NESTED KRYLOV METHODS AND PRESERVING THE ORTHOGONALITY

Eric De Sturler [1]
Delft University of Technology
Delft, The Netherlands

Diederik R. Fokkema [2]
University of Utrecht
Utrecht, The Netherlands

SUMMARY

Recently the GMRESR inner-outer iteration scheme for the solution of linear systems of equations has been proposed by Van der Vorst and Vuik. Similar methods have been proposed by Axelsson and Vassilevski [1] and Saad (FGMRES) [10]. The outer iteration is GCR, which minimizes the residual over a given set of direction vectors. The inner iteration is GMRES, which at each step computes a new direction vector by approximately solving the residual equation. However, the optimality of the approximation over the space of outer search directions is ignored in the inner GMRES iteration. This leads to suboptimal corrections to the solution in the outer iteration, as components of the outer iteration directions may reenter in the inner iteration process. Therefore we propose to preserve the orthogonality relations of GCR in the inner GMRES iteration. This gives optimal corrections; however, it involves working with a singular, non-symmetric operator. We will discuss some important properties and we will show by experiments that, in terms of matrix vector products, this modification (almost) always leads to better convergence. However, because we do more orthogonalizations, it does not always give an improved performance in CPU-time. Furthermore, we will discuss efficient implementations as well as the truncation possibilities of the outer GCR process. The experimental results indicate that for such methods it is advantageous to preserve the orthogonality in the inner iteration. Of course we can also use other iteration schemes than GMRES as the inner method. Especially methods with short recurrences like BICGSTAB seem of interest.

INTRODUCTION

For the solution of systems of linear equations the so-called Krylov subspace methods are very popular. However, for general matrices no Krylov method can satisfy a global optimality requirement and have short recurrences [5]. Therefore either restarted or truncated versions of optimal methods, like GMRES [11], are used or methods with short recurrences, which do not satisfy a global optimality requirement, like BiCG [6], BICGSTAB [14], BICGSTAB(l) [12], CGS [13] or QMR [8]. Recently Van der Vorst and Vuik proposed a class of methods, GMRESR [15], which are nested GMRES methods; see Fig. 2. The GMRESR algorithm is based upon the GCR algorithm [4]; see Fig. 1. For a given initial guess x_0, they both compute approximate solutions x_k, such that $x_k - x_0 \in span\{u_1, u_2, \ldots, u_k\}$ and $\|r_k\|_2 = \|b - Ax_k\|_2$ is minimal.

[1]Delft University of Technology, Faculty of Technical Mathematics and Informatics, P.O. Box 5031, NL-2600 GA Delft, The Netherlands, E-mail: witaeds@dutinfh.tudelft.nl. The author wishes to acknowledge Shell Research B.V. and STIPT for the financial support of his research.

[2]Mathematical Institute, University of Utrecht, P.O. Box 80.010. NL-3508 TA Utrecht, The Netherlands, E-mail: fokkema@math.ruu.nl. This work was supported in part by a NCF/Cray Research University Grant CRG 92.03

GCR:

1. Select x_0, m, tol;
   $r_0 = b - Ax_0$, $k = 0$;
2. **while** $\|r_k\|_2 > tol$ **do**
   $k = k + 1$;
   $u_k = r_{k-1}$; $c_k = Au_k$;
   for $i = 1, k-1$ **do**;
   $\alpha_i = c_i^T c_k$;
   $c_k = c_k - \alpha_i c_i$;
   $u_k = u_k - \alpha_i u_i$;
   $c_k = c_k / \|c_k\|$;
   $u_k = u_k / \|c_k\|$;
   $x_k = x_{k-1} + (c_k^T r_{k-1}) u_k$;
   $r_k = r_{k-1} - (c_k^T r_{k-1}) c_k$;

Figure 1: The GCR algorithm

GMRESR:

1. Select x_0, m, tol;
   $r_0 = b - Ax_0$, $k = 0$;
2. **while** $\|r_k\|_2 > tol$ **do**
   $k = k + 1$;
   $u_k = \mathcal{P}_{m,k}(A) r_{k-1}$; $c_k = Au_k$;
   for $i = 1, \ldots, k-1$ **do**
   $\alpha_i = c_i^T c_k$;
   $c_k = c_k - \alpha_i c_i$;
   $u_k = u_k - \alpha_i u_i$;
   $c_k = c_k / \|c_k\|_2$;
   $u_k = u_k / \|c_k\|_2$;
   $x_k = x_{k-1} + (c_k^T r_{k-1}) u_k$;
   $r_k = r_{k-1} - (c_k^T r_{k-1}) c_k$;

$\mathcal{P}_{m,k}(A)$ indicates the GMRES polynomial that is implicitly constructed in m steps of GMRES when solving $Ay = r_{k-1}$.

Figure 2: The GMRESR algorithm

However, they compute different direction vectors u_k. GCR sets u_k simply to r_{k-1}, while GMRESR computes u_k by applying m steps of GMRES to r_{k-1} (represented by $\mathcal{P}_{m,k}(A)r_{k-1}$ in Fig. 2). The inner GMRES iteration computes a new search direction by approximately solving the residual equation and then the outer GCR iteration minimizes the residual over the new search direction and all previous search directions u_i. The algorithm can be explained as follows.

Assume we are given the system of equations $Ax = b$, where A is a real, nonsingular, linear $(n \times n)$-matrix and b is a n-vector. Let U_k and C_k be two $(n \times k)$-matrices for which

$$C_k = AU_k, \quad C_k^T C_k = I_k, \tag{1}$$

and let x_0 be an initial guess. For $x_k - x_0 \in \mathit{range}(U_k)$ the minimization problem

$$\|b - Ax_k\|_2 = \min_{x \in \mathit{range}(U_k)} \|r_0 - Ax\|_2. \tag{2}$$

is solved by

$$x_k = x_0 + U_k C_k^T r_0 \tag{3}$$

and $r_k = b - Ax_k$ satisfies

$$r_k = r_0 - C_k C_k^T r_0, \quad r_k \perp \mathit{range}(C_k)\ . \tag{4}$$

In fact we have constructed the inverse of the restriction of A to $\mathit{range}(U_k)$ onto $\mathit{range}(C_k)$. This inverse is given by

$$A^{-1} C_k C_k^T = U_k C_k^T. \tag{5}$$

This principle underlies the GCR method. In GCR the matrices $U_k = [u_1\, u_2 \ldots u_k]$ and $C_k = [c_1\, c_2 \ldots c_k]$ are constructed such, that $\mathit{range}(U_k)$ is equal to the Krylov subspace $K^k(A; r_0) = \mathit{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\}$. Provided GCR does not break down; i.e. if $c_k \not\perp r_{k-1}$, it is a finite method and at step k it solves the minimization problem (2).

Consider the k-th step in GCR. Equations (1)-(3) indicate that if in the update $u_k = r_{k-1}$ (in GCR), we replace r_{k-1} by any other vector, then the algorithm still solves (2); however, the subspace U_k will be different. The optimal choice would be $u_k = e_{k-1}$, where e_{k-1} is the error in x_{k-1}. In order to find approximations to e_{k-1}, we use the relation $Ae_{k-1} = r_{k-1}$ and *any* method which gives an approximate solution to this equation can be used to find acceptable choices for u_k. In the GMRESR algorithm GMRES(m) is chosen to be the method to find such an approximation.

However, since we already have an optimal x_{k-1}, such that $x_{k-1} - x_0 \in \ range(U_{k-1})$, we need an approximation u_k to e_{k-1}, such that $c_k = Au_k$ is orthogonal to $range(C_{k-1})$. Such an approximation is computed explicitly by the orthogonalization loop in the outer GCR iteration. Because in GMRESR this is not taken into account in the inner GMRES iteration, a less than optimal minimization problem is solved, leading to suboptimal corrections [2] to the residual. Another disadvantage of GMRESR is that the inner iteration is essentially a restarted GMRES. It therefore also displays some of the problems of restarted GMRES. Most notably it can have the tendency to stagnate (see NUMERICAL EXPERIMENTS).

From this we infer that we should preserve the orthogonality of the correction to the residual also in the inner GMRES iteration. In order to do this we use $A_{k-1} = (I - C_{k-1}C_{k-1}^T)A$ as the operator in the inner iteration. This gives the proper corrections to the residual: $c_k \in K^m(A_{k-1}; A_{k-1}r_{k-1})$. However, the corresponding corrections to the approximate solution (contrary to ordinary implementations of Krylov methods) are found by $u_k = A^{-1}c_k \in A^{-1}K^m(A_{k-1}; A_{k-1}r_{k-1})$. These corrections can be computed since the inverse of A is known over this space. Equation (5) gives:

$$A^{-1}A_{k-1} = A^{-1}A - A^{-1}C_{k-1}C_{k-1}^T A = I - U_{k-1}C_{k-1}^T A. \tag{6}$$

This leads to a variant of the GMRESR iteration scheme, which has an improved performance for many problems.

In this article we will consider GMRES and BICGSTAB as inner methods. In the next section we will discuss the implications of the orthogonalization in the inner method. It will be proved that this leads to an optimal approximation over the space spanned by both the outer and the inner iteration vectors. It also introduces a potential problem: the possibility of breakdown in the generation of the Krylov space in the inner iteration, since we iterate with a singular operator. We will show, however, that such a breakdown can never happen before a specific (generally large) number of iterations. Furthermore, we will also show how to remedy such a breakdown. We will also discuss the efficient implementation of these methods and how we can truncate the outer GCR iteration. Outlines of the algorithms can be found in [7], [2].

CONSEQUENCES OF INNER ORTHOGONALIZATION

To keep this section concise, we will only give a short indication of the proofs or omit them completely. The proofs can be found in [2]. Throughout the rest of this article we will use the following notations:

- By $U_k = [u_1 \dots u_k]$ and $C_k = [c_1 \dots c_k]$ we denote matrices that satisfy the relations (1);
- By x_k and r_k we denote the vectors that satisfy the relations (2)-(4);
- By P_k and Q_k we denote the projections defined as $P_k = C_k C_k^T$ and $Q_k = U_k C_k^T A$;
- By A_k we denote the operator defined as $A_k = (I - P_k)A$;
- By $V_m = [v_1 \dots, v_m]$ we denote the orthonormal matrix generated by m steps of Arnoldi (GMRES) with A_k and such that $v_1 = r_k/\|r_k\|_2$.

From this and (6) it then follows that

$$AQ_k = P_k A, \quad \text{and} \quad A^{-1}A_k = (I - Q_k). \tag{7}$$

We will describe the $(k+1)$-th step of our variant of the GMRESR iteration scheme, where in the inner GMRES iteration the modified operator A_k is used. We use m (not fixed) steps of the GMRES algorithm to compute the correction to r_{k+1} in the space $K^m(A_k; A_k r_k)$. This leads to the optimal correction to the approximate solution x_{k+1} over the 'global' space $range(U_{k+1}) \oplus A^{-1}K^m(A_k; A_k r_k)$.

Theorem 1 *The Arnoldi process in the inner GMRES iteration defines the relation $A_k V_m = V_{m+1}\bar{H}_m$, with $\bar{H}_m$ an $((m+1) \times m)$ Hessenberg matrix. Let y be defined by*

$$y : \min_{\tilde{y} \in \mathbb{R}^m} \|r_k - A_k V_m \tilde{y}\|_2 = \min_{\tilde{y} \in \mathbb{R}^m} \|r_k - V_{m+1}\bar{H}_m \tilde{y}\|_2. \tag{8}$$

Then the minimal residual solution of the inner GMRES iteration: $(A^{-1}A_k V_m y)$ gives the outer approximation

$$x_{k+1} = x_k + (I - Q_k)V_m y, \tag{9}$$

which is also the solution to the 'global' minimization problem

$$x_{k+1} : \min_{\substack{\tilde{x} \in range(U_k) \oplus \\ range(V_m)}} \|b - A\tilde{x}\|_2 \tag{10}$$

It also follows from this theorem that the GCR optimization (in the outer iteration) is given by (9), so that the residual computed in the inner GMRES iteration equals the residual of the outer GCR iteration: $r_{k+1} = b - Ax_{k+1} = b - Ax_k - A_k V_m y = r_k - A_k V_m y$. From this it follows that in the outer GCR iteration the vectors u_{k+1} and c_{k+1} are given by

$$c_{k+1} = (A_k V_m y)/\|A_k V_m y\|_2, \tag{11}$$

$$u_{k+1} = ((I - Q_k)V_m y)/\|A_k V_m y\|_2. \tag{12}$$

Note that $(I - Q_k)V_m y$ has been computed already as the the approximate solution in the inner GMRES iteration; see (9), and $A_k V_m y$ is easily computed from the relation $A_k V_m y = V_{m+1}\bar{H} y_m$. Moreover, as a result of using GMRES in the inner iteration, the norm of the residual r_{k+1} as well as the norm of $A_k V_m y$ is already known at no extra computational costs. Consequently, the outer GCR iteration becomes very simple.

We will now consider the possibility of breakdown when generating a Krylov space with a singular, nonsymmetric operator. Although GMRES is still optimal in the sense that at each iteration it delivers the minimum residual solution over the generated Krylov subspace, the generation of the Krylov subspace itself, from a singular operator, may terminate too early. The following simple example shows that this may happen before the solution is found, even when the solution and the right hand side are both in the range of the given (singular) operator and in the orthogonal complement of its null-space.

Define the matrix $A = (e_2\ e_3\ e_4\ 0)$, where e_i denotes the i-th Cartesian basis vector. Note that $A = (I - e_1 e_1^T)(e_2\ e_3\ e_4\ e_1)$, which is the same type of operator as A_k, an orthogonal projection times a nonsingular operator. Now consider the system of equations $Ax = e_3$. Then GMRES (or any other Krylov method) will search for a solution in the space

$$span\{e_3, Ae_3, A^2e_3, \ldots\} = span\{e_3, e_4, 0, 0, \ldots\}\ .$$

So we have a breakdown of the Krylov space and the solution is not contained in it. We remark that the singular unsymmetric case is quite different from the symmetric one.

In the remainder of this section we will prove that a breakdown in the inner GMRES method cannot occur before the total number of iterations exceeds the dimension of the Krylov space $K(A; r_0)$. This means that, in practice, a breakdown will be rare. Furthermore, we will show how such a breakdown can be overcome.

We will now define breakdown of the Krylov space for the inner GMRES iteration more formally.

Definition 1 *We say there is a breakdown of the Krylov subspace in the inner GMRES iteration if* $A_k v_m \in range(V_m)$ *, since this implies we can no longer expand the Krylov subspace. We call it a* **lucky breakdown** *if* $v_1 \in range(A_k V_m)$ *, because we then have found the solution (the inverse of A is known over the space* $range(A_k V_m)$ *). We call it a* **true breakdown** *if* $v_1 \notin range(A_k V_m)$ *, because then the solution is not contained in the Krylov subspace.*

The following theorem relates true breakdown to the invariance of the sequence of subspaces in the inner method for the operator A_k. Part four indicates that it is always known whether a breakdown is true or lucky.

Theorem 2 *The following statements are equivalent:*

1. *A true breakdown occurs in the inner GMRES iteration at step m;*
2. $range(A_k V_{m-1})$ *is an invariant subspace of* A_k*;*
3. $A_k v_m \in range(A_k V_{m-1})$ *;*
4. $A_k V_m = V_m H_m$*, and* H_m *is a singular* $m \times m$ *matrix.*

From theorem 1, one can already conclude that a true breakdown occurs if and only if A_k is singular over $K^m(A_k; r_k)$. From the definition of A_k we know $null(A_k) = range(U_k)$. We will make this more explicit in the following theorem, which relates true breakdown to the intersection of the inner search space and the outer search space.

Theorem 3 *A true breakdown occurs if and only if*

$$range(V_m) \cap range(U_k) \neq \{0\}.$$

The following theorem indicates that no true breakdown in the inner GMRES iteration can occur before the total number of iterations exceeds the dimension of the Krylov space $K(A; r_0)$.

Theorem 4 *Let* $m = \dim(K(A; r_0))$ *and let* l *be such that* $r_k = \mathcal{P}_l(A) r_0$ *for some polynomial* $\mathcal{P}_l$ *of degree* l*. Then*

$$\dim(K^{j+1}(A_k; r_0)) = j + 1 \quad for\ j + l < m$$

and therefore no true breakdown occurs in the first j *steps of the inner GMRES iteration.*

We will now show how a true breakdown can be overcome. There are basically two ways to continue:
In the inner iteration: by finding a suitable vector to expand the Krylov space.

In the outer iteration: by computing the solution of the inner iteration just before the true breakdown and then by making one LSQR-step (see below) in the outer iteration.

We will consider the continuation in the inner GMRES iteration first. The following theorem indicates how one can continue the generation of the Krylov space $K(A; r_k)$ if in the inner GMRES iteration a true breakdown occurs.

Theorem 5 *If a true breakdown occurs in the inner GMRES iteration then*

$$\exists c \in \ range(C_k) \ : \ A_k c \notin \ range(A_k V_{m-1}) \tag{13}$$

This implies that one can try the vectors c_i until one of them works. However, one should realize that the minimization problem (8) is slightly more complicated.

Another way to continue after a true breakdown in the inner GMRES iteration is to compute the inner iteration solution just before the breakdown and then apply an LSQR-switch (see below) in the outer GCR iteration. The following theorem states the reason why one has to apply an LSQR-switch.

Theorem 6 *Suppose one computes the solution of the inner GMRES iteration just before a true breakdown. Then stagnation will occur in the next inner iteration, that is* $r_{k+1} \perp K(A_{k+1}; r_{k+1})$. *This will lead to a breakdown of the outer GCR iteration.*

The reason for this stagnation in the inner GMRES iteration is that the new residual r_{k+1} remains in the same Krylov space $K(A_k; r_k)$, which contains a $u \in \ range(U_k)$. So we have to 'leave' this Krylov space. We can do this using the so-called LSQR-switch, which was introduced in [15], to remedy stagnation in the inner GMRES iteration. Just as in the GMRESR method, stagnation in the inner GMRES iteration will result in a breakdown in the outer GCR iteration, because the residual cannot be updated. The following theorem states that this LSQR-switch actually works.

Theorem 7 *If stagnation occurs in the inner GMRES iteration, that is if* $\min_{\tilde{y} \in \mathbb{R}^m} \|r_{k+1} - A_k V_m \tilde{y}\|_2 = \|r_{k+1}\|_2$, *then one can continue by setting (LSQR-switch)*

$$c_{k+2} = \gamma A_{k+1} A^T r_{k+1} \quad \textit{and} \tag{14}$$

$$u_{k+2} = \gamma (I - Q_{k+1}) A^T r_{k+1}, \tag{15}$$

where $\gamma = \|c_{k+2}\|_2^{-1}$. *This leads to*

$$r_{k+2} = r_{k+1} - (r_{k+1}^T c_{k+2}) c_{k+2} \quad \textit{and} \tag{16}$$

$$x_{k+2} = x_{k+1} - (r_{k+1}^T c_{k+2}) u_{k+2}, \tag{17}$$

which always gives an improved approximation. Therefore, these vectors can be used as the start vectors for a new inner GMRES iteration.

IMPLEMENTATION

We will now describe how to implement these methods efficiently (see also [2],[7]). First we will discuss the outer GCR iteration and then the inner GMRES iteration. The implementation of a method like

BICGSTAB in the inner iteration will then be obvious. Instead of the matrices U_k and C_k we will use in the actual implementation the matrices $\hat{U}_k$, $\bar{C}_k$, N_k, Z_k and the vector d_k which are defined below.

Definition 2 *The matrices $\hat{U}_k$, $\bar{C}_k$, N_k, Z_k and the vector d_k are defined as follows.*

$$\begin{aligned} C_k &= \bar{C}_k N_k, \quad \textit{where} && (18)\\ N_k &= diag(\|\bar{c}_1\|_2^{-1}, \|\bar{c}_2\|_2^{-1}, \ldots, \|\bar{c}_k\|_2^{-1}), && (19)\\ A\hat{U}_k &= \bar{C}_k Z_k, && (20) \end{aligned}$$

where Z_k is assumed to be upper-triangular. Finally d_k is defined by the relation

$$r_k = r_0 - \bar{C}_k d_k \tag{21}$$

From this the approximate solution x_k, corresponding to r_k, is implicitly represented as

$$x_k = x_0 + \hat{U}_k Z_k^{-1} d_k. \tag{22}$$

Using this relation x_k can be computed at the end of the complete iteration or before truncation (see next section). The implicit representation of U_k saves all the intermediate updates of previous u_i to a new u_{k+1}, which is approximately 50% of the computational costs in the outer GCR iteration (see (11) and (12)).

GMRES as inner iteration. After k outer GCR iterations we have $\hat{U}_k$, $\bar{C}_k$ and r_k. Then, in the inner GMRES iteration, the orthogonal matrix V_{m+1} is constructed such that $\bar{C}_k^T V_{m+1} = O$ and

$$\begin{aligned} AV_m &= \bar{C}_k B_m + V_{m+1}\bar{H}_m && (23)\\ B_m &= N_k^2 \bar{C}_k^T A V_m && (24) \end{aligned}$$

This algorithm is equivalent to the usual GMRES algorithm, except that the vectors Av_i are first orthogonalized on $\bar{C}_k$. From (23) and (24) it is obvious that $AV_m - \bar{C}_k B_m = A_k V_m = V_{m+1}\bar{H}_m$ (cf. theorem 1). Next we compute y according to (8) and we set (cf. (11) without normalization):

$$\begin{aligned} \bar{c}_{k+1} &= V_{m+1}\bar{H}_m y && (25)\\ \hat{u}_{k+1} &= V_m y. && (26) \end{aligned}$$

This leads to $A\hat{u}_{k+1} = AV_m y = \bar{C}_k B_m y + V_{m+1}\bar{H}_m y = \bar{C}_k B_m y + \bar{c}_{k+1}$, so that if we set $z_{k+1} = ((B_m y)^T\, 1)^T$ the relation $A\hat{U}_{k+1} = \bar{C}_{k+1} Z_{k+1}$ is again satisfied. It follows from theorem 1 that the new residual of the outer GCR iterations is equal to the final residual of the inner iteration $r_{k+1} = r_m^{inner}$ and is given by $r_{k+1} = r_k - \bar{c}_{k+1}$, so that $d_{k+1} = 1$. Obviously the residual norm only needs to be computed once. If we replace, in the formula above, the new residual of the outer GCR iteration r_{k+1} by the residual of the inner GMRES iteration r_m^{inner}, we see an important relation that holds more generally $\bar{c}_{k+1} = r_k - r_m^{inner}$. This relation is important, since in general (when other Krylov methods are used for the inner iteration) $\bar{c}_{k+1}$ or c_{k+1} cannot be computed from u_{k+1}, because u_{k+1} is not always computed explicitly, nor does a relation like (25) always exist. Finally, we need to compute the new coefficient of N_{k+1}, $\|\bar{c}_{k+1}\|_2^{-1}$ in order to satisfy the relations in definition 2.

TRUNCATION

In practice, since memory space may be limited and since the method becomes increasingly expensive for large k (the number of outer search vectors), we want to truncate the set of outer iteration vectors ($\hat{u}_i$) and

$(\bar{c}_i)$ at $k = k_{max}$, where k_{max} is some positive integer. Basically, there are two ways to do this: one can discard one or more iteration vector(s) (dropping) or one can assemble two or more iteration vectors into one single iteration vector (assembly). We will first discuss the strategy for truncation and then its implementation.

A strategy for Truncation. In each outer GCR iteration step the matrices $\hat{U}_k$ and $\bar{C}_k$ are augmented with one extra column. To keep the memory requirement constant, at step $k = k_{max}$, it is therefore sufficient to diminish the matrices $\hat{U}_{k_{max}}$ and $\bar{C}_{k_{max}}$ by one column. From (22) we have $x_k = x_0 + \hat{U}_k Z_k^{-1} d_k$. Denote $\xi_k = Z_k^{-1} d_k$. Consider the sequence of vectors (ξ_k). The components $\xi_k{}^{(i)}$ of these vectors ξ_k are the coefficients for the updates $\hat{u}_i$ of the approximate solution x_k. These coefficients $\xi_k{}^{(i)}$ converge to the limits $\xi^{(i)}$ as k increases. Moreover, $(\xi_k{}^{(1)})$ converges faster than $(\xi_k{}^{(2)})$, and $(\xi_k{}^{(2)})$ converges faster than $(\xi_k{}^{(3)})$ etc. . Suppose that the sequence $(\xi_k{}^{(1)})$ has converged to $\xi^{(1)}$ within machine precision. From then on it makes no difference for the computation of x_k when we perform the update $x_0 + \xi^{(1)}\hat{u}_1$. In terms of direction vectors this means that the outer direction vector $\hat{u}_1$ will not reenter as component in the inner iteration process. Therefore one might hope that discarding the vector $\bar{c}_1$ will not spoil the convergence. This leads to the idea of dropping the vector $\bar{c}_1 (= A\hat{u}_1)$ or of assembling $\bar{c}_1$ with $\bar{c}_2$ into $\tilde{c}$ (say) when

$$\delta(k) = \left| \frac{\xi_{k-1}^{(1)} - \xi_k^{(1)}}{\xi_k^{(1)}} \right| < \epsilon, \tag{27}$$

where $\epsilon > 0$ is a small constant. The optimal ϵ, which may depend on k, can be determined from experiments. When $\delta(k) > \epsilon$ we drop $\bar{c}_{k_{max-1}}$ or we assemble $\bar{c}_{k_{max-1}}$ and $\bar{c}_{k_{max}}$ (of course other choices are feasible as well, but we will not consider them in this article). With this strategy we hope to avoid stagnation by keeping the most relevant part of the subspace $range(C_k)$ in store as a subspace of dimension $k-1$. In the next subsections we describe how to implement this strategy and its consequences for the matrices $\bar{C}_k$ and $\hat{U}_k$.

Dropping a vector. Let $1 \leq j \leq k = k_{max}$. Dropping the column $\bar{c}_j$ is easy. We can discard it without consequences. So let $\bar{C}'_{k-1}$ be the matrix $\bar{C}_k$ without the column $\bar{c}_j$. Dropping a column from $\hat{U}_k$ needs more work, since x_k is computed as $x_k = x_0 + \hat{U}_k Z_k^{-1} d_k$. Moreover, in order to be able to apply the same strategy in the next outer iteration we have to be able to compute x_{k+1} in a similar way. For that purpose, assume that x_k can be computed as

$$x_k = x'_{k-1} = x_0{}' + \hat{U}'_{k-1}(Z'_{k-1})^{-1} d'_{k-1}, \tag{28}$$

where $\hat{U}'_{k-1}$ and Z'_{k-1} are matrices such that $A\hat{U}'_{k-1} = \bar{C}'_{k-1} Z'_{k-1}$ (see (20)). These matrices $\hat{U}'_{k-1}$ and Z'_{k-1} are easily computed by using the j-th row of (20) to eliminate the j-th column of $\bar{C}_k$ in (20). In order to determine x'_0 and d'_{k-1} we introduce the matrix $\bar{U}_k = A^{-1}\bar{C}_k = \hat{U}_k Z_k^{-1}$. This enables us to write

$$x_k = (x_0 + d_k^{(j)} \bar{u}_j) + \sum_{\substack{i=1 \\ i \neq j}}^{k} d_k^{(i)} \bar{u}_i \qquad \text{and} \qquad \bar{u}_j = (\hat{u}_j - \sum_{i=1}^{j-1} z_{ij} \bar{u}_i)/z_{jj}. \tag{29}$$

Substituting the equation for $\bar{u}_j$ into the equation for x_k we can compute x_k from

$$x_k = (x_0 + \frac{d_k^{(j)}}{z_{jj}} \hat{u}_j) + \sum_{i=1}^{j-1} (d_k^{(i)} - d_k^{(j)} \frac{z_{ij}}{z_{jj}}) \bar{u}_i + \sum_{i=j+1}^{k} d_k^{(i)} \bar{u}_i. \tag{30}$$

Notice that this equation precisely defines x_0' and d_{k-1}':

$$\begin{aligned} x_0' &= x_0 + (d_k{}^{(j)}/z_{jj}), \\ d_{k-1}^{(i)\prime} &= d_k^{(i)} - d_k^{(j)}(z_{ij}/z_{jj}) \qquad \text{for } i = 1,\ldots,j-1 \text{ and} \\ d_{k-1}^{(i)\prime} &= d_k^{(i+1)} \qquad\qquad\qquad\quad \text{for } i = j,\ldots,k-1. \end{aligned} \tag{31}$$

Now we have deallocated two vectors and we compute x_k as in (28). We can continue the algorithm.

Assembly of two vectors. Let $1 \le j < l \le k = k_{max}$. Again assembling $\bar{c}_j$ and $\bar{c}_l$ is easy. Let $\tilde{c} = (d_k^{(j)}\bar{c}_j + d_k^{(l)}\bar{c}_l)$ overwrite the l-th column of $\bar{C}_k$. Then, let $\bar{C}_{k-1}'$ be this new matrix $\bar{C}_k$ without j-th column. Analogous to the above, we wish to compute x_k as (28). For the purpose of determining the matrices $\widehat{U}_{k-1}'$ and Z_{k-1}', let $\tilde{u} = (d_k^{(j)}\bar{u}_j + d_k^{(l)}\bar{u}_l)$ and compute $t_1^{(m)}$ and $t_2^{(m)}$ such that $z_{jm}\bar{u}_j + z_{lm}\bar{u}_l + t_1^{(m)}\bar{u}_j = t_2^{(m)}\tilde{u}$, which gives $t_1^{(m)} = z_{lm}(d_k^{(j)}/d_k^{(l)}) - z_{jm}$ and $t_2^{(m)} = z_{lm}/d_k^{(l)}$. This enables us to write $\hat{u}_m = \sum_{i=1}^{m} z_{im}\bar{u}_i$, for $m = 1,\ldots,j-1$ and

$$\hat{u}_m = \sum_{\substack{i=1 \\ i\neq j,l}}^{m} z_{im}\bar{u}_i + t_2^{(m)}\tilde{u} - t_1^{(m)}\bar{u}_j, \qquad \text{for } m = j,\ldots,k. \tag{32}$$

Substituting $\bar{u}_j = (\hat{u}_j - \sum_{i=1}^{j-1} z_{ij}\bar{u}_i)/z_{jj}$, to eliminate $\bar{u}_j$ from (32) we get $\hat{u}_m = \sum_{i=1}^{m} z_{im}\bar{u}_i$, for $m = 1,\ldots,j-1$ and

$$\hat{u}_m + \frac{t_1^{(m)}}{z_{jj}}\hat{u}_j = \sum_{\substack{i=1 \\ i\neq j,l}}^{m} (z_{im} + t_1^{(m)}\frac{z_{ij}}{z_{jj}})\bar{u}_i + t_2^{(m)}\tilde{u} \qquad \text{for } m = j+1,\ldots,k. \tag{33}$$

This equation determines the matrices $\widehat{U}_{k-1}'$ and Z_{k-1}'. In order to determine x_0' and d_{k-1}', note that x_k can be computed as

$$x_k = x_0 + \sum_{\substack{i=1 \\ i\neq j,l}}^{k} d_k^{(i)}\bar{u}_i + \tilde{u}. \tag{34}$$

Therefore x_0' is just x_0 and d_{k-1}' equals the vector d_k without the j-th element and the l-th element overwritten by 1. Similarly, as before, we have deallocated two vectors from memory. The assembled vectors $\tilde{u}$ and $\tilde{c}$ overwrite $\hat{u}_l$ and $\hat{c}_l$. The locations of $\hat{u}_j$ and $\hat{c}_j$ can therefore be used in the next step. Finally, we remark that these computations can be done with rank one updates.

NUMERICAL EXPERIMENTS

We will discuss the results of some numerical experiments, which concern the solution of two dimensional convection diffusion problems on regular grids, discretized using a finite volume technique, resulting in a pentadiagonal matrix. The system is preconditioned with ILU applied to the scaled system; see [3],[9]. The first two problems are used to illustrate and compare the following solvers:

- (full) GMRES;
- BICGSTAB;
- GMRESR(m), where m indicates the number of inner GMRES iterations between the outer iterations;
- GCRO(m), which is GCR with m adapted GMRES iterations as inner method, using A_k;
- GMRESRSTAB, which is GMRESR with BICGSTAB as inner method;

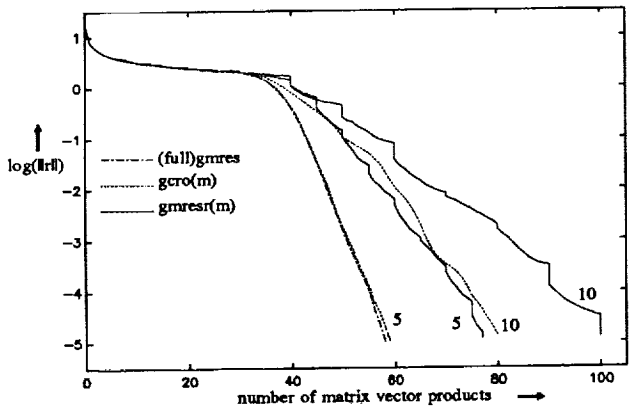

Figure 3: Convergence history for problem 1

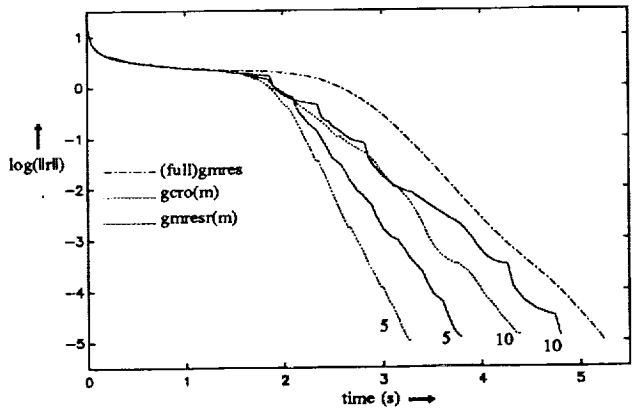

Figure 4: Convergence in time for problem 1

• and GCROSTAB, which is GCR with the adapted BICGSTAB as inner method, using A_k.

We will compare the convergence of these methods both with respect to the number of matrix vector products and with respect to CPU-time on one processor of the Convex 3840. This means e.g. that each step of BICGSTAB (and variants) is counted for two matrix vector products. We give both these convergence rates because the main trade off between (full) GMRES, the GCRO variants and the GMRESR variants is less iterations against more dot products and vector updates per iteration. Any gain in CPU-time then depends on the relative cost of the matrix vector multiplication and preconditioning versus the orthogonalization cost on the one hand and on the difference in iterations on the other hand. We will use our third problem to show the effects of truncation and compare two strategies.

Problem 1. This problem comes from the discretization of

$$-(u_{xx} + u_{yy}) + bu_x + cu_y = 0$$

on $[0,1] \times [0,4]$, where

$$b(x,y) = \begin{cases} 100 & \text{for } 0 \le y < 1 \text{ and } 2 \le y < 3 \\ -100 & \text{for } 1 \le y < 2 \text{ and } 3 \le y \le 4 \end{cases}$$

and $c = 100$. The boundary conditions are $u = 1$ on $y = 0$, $u = 0$ on $y = 4$, $u' = 0$ on $x = 0$ and $u' = 0$ on $x = 1$, where u' denotes the (outward) normal derivative. The stepsize in x-direction is $1/100$ and in y-direction is $1/50$.

In this example we compare the performances of GMRES, GCRO(m) and GMRESR(m), for $m = 5$ and $m = 10$. The convergence history of problem 1 is given in Fig. 3 and Fig. 4. Fig. 3 shows that GMRES converges fastest (in matrix vector products), which is of course to be expected, followed by GCRO(5), GMRESR(5), GCRO(10) and GMRESR(10). From Fig. 3 we also see that GCRO(m) converges smoother and faster than GMRESR(m). Note that GCRO(5) has practically the same convergence behavior as GMRES. The vertical 'steps' of GMRESR(m) are caused by the optimization in the outer GCR iteration, which does not involve a matrix vector multiplication. We also observe that the GMRESR(m) variants tend to lose their superlinear convergent behavior, at least during certain stages of the convergence history. This seems to be caused by stagnation or slow convergence in the inner GMRES iteration, which (of course) essentially behaves like a restarted GMRES. For GCRO(m), however, we see a much smoother and faster convergence behavior and the superlinearity of (full) GMRES is preserved. This is explained by the 'global' optimization over both the inner and the outer search vectors (the latter form a sample of the entire, previously searched Krylov subspace). So we may view this as a semi-full gmres. Fig. 4 gives the

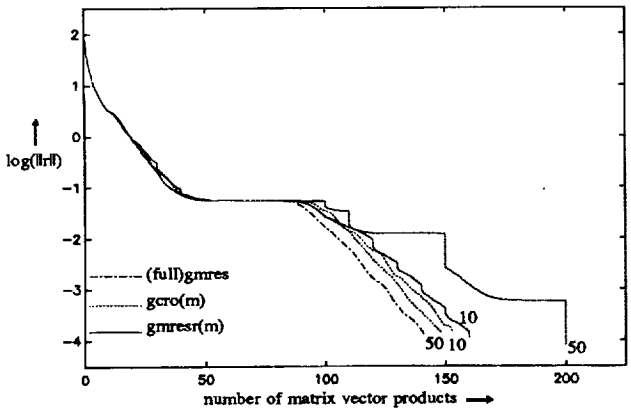

Figure 5: Convergence history for problem 2

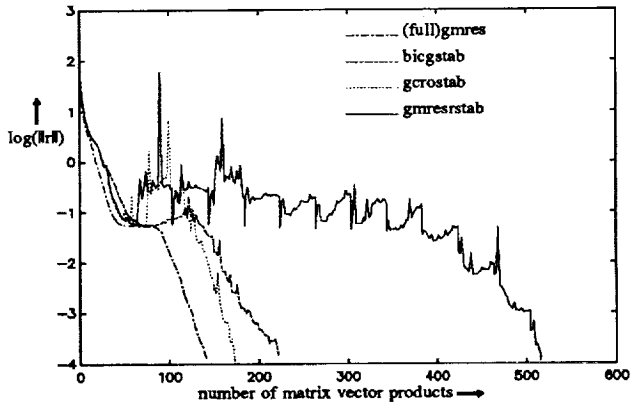

Figure 6: Convergence history for BICGSTAB variants for problem 2

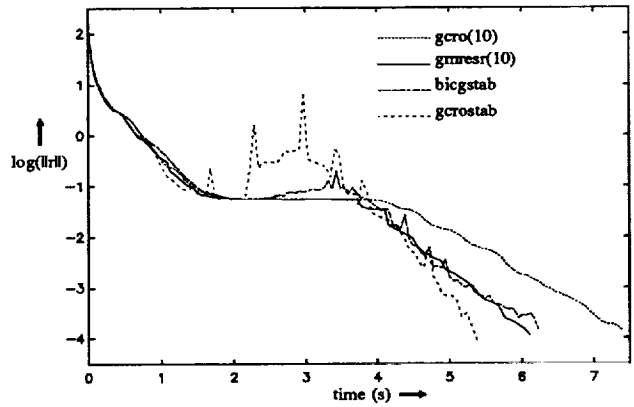

Figure 7: Convergence in time for problem 2

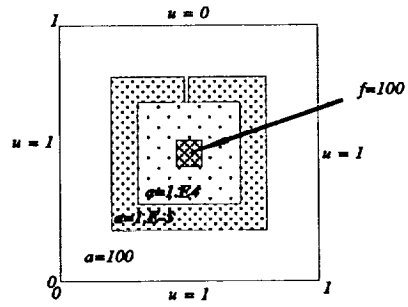

Figure 8: Coefficients for problem 2

convergence with respect to CPU-time. In this example GCRO(5) is the fastest, which is not surprising in view of the fact that it converges almost as fast as GMRES, but against much lower costs. Also, we see that GCRO(10), while slower than GMRESR(5), is still faster than GMRESR(10). In this case the extra orthogonalization costs in GCRO are outweighed by the improved convergence behavior.

Problem 2. This problem is taken from [14]. The linear system comes from the discretization of

$$-(au_x)_x - (au_y)_y + bu_x = f$$

on the unit square, with $b = 2\exp 2(x^2 + y^2)$. Along the boundaries we have Dirichlet conditions: $u = 1$ for $y = 0, x = 0$ and $x = 1$, and $u = 0$ for $y = 1$. The functions a and f are defined as shown in Fig. 8; $f = 0$ everywhere, except for the small subsquare in the center where $f = 100$. The stepsize in x-direction and in y-direction is $1/128$.

If Fig. 5 a convergence plot is given for (full) GMRES, GCRO(m) and GMRESR(m). We used $m = 10$ and $m = 50$ to illustrate the difference in convergence behavior in the inner GMRES iteration of GMRESR(m) and GCRO(m). GMRESR(50) stagnates in the inner GMRES iteration whereas GCRO(50) more or less displays the same convergence behavior as GCRO(10) and full GMRES. For the number of matrix vector products, it seems that for GMRESR(m) small m are the best choice.

In Fig. 6 a convergence plot is given for (full) GMRES, BICGSTAB, and the the BICGSTAB variants, GMRESRSTAB and GCROSTAB. To our experience the following strategy gave the best results for the BICGSTAB variants:

- For GMRESRSTAB we ended an inner iteration after either 20 steps or a relative improvement of the residual of 0.01;
- For GCROSTAB we ended an inner iteration after either after 25 steps or a relative improvement of the residual of 0.01.

The convergence of GMRESRSTAB for this example is somewhat typical for GMRESRSTAB in general (albeit very bad in this case). This might be explained from the fact that the convergence of BICGSTAB depends on a 'shadow' Krylov subspace, which it implicitly generates. Now, if if one restarts, then BICGSTAB also starts to build a new, possibly different, 'shadow' Krylov subspace. This may lead to erratically convergent behavior in the first few steps. Therefore, it may happen that, if in the inner iteration BICGSTAB does not converge (to the relative precision), the 'solution' of the inner iteration is not very good and therefore the outer iteration may not give much improvement either. At the start the same more or less holds for GCROSTAB; however, after a few outer GCR iterations the 'improved' operator (A_k) somehow yields a better convergence than BICGSTAB by itself. This was also observed for more tests, although it also may happen that GCROSTAB converges worse than BICGSTAB.

In Fig. 7 a convergence plot versus the CPU-time is given for GCROSTAB, BICGSTAB, GCRO(10) and GMRESR(10). The fastest convergence in CPU-time is achieved by GCROSTAB(10), which is $\approx 20\%$ faster than BICGSTAB notwithstanding the extra work in orthogonalizations. We also see, that although GCRO(10) takes fewer iterations than GMRESR(10), in CPU-time the latter is faster. So in this case the decrease in iterations does not outweigh the extra work in orthogonalizations. For completeness we mention that GMRESRSTAB took almost 15 seconds to converge, whereas GMRES took almost 20 seconds.

Problem 3. The third problem is taken from [10]. The linear system stems from the discretization of the partial differential equation

$$-u_{xx} - u_{yy} + 1000(xu_x + yu_y) + 10u = f$$

on the unit square with zero Dirichlet boundary conditions. The stepsize in both x-direction and y-direction is $1/65$. The right-hand side is selected once the matrix is constructed so that the solution is known to be $x = (1, 1, \ldots, 1)^T$. The zero vector was used as an initial guess.

In Fig. 9 we see a plot of the convergence history of full GMRES, GMRESR(5), GCRO(5) and GCRO(10,5) for two different truncation strategies, where the first parameter gives the dimension of the outer search space and the second the dimension of the inner search space. The number of vectors in the outer GCR iteration is twice the dimension of the search space. For the truncated version:

- 'da' means that we took $\epsilon = 10^{-3}$ and dropped the vectors $\hat{u}_1$ and $\bar{c}_1$ when $\delta(k) < \epsilon$ and assembled the vectors $\hat{u}_9$ and $\hat{u}_{10}$ as well as the vectors $\bar{c}_9$ and $\bar{c}_{10}$ when $\delta(k) > \epsilon$;
- 'tr' means that we dropped the vectors $\hat{u}_9$ and $\bar{c}_9$ each step ($\epsilon = 0$, see also [16]).

Notice that GCRO(5) displays almost the same convergence behavior as full GMRES. GMRESR(5) converges eventually, but only after a long period of stagnation. The truncated versions of GCRO(5) also display stagnation, but for a much shorter period. After that the 'da' version seems to converge as superlinear, whereas the 'tr' version still displays periods of stagnation, most notably at the end. This indicates that the 'da' version is more capable of keeping most of the 'convergence history' than the 'tr' version. This kind of behavior was seen in more tests: 'assembled' truncation strategies seem to work better than just discarding one or more iteration vectors.

In Table 1 we give the number of matrix vector products, the number of memory vectors and the CPU-time on a Sun workstation. From this table we see that GCRO(5) is by far the fastest method and

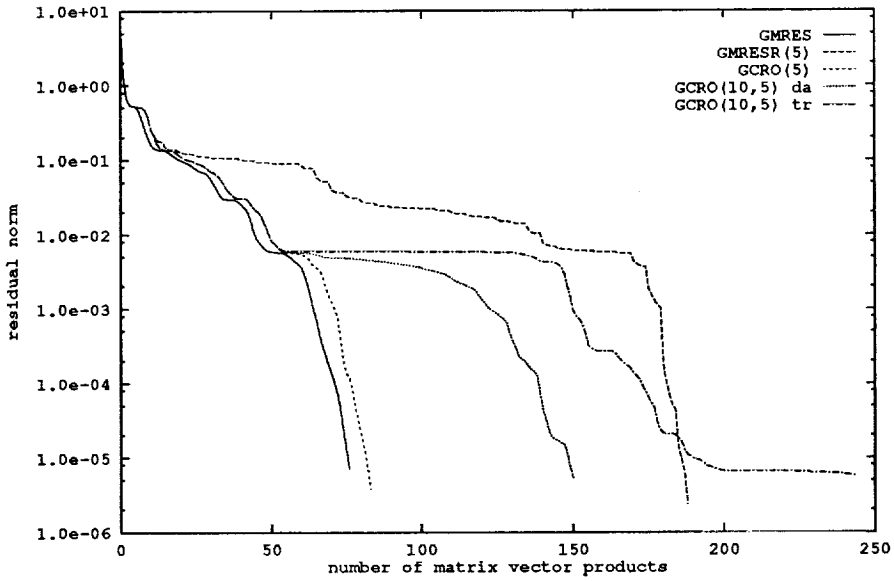

Figure 9: Convergence history for problem 3

uses about half the amount of memory vectors full GMRES and GMRESR(5) use. More interesting is that GCRO(10,5) 'da' converges in the same time as GMRESR(5), but uses only one third of the memory space.

CONCLUSIONS

We have derived from the GMRESR inner-outer iteration schemes a modified set of schemes, which preserve the optimality of the outer iteration. This optimality is lost in GMRESR since it essentially uses 'restarted' inner GMRES iterations, which do not take advantage of the outer 'convergence history'. Therefore, GMRESR may loose superlinear convergence behavior, due to stagnation or slow convergence of the inner GMRES iterations.

Method	Mat-Vec	Memory Vectors	CPU-time
GMRES	77	77	21.3
GMRESR(5)	188	81	18.5
GCRO(5)	83	39	9.4
GCRO(10,5) 'da'	150	25	18.3
GCRO(10,5) 'tr'	244	25	30.3

Table 1: Number of matrix vector products, number of memory vectors and CPU-time in seconds for problem 3

In contrast, the GCRO variants exploit the 'convergence history' to generate a search space that has no components in any of the outer directions in which we have already minimized the error. For GCRO(m) this means we minimize the error over both the inner search space and a sample of the entire previously searched Krylov subspace (the outer search space), resulting in a semi-full GMRES. This probably leads to the smooth convergence (much like GMRES) and the absence of stagnation, which may occur in the inner GMRES iteration of GMRESR. Apparently the small subset of Krylov subspace vectors that is kept approximates the entire Krylov subspace that is generated, sufficiently well. For both GMRESR(m) and GCRO(m) it seems that a small number of inner iterations works well.

We may also say, that the GCRO variants construct a new (improved) operator (of decreasing rank) after each outer GCR iteration. Although there is the possibility of breakdown in the inner method for GCRO, this seems to occur rarely as is indicated by theorem 4 (it has never happened in any of our experiments).

With respect to performance of the discussed methods we see that GCRO(m) (almost) always converges in fewer iterations than GMRESR(m). Because GCRO(m) is on average more expensive per iteration, this does not always lead to faster convergence in CPU-time. This depends on the relative costs of the matrix vector product and preconditioner w.r.t. the cost of the orthogonalizations and the reduction in iterations for GCRO(m) relative to GMRESR(m). Our experiments, with a cheap matrix vector product and preconditioner, show that already in this case the GCRO variants are very competitive with other solvers. However, especially when the matrix vector product and preconditioner are expensive or when not enough memory is available for (full) GMRES, GCRO(m) is very attractive. GCRO with BICGSTAB also seems to be a useful method, especially when a large number of iterations is necessary or when the available memory space is small relative to the problem size. GMRESR with BICGSTAB does not seem to work so well, probably because, to our observation, restarting BICGSTAB does not work so well.

We have derived sophisticated truncation strategies and shown by example that superlinear convergence behavior can be maintained. From our experience, the 'assembled' version seems to have the most promise.

Acknowledgements. The authors are grateful to Gerard Sleijpen and Henk van der Vorst for encouragement, helpful comments and inspiring discussions.

References

[1] O. Axelsson and P.S. Vassilevski. A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning. *SIAM J. Matrix Anal. Appl.*, 12:625–644, 1991.

[2] E. De Sturler. Nested Krylov methods based on GCR. Technical Report 93-.., Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1993.

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

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

[5] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:352–362, 1984.

[6] R. Fletcher. Conjugate gradient methods for indefinite systems. In G.A. Watson, editor, *Numerical Analysis Dundee 1975, Lecture Notes in Mathematics 506*, pages 73–89, Berlin, Heidelberg, New York, 1976. Springer-Verlag.

[7] D.R. Fokkema. Hybrid methods based on the GCR principle (to appear). Technical report, Mathematical Institute, University of Utrecht, Utrecht, The Netherlands, 1993.

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

[9] J.A. Meijerink and H.A. Van der Vorst. An iterative solution method for linear equations systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977.

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

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

[12] G.L. Sleijpen and D.R. Fokkema. BiCGstab(l) for linear equations involving matrices with complex spectrum. Technical Report 772, Mathematical Institute, University of Utrecht, Utrecht, The Netherlands, 1993.

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

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

[15] H.A. Van der Vorst and C. Vuik. GMRESR: A family of nested GMRES methods. Technical Report 91-80, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1991.

[16] C. Vuik. Further experiences with GMRESR. Technical Report 92-12, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1992.