_n)\|, \tag{18}$$

where $\kappa \approx \|A\|$ (e.g. from power iteration), and where the quantities $\hat{\delta}_n, \hat{\alpha}_n, \hat{v}_n, \hat{w}_n$ are values based on the estimate $|\zeta_{n+1}| = |\tilde{r}_0^T A s_{n+1}| \approx \kappa \|\tilde{r}_0\| \|s_{n+1}\|$.

This approximate test may be misleading in the rare case where the 2×2 matrix in equations (8) and (9) is near singular, but is not revealed by the approximation (18). This will result in a 2×2 step where a 1×1 step is desired. Hence, we include an additional check to monitor this case by computing the exact value of ζ_{n+1} which requires an additional matrix-vector multiplication. In doing this, a matrix vector multiplication is wasted only in the rare cases where a 2×2 step has to be aborted.

It is important to note, then, that even with the approximation, our stepping strategy still avoids exact breakdown. Moreover, we want to emphasize that this stepping strategy not only skips exact breakdowns, it is also designed to yield smoother residuals. It may use more composite steps than necessary to overcome exact breakdowns, but it has the advantage of not involving any user specified tolerance parameters.

In Table 1, we present the CSCGS algorithm. Note that the variable r_n is defined as the CSCGS residual $r_n = \phi_n^2(A) r_0$ and $p_n = \psi_n^2(A) r_0$. Note also that if only 1×1 steps are taken, we have exactly the CGS algorithm. For 2×2 steps, we use some quantities from CGS, but also introduce some new quantities such as $t_{n+1} = \phi_n \xi_{n+1}$.

Note that there are 5 matrix-vector multiplications required for a 2×2 step, whereas in two steps of CGS, only 4 are needed. This is the price we pay for the composite step. However, it is still a considerable savings from BCG and also significantly less work than QMRS [17] and TFiQMR [10], where an extra matrix-vector multiply is required at *every* step.

5. Residual Smoothing. One other technique we can incorporate to obtain smoother convergence is the Minimal Residual Smoothing (MRS) algorithm, due to Schönauer and Weiss [26, 30] and further extended by Zhou and Walker [31] and Brezinski and Redivo-Zaglia [6]. The idea is to generate a sequence $\{y_n\}$ using the relation $y_n = (1 - \eta_n) y_{n-1} + \eta_n x_n$, where x_n is the iterate of a particular iterative method and η_n is chosen so that the new resulting residuals $b - Ay_n$ have monotone decreasing norms and $\|b - Ay_n\|_2 \leq \|b - Ax_n\|_2$, for each n. Note that although doing this will not cure the problem of breakdown, it does track the true residual more accurately, and in a smooth fashion, as shown in [31].

To incorporate Minimal Residual Smoothing into the Composite Step algorithm, use $\{x_n\}$ from CSCGS. Choosing this set of $\{x_n\}$ will provide a more stable basis for smoothing than would, say, CGS. Then, we generate y_n as described above for 1×1 steps and for 2×2 steps, define $y_n = (1 - \eta_n) y_{n-2} + \eta_n x_n$ and choose η_n to minimize $\|b - Ay_n\|_2$.

The MRS algorithm we have implemented (shown in Table 2), is one of several smoothing techniques described in [31]. Although we have selected this particular algorithm, note that the others perform similarly.

6. Numerical Experiments. All experiments are run in MATLAB 4.0 on a SUN Sparc station with machine precision about 10^{-16}. We shall use CSCGS* to refer to an implementation of CSCGS where the norm estimate is *not* used in the composite step decision strategy (i.e., *all* matrix-vector multiplications are performed).

6.1. Example 1. We begin the numerical experiments with a contrived example to illustrate the superior numerical stability of composite step methods over those

TABLE 1
Algorithm CSCGS

$\rho_0 = \tilde{r}_0^T r_0$; $p_0 = u_0 = r_0$; $b_0 = e_0 = Ap_0$
Compute κ = estimate for $\|A\|$
$n \leftarrow 0$
While *method not converged yet* do:
$\sigma_n = \tilde{r}_0^T b_n$
$q_{n+1} = \sigma_n u_n - \rho_n b_n$; $c_{n+1} = Aq_{n+1}$
$s_{n+1} = \sigma_n^2 r_n - \rho_n \sigma_n e_n - \rho_n c_{n+1}$; $\xi_{n+1} = \|s_{n+1}\|$; $\phi_n = \|r_n\|$
% Decide whether to take a 1×1 step or a 2×2 step.
If $\xi_{n+1} < \sigma_n^2 \phi_n$, Then % $\|r_{n+1}\| < \|r_n\|$
one-step $= 1$
Else
$\theta_{n+1} = \tilde{r}_0^T s_{n+1}$; $\hat{\zeta}_n = \kappa \phi_0 \xi_{n+1}$; $\hat{\delta}_n = \sigma_n \hat{\zeta}_n \rho_n^2 - \theta_{n+1}^2$
$\hat{\alpha}_n = \hat{\zeta}_n \rho_n^3$; $\hat{\alpha}_{n+1} = \theta_{n+1} \rho_n^2$
$t_{n+1} = \sigma_n r_n - \rho_n e_n$
$\hat{v}_n = \hat{\delta}_n u_n - \hat{\alpha}_n b_n - \hat{\alpha}_{n+1} c_{n+1}$; $\hat{w}_n = \hat{\delta}_n t_{n+1} - \hat{\alpha}_n c_{n+1} - \hat{\alpha}_{n+1} \kappa s_{n+1}$
$\hat{\nu}_{n+2} = \|\hat{\delta}_n^2 r_n\| + \kappa \|\hat{\alpha}_n (\hat{\delta}_n u_n + \hat{v}_n) + \alpha_{n+1} (\hat{\delta}_n t_{n+1} + \hat{w}_n)\|$
If $\hat{\delta}_n^2 \xi_{n+1} < \sigma_n^2 \hat{\nu}_{n+2}$, Then % $\|r_{n+1}\| < \|r_{n+2}\|$
one-step $= 1$
Else
$d_{n+1} = As_{n+1}$; $\zeta_{n+1} = \tilde{r}_0^T d_{n+1}$; $\delta_n = \sigma_n \zeta_{n+1} \rho_n^2 - \theta_{n+1}^2$
If $\delta_n^2 \xi_{n+1} < \sigma_n^2 \hat{\nu}_{n+2}$, Then % Test again with true δ_n
one-step $= 1$
Else
one-step $= 0$
End If
End If
End If
% Compute next iterate.
If *one-step*, Then % Usual CGS
$\alpha_n = \rho_n / \sigma_n$
$r_{n+1} = r_n - \alpha_n (e_n + c_{n+1}/\sigma_n)$
$x_{n+1} = x_n + \alpha_n (u_n + q_{n+1}/\sigma_n)$
$\rho_{n+1} = \tilde{r}_0^T r_{n+1}$; $\beta_n = \rho_{n+1}/\rho_n$
$u_{n+1} = r_{n+1} + \beta_n q_{n+1}/\sigma_n$; $e_{n+1} = Au_{n+1}$
$p_{n+1} = u_{n+1} + \beta_n (q_{n+1}/\sigma_n + \beta_n p_n)$
$b_{n+1} = e_{n+1} + \beta_n (c_{n+1}/\sigma_n + \beta_n b_n)$
$n \leftarrow n + 1$
Else % 2×2 step CSCGS
$\alpha_n = \zeta_{n+1} \rho_n^3 / \delta_n$; $\alpha_{n+1} = \theta_{n+1} \rho_n^2 / \delta_n$
$v_n = u_n - \alpha_n b_n - \alpha_{n+1} c_{n+1}$; $w_{n+1} = t_{n+1} - \alpha_n c_{n+1} - \alpha_{n+1} d_{n+1}$
$r_{n+2} = r_n - A[\alpha_n (u_n + v_n) + \alpha_{n+1} (t_{n+1} + w_{n+1})]$
$x_{n+2} = x_n + [\alpha_n (u_n + v_n) + \alpha_{n+1} (t_{n+1} + w_{n+1})]$
$\rho_{n+2} = \tilde{r}_0^T r_{n+2}$; $\beta_n = \rho_{n+2}/\rho_n$; $\beta_{n+1} = \sigma_n \rho_{n+2}/\theta_{n+1}$
$u_{n+2} = r_{n+2} + \beta_n v_n + \beta_{n+1} w_{n+1}$; $e_{n+2} = Au_{n+2}$
$p_{n+2} = u_{n+2} + \beta_n (v_n + \beta_n p_n + \beta_{n+1} q_{n+1}) + \beta_{n+1} (w_{n+1} + \beta_n q_{n+1} + \beta_{n+1} s_{n+1})$
$b_{n+2} = Ap_{n+2}$
$n \leftarrow n + 2$
End If
End While

TABLE 2
Algorithm MRS for CSCGS

Set $s_0 = r_0^{CSCGS}$, $y_0 = \hat{x}_0 = x_0^{CSCGS}$, and $u_0 = v_0 = 0$
Let $\hat{x}_0, \hat{x}_1, \hat{x}_2, \ldots$ be consecutive iterates of the CSCGS method (using either 1×1 steps or 2×2 steps).
For $n = 1, 2, 3, \ldots$

$$
\begin{aligned}
p_n &= \hat{x}_n - \hat{x}_{n-1}; \\
u_n &= u_{n-1} + Ap_n; \\
v_n &= v_{n-1} + p_n; \\
\eta_n &= s_{n-1}^T u_n / u_n^T u_n; \\
s_n &= s_{n-1} - \eta_n u_n; \\
y_n &= y_{n-1} - \eta_n v_n; \\
u_n &\leftarrow (1 - \eta_n) u_n; \\
v_n &\leftarrow (1 - \eta_n) v_n;
\end{aligned}
$$

without composite step. Let A be a modification of an example found in [24]:

$$A = \begin{pmatrix} \epsilon & 1 \\ -1 & \epsilon \end{pmatrix} \otimes I_{N/2},$$

i.e., A is a $N \times N$ block diagonal with 2×2 blocks, and $N = 40$. By choosing $b = (1 \quad 0 \quad 1 \quad 0 \quad \cdots)^T$ and a zero initial guess, we set $\sigma_0 = \epsilon$, and thus, we can forsee numerical problems with methods such as BCG and CGS when ϵ is small. Although these methods converge in 2 steps in exact arithmetic when $\epsilon \neq 0$, in finite precision, convergence gets increasingly unstable as ϵ decreases. Table 3 shows the relative error in the solution after 2 steps of BCG, CGS, and their composite step counterparts. Note that the loss of significant digits in BCG and CGS is proportional to $O(\epsilon^{-1})$ and $O(\epsilon^{-2})$, respectively, whereas the accuracy of CSBCG and CSCGS is insensitive to ϵ.

TABLE 3
Example 1

Rel. error in the soln. after 2 steps ($N = 40$)				
	BCG	CSBCG	CGS	CSCGS
$\epsilon = 10^{-4}$	1.5×10^{-12}	1.1×10^{-16}	2.5×10^{-8}	0
$\epsilon = 10^{-8}$	2.5×10^{-8}	1.1×10^{-16}	1.0×10^{0}	1.1×10^{-16}
$\epsilon = 10^{-12}$	4.9×10^{-4}	2.0×10^{-28}	1.3×10^{8}	2.0×10^{-28}

6.2. Example 2. This example comes from the Harwell-Boeing set of sparse test matrices [12]. The matrix is a discretization of the convection-diffusion equation:

$$L(u) = -\Delta u + 100(xu_x + yu_y) - 100u$$

on the unit square for a 63×63 grid. We use a random right hand side, zero initial guess, and left diagonal preconditioning. Figure 1 plots the number of iterations versus

the true residual norm for CSCGS* and CGS, and smoothed versions of these: MRS on CSCGS* and TFQMR [14]. This is done in order to illustrate the cutting off of the "peaks" of CGS, as seen in this figure. Specifically, note that the maximum point of the CGS curve is around 10^{10} whereas it is only 10^6 for the Composite Step version. For this particular example, the CGS residual stagnates at 10^{-6} and so does its smoothed counterpart TFQMR, whereas CSCGS* reaches the stopping criterion $\|r_n\|/\|r_0\| < 10^{-8}$.

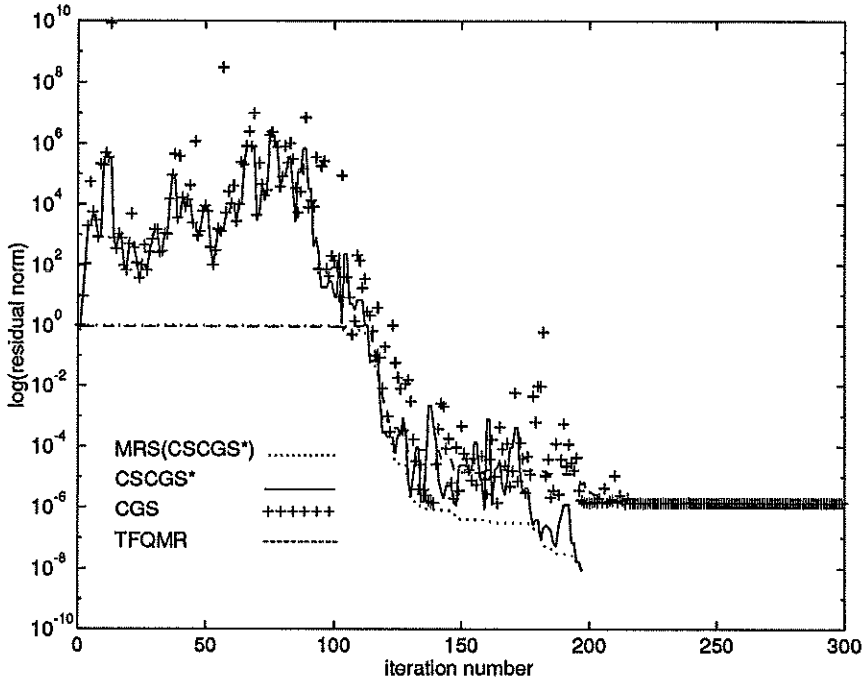

FIG. 1. *Example 2a*

The next plot, Fig. 2, shows the same example with different axes: the number of matrix-vector multiplications versus the norm of the residual. Instead of CSCGS*, we now illustrate the behavior of CSCGS - using the norm estimation strategy described in section 4 . Recall that for CSCGS, an extra matrix-vector multiply is required when a 2×2 step is performed. Note, however, that this does not make a significant impact when compared to the previous figure. We see that even with the approximation, the composite step method manages to control the wild behavior of CGS and eventually converge to the desired residual tolerance. In this example, the total number of 2×2 steps taken is 23, whereas 133 1×1 steps are taken. The 2×2 step was aborted 2 times.

Hence, Figures 1 and 2 clearly indicate the advantage of CSCGS and MRS(CSCGS) over CGS for this example matrix.

6.3. Example 3. The next example, taken from [11], is a discretization of

$$L(u) = -\Delta u + 2e^{2(x^2+y^2)}u_x - 100u,$$

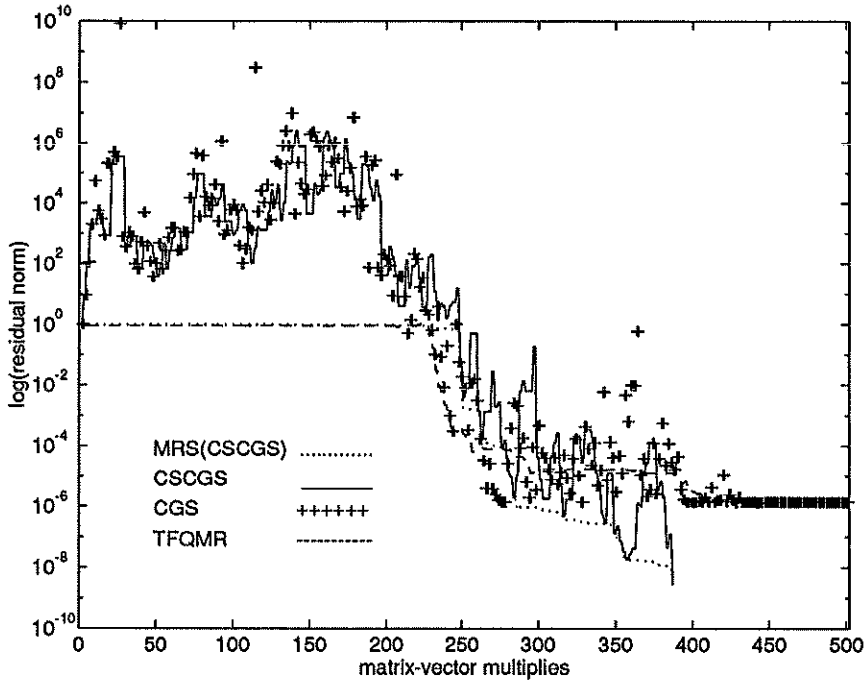

FIG. 2. *Example 2b*

on the unit square for a 40×40 grid. We use a random right hand side, and the same preconditioning and stopping criterion as in the previous example. Figure 3 illustrates the superior behavior of MRS(CSCGS) as compared with not only CGS, but also Bi-CGSTAB [28], another transpose-free product method, as well as CSBCG. We did not include the CSCGS plot, so as not to over-complicate the picture, but mention that it converges in about the same number of matrix-vector multiplications as MRS(CSCGS). We see that the new method reaches the desired tolerance with less work than the other methods. There were 239 1×1 steps and 37 2×2 steps.

Acknowledgement: We would like to thank Claude Brezinski for his helpful comments and suggestions.

REFERENCES

[1] R. E. BANK AND T. F. CHAN, *A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, UCLA CAM Tech. Report 93-21 (1993). Numerical Algorithms, this issue.

[2] R. E. BANK AND T. F. CHAN, *An analysis of the composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, UCLA CAM Tech. Report 92-53(1992). Numer. Math., to appear.

[3] C. BREZINSKI AND H. SADOK, *Lanczos-type algorithms for solving systems of linear equations*, Applied Numerical Mathematics, 11(1993), pp. 443-473.

[4] C. BREZINSKI AND H. SADOK, *Avoiding breakdown in the CGS algorithm*, Numer. Alg. 1(1991) 199-206.

[5] C. BREZINSKI AND M. REDIVO-ZAGLIA, *Breakdowns in the computation of orthogonal polynomials*, Nonlinear Numerical Methods and Rational Approximation, A. Cuyt ed., Kluwer,

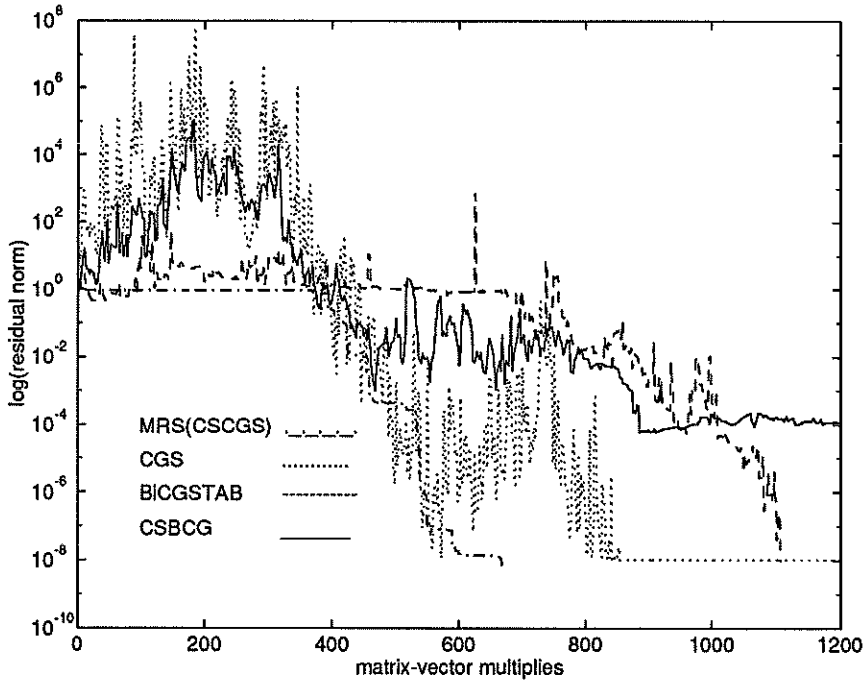

FIG. 3. *Example 3*

Dordrecht. To appear.

[6] C. BREZINSKI AND M. REDIVO-ZAGLIA, *Hybrid procedures for solving linear systems*, Numer. Math. in press. (1993).

[7] C. BREZINSKI AND M. REDIVO-ZAGLIA, *Treatment of near-breakdown in the CGS algorithm*, Numer. Alg., this issue.

[8] C. BREZINSKI AND M. REDIVO-ZAGLIA AND H. SADOK, *A breakdown-free Lanczos type algorithm for solving linear systems*, Numer. Math. 63, 29-38 (1992).

[9] C. BREZINSKI AND M. REDIVO-ZAGLIA AND H. SADOK, *Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Alg. 1(1991) 261-284.

[10] T. F. CHAN, L. DEPILLIS, H. VAN DER VORST, *A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems*, UCLA CAM Tech. Report 91-17 (1991).

[11] T. F. CHAN, E. GALLOPOULOS, V. SIMONCINI, T. SZETO, C. TONG, *QMRCGSTAB: A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems*, UCLA CAM Tech. Report 92-26 (1992). To appear in SIAM J. Sci. Stat. Comput.

[12] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Softw., 15(1989), pp. 1-14.

[13] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, Lecture Notes on Math. 506, G.A. Watson, ed., Springer-Verlag, Berlin, 1976, pp. 73-89.

[14] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Stat. Comput., 14(1993), pp. 470-482.

[15] R. W. FREUND AND N. M. NACHTIGAL, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numerische Mathematik 60(1991), pp. 315-339.

[16] R. W. FREUND AND M.H. GUTKNECHT AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comp., 13 (1992) 137-158.

[17] R. W. FREUND AND T. SZETO, *A Quasi-minimal residual squared algorithm for non-Hermitian linear systems*, UCLA CAM Tech. Report 92-19 (1992). Presented at the Copper Mountain Conference on Iterative Methods, April 1992.

[18] M. H. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algo-*

rithms, Part I, SIAM J. Matrix Anal. Appl. 13(1992), pp. 594-639.

[19] M. H. GUTKNECHT, *The unsymmetric Lanczos algorithms and their relations to Pade approximation, continued fraction and the QD algorithm*, in Proc. of the Copper Mt. Conf. on Iterative Methods, 1990.

[20] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49(1952), pp. 409-436.

[21] W. JOUBERT, *Generalized conjugate gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations*, Ph.D. Thesis, The University of Texas at Austin, Austin, TX (1990).

[22] C. LANCZOS, *Solution of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand. 49 (1952), pp. 33-53.

[23] D. G. LUENBERGER, *Hyperbolic Pairs in the Method of Conjugate Gradients*, SIAM J. Appl. Math. 17(1969), pp.1263-1267.

[24] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, Tech. Report, MIT Dept. of Math., Cambridge, MA (1990).

[25] B. N. PARLETT, D. R. TAYLOR, AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44(1985), pp.105-124.

[26] W. SCHÖNAUER, *Scientific computing on vector computers*, North-Holland, Amsterdam, New York, Oxford, Tokyo (1987).

[27] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10(Jan 1989), pp. 36-52.

[28] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13(March 1992), pp. 631-644.

[29] H. A. VAN DER VORST, *The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L. Yu. Kolotilina (eds.), Lecture Notes in Mathematics 1457, Springer Verlag, Berlin, 1990.

[30] R. WEISS, *Convergence behavior of generalized conjugate gradient methods*, Ph. D. thesis, University of Karlsruhe (1990).

[31] L. ZHOU AND H. F. WALKER, *Residual smoothing techniques for iterative methods*, Tech. rep., Dept. of Mathematics and Statistics, Utah State University (1992).