Numerical Linear Algebra with Applications, Vol. 1(4), 369–386 (1994)

GMRESR: a Family of Nested GMRES Methods

H. A. Van der Vorst

Mathematical Institute, Utrecht University, Budapestlaan 6, 3584 CD Utrecht, The Netherlands

and

C. Vuik

Department of Technical Mathematics and Computer Science, Delft University, Mekelweg 4, 2602 CD Delft, The Netherlands

Recently Eirola and Nevanlinna have proposed an iterative solution method for unsymmetric linear systems, in which the preconditioner is updated from step to step. Following their ideas we suggest variants of GMRES, in which a preconditioner is constructed per iteration step by a suitable approximation process, e.g., by GMRES itself. Our numerical experiments indicate that this may lead to considerable savings in CPU-time and memory requirements in typical CFD applications.

KEY WORDS GMRES Unsymmetric linear systems Iterative solver

1. Introduction

The GMRES method, proposed in [13], is a popular method for the iterative solution of sparse linear systems with an unsymmetric nonsingular matrix. In its original form, so-called full GMRES, it is optimal in the sense that it minimizes the residual over the current Krylov subspace. However, it is often too expensive since the required orthogonalization per iteration step grows quadratically with the number of steps. For that reason, one often uses in practice variants of GMRES. The best-known variant, already suggested in [13], is to restart after each cycle of m iteration steps: GMRES(m). A disadvantage of this approach is that the convergence behavior in many situations seems to depend quite critically on the value of m (for examples see, e.g., [9]). Even in situations in which satisfactory convergence takes place, the convergence is less than optimal, since the history is thrown away so that potential superlinear convergence behavior is inhibited [18].

CCC 1070–5325/94/040369–18
Received January 1992
Revised 17 May 1994

Another approach is to apply polynomial preconditioning (in combination with a possibly available preconditioner, e.g., ILU). To that end a fixed low degree polynomial is constructed, e.g., the m-th degree iteration polynomial obtained from the first m iteration steps [11]. A disadvantage of this approach is that this polynomial does not take advantage of the current residual, i.e., this preconditioner may be strong in reducing eigenvector components that have already vanished in the iteration process.

We propose a variant of the GMRES algorithm, in which it is allowed to take a different preconditioner in each iteration step. For a special, though rather meaningful, choice we prove robustness of the resulting algorithm (GMRESR). Numerical examples have been added in order to demonstrate the possibilities of this novel approach. Though we have made a choice for the implementation of the new algorithm, it will be clear that there are other possibilities. For example, the preconditioner can be constructed differently for parts of the domain (in a PDE context), and the choice of preconditioner per domain, or even the choice of domains, may be different for each iteration step of GMRESR.

Other GMRES-like iteration schemes with a variable preconditioner have been proposed recently by Saad [14] and Axelsson and Vassilevski [2]. In Saad's scheme (FGMRES) a Krylov subspace is generated that is different from ours, and it is possible to give an example for which FGMRES suffers from breakdown (in this example it is essential that the preconditioner is different in each iteration step). Furthermore, in GMRESR it is easy to truncate the orthogonalization process, or to select specific orthogonalization directions. We also believe that the derivation of our algorithm is rather general (in fact GMRESR is only a special case), and that it gives new insights into the method, as well as into GMRES.

The method proposed in [2] is a generalized conjugate gradient method. Variant 1 in [2] (algorithm 1) produces, in exact arithmetic, and with the same innerloop as in GMRESR, identical results as GMRESR (though at considerably higher computational costs and with a Gram–Schmidt orthogonalization procedure instead of a modified version). Our proposed innerloop leads to a wider applicability of the method.

The outline of this paper is as follows. In section 2 basic ideas behind the new algorithm are presented, while in section 3 the iteration scheme for GMRESR is proposed. In section 4 some theoretical properties are analysed. Implementation aspects for specific choices of the algorithm are discussed in section 5. Finally, we present some numerical examples in section 6. These examples illustrate the effectiveness of our approach in relevant situations.

2. Rank-one updates for the preconditioner

Iterative methods can be derived from a splitting $A = H^{-1} - R$ of the matrix. In [6] the suggestion is made to update the matrix splitting with information obtained in the iteration process. We will give the flavour of this method here since it turns out that it has an interesting relation with GMRES. This relation is exploited for the construction of new classes of GMRES-like methods, that can be used as cheap alternatives for the increasingly expensive full GMRES method. One such alternative, GMRESR, will be discussed in more detail in section 3.

Assume that the matrix splitting in the kth iteration step is given by $A = H_k^{-1} - R_k$. Then we obtain the iteration formula

$$x_k = x_{k-1} + H_k r_{k-1} \quad \text{with} \quad r_k = b - Ax_k$$

The idea is now to construct H_k by a suitable rank-one update to H_{k-1}:

$$H_k = H_{k-1} + u_{k-1}v_{k-1}^T$$

which leads to

$$x_k = x_{k-1} + (H_{k-1} + u_{k-1}v_{k-1}^T)r_{k-1} \tag{2.1}$$

or

$$\begin{aligned} r_k &= r_{k-1} - A(H_{k-1} + u_{k-1}v_{k-1}^T)r_{k-1} \\ &= (I - AH_{k-1})r_{k-1} - Au_{k-1}v_{k-1}^T r_{k-1} \\ &= (I - AH_{k-1})r_{k-1} - \mu_{k-1}Au_{k-1} \end{aligned} \tag{2.2}$$

The optimal choice for the update would have been to select u_{k-1} such that

$$\mu_{k-1}Au_{k-1} = (I - AH_{k-1})r_{k-1}$$

or

$$\mu_{k-1}u_{k-1} = A^{-1}(I - AH_{k-1})r_{k-1}$$

However, A^{-1} is unknown and the best approximation we have for it is H_{k-1}. This leads to the choice

$$\bar{u}_{k-1} = H_{k-1}(I - AH_{k-1})r_{k-1} \tag{2.3}$$

The constant μ_{k-1} is chosen such that $\|r_k\|_2$ is minimal as a function of μ_{k-1}. This leads to

$$\mu_{k-1} = \frac{1}{\|A\bar{u}_{k-1}\|_2^2}(A\bar{u}_{k-1})^T(I - AH_{k-1})r_{k-1}$$

Since v_{k-1} has to be chosen such that $\mu_{k-1} = v_{k-1}^T r_{k-1}$, we have the following obvious choice for it

$$\bar{v}_{k-1} = \frac{1}{\|A\bar{u}_{k-1}\|_2^2}(I - AH_{k-1})^T A\bar{u}_{k-1} \tag{2.4}$$

(note that from the minimization property we have that $r_k \perp A\bar{u}_{k-1}$).

In principle the implementation of the method is quite straightforward, but note that the computation of r_{k-1}, $\bar{u}_{k-1}$ and $\bar{v}_{k-1}$ involves four matrix vector multiplications with A (and also some with H_{k-1}). This would make the method too expensive for being of practical interest. Also the updated splitting is most likely a dense matrix if we carry out the updates explicitly.

We will now show, still following the lines set forth in [6], that there are orthogonality properties, following from the minimization step, by which the method can be implemented much more efficiently.

We define

1. $c_k = \frac{1}{\|A\bar{u}_k\|_2}A\bar{u}_k$ (note that $r_{k+1} \perp c_k$), $u_k = \frac{1}{\|A\bar{u}_k\|_2}\bar{u}_k$
2. $E_k = I - AH_k$

From (2.2) we have that $r_k = E_k r_{k-1}$, and from (2.3):

$$A\bar{u}_k = AH_k E_k r_k = \alpha_k c_k$$

or

$$c_k = \frac{1}{\alpha_k}(I - E_k)E_k r_k = \frac{1}{\alpha_k}E_k(I - E_k)r_k \tag{2.5}$$

Furthermore:

$$\begin{aligned} E_k &= I - AH_k = I - AH_{k-1} - A\bar{u}_{k-1}\bar{v}_{k-1}^T \qquad (2.6)\\ (3) \Rightarrow \quad &= I - AH_{k-1} - A\bar{u}_{k-1}(A\bar{u}_{k-1})^T(I - AH_{k-1})\frac{1}{\|A\bar{u}_k\|_2^2}\\ &= (I - c_{k-1}c_{k-1}^T)E_{k-1}\\ &= \prod_{i=0}^{k-1}(I - c_i c_i^T)E_0 = (I - P_{k-1})E_0 \end{aligned}$$

We see that the operator E_k has the following effect on a vector. The vector is multiplied by E_0 and then orthogonalized with respect to $c_0, \ldots, c_{k-1}$. Now we have from (2.5) that

$$c_k = \frac{1}{\alpha_k}E_k y_k$$

and hence

$$c_k \perp c_0, \ldots, c_{k-1} \tag{2.7}$$

A consequence from (2.7) is that

$$\prod_{j=0}^{k-1}(I - c_j c_j^T) = I - \sum_{j=0}^{k-1} c_j c_j^T = I - P_{k-1}$$

and therefore

$$P_k = \sum_{j=0}^{k} c_j c_j^T \tag{2.8}$$

The actual implementation is based on the above properties. Given r_k we compute r_{k-1} as follows (and we update x_k in the corresponding way):

$$r_{k+1} = E_{k+1}r_k$$

With $\xi^{(0)} = E_0 r_k$ we first compute (with the c_j from previous steps):

$$E_k r_k = \xi^{(k)} \equiv (I - \sum_{j=0}^{k-1} c_j c_j^T)\xi^{(0)} = \prod_{j=0}^{k-1}(I - c_j c_j^T)\xi^{(0)}$$

The expression with $\sum$ leads to a Gram–Schmidt formulation; the expression with $\prod$ leads to Modified Gram–Schmidt.

The computed updates $-c_j^T\xi^{(0)}c_j$ for r_{k+1} correspond to updates

$$c_j^T\xi^{(0)}A^{-1}c_j = c_j^T\xi^{(0)}\bar{u}_j/\|A\bar{u}_j\|_2$$

for x_{j+1}. These updates are in the scheme, given below, represented by η.

From (2.3) we know that

$$\bar{u}_k = H_k E_k r_k = H_k \xi^{(k)}$$

Now we have to make $A\bar{u}_k \sim c_k$ orthogonal w.r.t. $c_0, ..., c_{k-1}$, and to update $\bar{u}_k$ accordingly. Once we have done that we can do the final update step to make H_{k+1}, and we can update both x_k and r_k by the corrections following from including c_k. The orthogonalization step can be carried out easily as follows. Define $c_k^{(k)} \equiv \alpha_k c_k = AH_k E_k r_k = (I - E_k)E_k r_k$ (see (2.5)) $= (I - E_0 + P_{k-1}E_0)\xi^{(k)}$ (see (2.6)) $= AH_0\xi^{(k)} + P_{k-1}(I - AH_0)\xi^{(k)} = c_k^{(0)} + P_{k-1}\xi^{(k)} - P_{k-1}c_k^{(0)}$. Note that the second term vanishes since $\xi^{(k)} \perp c_0, ..., c_{k-1}$.

The resulting scheme for the kth iteration step becomes:

```
1. $\xi^{(0)} = (I - AH_0)r_k$; $\eta^{(0)} = H_0 r_k$;
   for $i = 0, ..., k-1$ do
   $\alpha_i = c_i^T \xi^{(i)}$; $\xi^{(i+1)} = \xi^{(i)} - \alpha_i c_i$; $\eta^{(i+1)} = \eta^{(i)} + \alpha_i u_i$;
2. $u_k^{(0)} = H_0 \xi^{(k)}$; $c_k^{(0)} = Au_k^{(0)}$;
   for $i = 0, ..., k-1$ do
   $\beta_i = -c_i^T c_k^{(i)}$; $c_k^{(i+1)} = c_k^{(i)} + \beta_i c_i$; $u_k^{(i+1)} = u_k^{(i)} + \beta_i u_i$;
   $c_k = c_k^{(k)}/\|c_k^{(k)}\|_2$; $u_k = u_k^{(k)}/\|c_k^{(k)}\|_2$;
3. $x_{k+1} = x_k + \eta^{(k)} + u_k c_k^T \xi^{(k)}$;
   $r_{k+1} = (I - c_k c_k^T)\xi^{(k)}$;
```

Remarks

1. The above scheme is a Modified Gram–Schmidt variant, given in [20], of the original scheme in [6].
2. If we keep H_0 fixed, i.e., $H_0 = I$, then the method is not scaling invariant (the results for $\rho Ax = \rho b$ depend on ρ). In [20] a scaling invariant method is suggested.
3. Note that in the above implementation we have 'only' two matrix vector products per iteration step. In [20] it is argued that in many cases we may also expect comparable convergence as for GMRES in half the number of iteration steps.
4. A different choice for $\bar{u}_{k-1}$ does not change the formulas for $\bar{v}_{k-1}$ and E_{k-1}. For each different choice we can derive similar schemes as the one above.
5. From (2.2) we have

$$r_k = r_{k-1} - AH_{k-1}r_{k-1} - \mu_{k-1}Au_{k-1}$$

In view of the previous remark we might also make the different choice $\bar{u}_{k-1} = H_{k-1}r_{k-1}$. With this choice, we obtain a variant which is algebraically identical to GMRES (for a proof of this see [20]).This GMRES variant is obtained by the following changes in the previous scheme:

Take $H_0 = 0$ (note that in this case we have that $E_{k-1}r_{k-1} = r_{k-1}$, and hence we may skip part 1 of the above algorithm). and set $\xi^{(k)} = r_k$, $\eta^{(k)} = 0$. In step 2 start with $u_k^{(0)} = \xi^{(k)}$.

The result is a different formulation of GMRES in which we can obtain explicit formulas for the updated preconditioner (i.e.. the inverse of A is approximated increasingly well): The update for H_k is $\bar{u}_k c_k^T E_k$ and the sum of these updates gives an approximation for A^{-1}.

3. A recursive variant of GMRES

In the GMRES-variant discussed at the end of the previous section, we are still free to select u_k a little bit different. Remember that the leading factor H_{k-1} in (2.3) was introduced as an approximation for the actually desired A^{-1}. With $\tilde{u}_{k-1} = A^{-1}r_{k-1}$, we would have that $r_k = E_{k-1}r_{k-1} - \mu_{k-1}r_{k-1} = 0$ for the minimizing μ_{k-1}. We could take other approximations for the inverse (with respect to the given residual r_{k-1}, e.g., the result vector y obtained by a few steps GMRES applied to $Ay = r_{k-1}$. This leads to the GMRESR family of nested methods that we will describe now in more detail.

When we do the approximate solution for $Ay = r_{k-1}$ by a few steps of GMRES, then this is equivalent by stating that $A^{-1}r$ is approximated by $\mathcal{P}_m(A)r$, where $\mathcal{P}_m$ represents the polynomial that is implicitly constructed in m iteration steps of GMRES. Note that this polynomial depends on the residual r, so that we have effectively different polynomials in different steps of the (outer) iteration process. We will make this dependence explicit by adding the number of the current GMRES-EN iteration as an index to $\mathcal{P}$.

The above sketched approach leads to a nested GMRES iteration process, in which the outer iterations are formulated in the GMRES-EN way and the result of the m inner iterations, in the kth outer iteration step, is represented by $u_k^{(0)} = \mathcal{P}_{m,k}r_k$. The combined process will be referred to as GMRESR and can be represented by the following iteration scheme for the solution of $Ax = b$ (if one wants to include conventional preconditioning (e.g., ILU), then we assume that $Ax = b$ represents the explicitly preconditioned system to be solved):

GMRESR algorithm

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

Note: If the inner iteration process stagnates, i.e., if $Au_k^{(0)} - r_k = r_k$ then, in order to avoid breakdown, we replace this inner iteration process by 1 step of LSQR [12]: $u_k^{(0)} = A^T r_k$. The GMRESR algorithm with this strategy will be referred to as 'GMRESR with LSQR-switch'. In practical situations other strategies may turn out to be more effectively, and it may also be more practical to relax the switch condition (for an example, see section 6).

If $u_k^{(0)}$ in the above scheme is computed as $u_k^{(0)} = H_0 r_k$ for any fixed nonsingular

Table 1. Comparison between GMRES variants for an example

Method	matvec	daxpy	ddot	Memory	CPU time
GMRES(50)	1220	35,000	35,000	50	17.0
GMRES	184	17,000	17,000	184	7.2
GMRESR($\star,\star$,1,10)	198	1386	1224	46	1.2

preconditioner H_0, then we have precisely the GMRES-EN variant described in [20]. For $\mathcal{P}_{m,k}$ replaced by I we obtain GCR (for GCR see [7]).

The GMRESR scheme leaves us an enormous amount of freedom in designing iteration schemes. For instance, we are free to select a different m in each iteration step. This means that we could solve the inner iteration also with a specified tolerance. The inner iteration can be done with any iteration scheme, but this might prove only a theoretical advantage. For, if one has decided to solve a given system with GMRES then it seems obvious to do the inner iterations, which are done with the same operator A as the outer iterations, also by GMRES. The above scheme opens the possibility for a highly recursive scheme, since the iterations could be done with a similar scheme as above.

A more practical scheme, in our opinion, arises when the outer iteration is restarted after k iterations, just as is common practice with GMRES, in order to limit memory requirements, or to include only updates from the last j outer iterations (the truncated GMRESR version). This *truncated GMRESR* variant is obtained if we replace the for-loop by

$$\text{for } i = \max(0, k-j), \ldots, k-1 \text{ do}$$

In our limited experience a truncation strategy seems to be much more efficient than a complete restart after each j cycles. One might also discard those c_is which do not lead to a significant reduction of r_{k+1}.

For the inner iterations we suggest using $n(\varepsilon)$ cycles of GMRES(m), where $n(\varepsilon)$ may depend on a tolerance ε. The resulting scheme is denoted by GMRESR(k, j, $n(\varepsilon)$, m). If the outer iterations are not restarted this will be denoted by a $\star$, instead of k. Likewise, a $\star$ for the second parameter will denote that the process is not truncated.

Before further analyzing these schemes, we will give an example which serves to demonstrate the potential of the new class of schemes. In Table 1 we have listed the amount of work (in terms of matrix vector products, vector updates and inner products), the amount of workspace (in terms of n-vectors) and the CPU-time (in seconds, for one processor of a Convex C-240), required by some methods in order to solve a certain discretized Navier–Stokes problem [19].

Note that the new scheme, besides being more economic in memory space than its competitors, is much faster in terms of CPU-time for this specific example. This motivates us to investigate the method in more detail. A more elaborate comparison with variants of GMRES, as well as with CGS [15] and BiCGSTAB [17] will be made for relevant problems in section 6. In sections 4 and 5 we will discuss theoretical properties and implementation aspects of GMRESR.

4. Properties of GMRESR

In this section we analyze some properties of GMRESR($\star$, $\star$, 1, m), or GMRESR(m) for short. We will assume that the inner iteration is always started with initial guess $u_{k,0}^{(0)} = 0$ (note that in this notation $u_k^{(0)} = u_{k,m}^{(0)} = \mathcal{P}_{m,k}(A)r_k$). We will also assume that the inner iterations are done with GMRES; the generalization to other schemes for the inner iteration is obvious.

From the GMRESR scheme in section 3 we conclude that c_k is undefined if $c_k^{(k)} = 0$. If this happens, while $r_k \neq 0$, then we will speak of a breakdown of GMRESR. In this section we will consider the breakdown situation in more detail.

Furthermore, we will show that the breakdown situation is avoided by including the LSQR-switch strategy (see section 3), and that then GMRESR is a finite method, just as full GMRES. We will also show that it is not necessary to carry out all m inner iterations if that would imply that we are beyond the tolerance for the outer iterations. Finally, we will show that, in contrast to the original EN-method of which GMRESR has been derived, GMRESR is scaling invariant.

In this section we will assume exact arithmetic. The residuals obtained by GMRESR are denoted by an upper index GR, those of GMRES by an upper index G.

The next theorem says that GMRESR($\star$, $\star$, 1, m) is a robust method, and that it is a minimum residual method.

Theorem 4.1. *(a) 'GMRESR($\star$, $\star$, 1, m) with LSQR-switch' does not break down.*
(b) In GMRESR the residual r_k is minimized over the space

$$r_0 + span\{c_0, c_1, \dots, c_{k-1}\}$$

Proof (a) Suppose that $\|r_k^{GR} - A\mathcal{P}_{m,k}(A)r_k^{GR}\|_2 \leq \|r_k^{GR}\|_2$ (the $>$-case is precluded in GMRES), and that $r_k^{GR} \neq 0$.

We first consider the $<$-case.

Consequently it holds that $c_k^{(0)} = A\mathcal{P}_{m,k}(A)r_k^{GR} \neq 0$. Since GMRES minimizes the residual in the innerloop, we have that

$$(r_k^{GR} - c_k^{(0)}) \perp c_k^{(0)}$$

and it follows that $c_k^{(0)^T} r_k^{GR} = c_k^{(0)^T} c_k^{(0)} \neq 0$.

In the case of equality sign, the LSQR-switch is active, and with

$$u_k^{(0)} = A^T r_k^{GR} \text{ and } c_k^{(0)} = AA^T r_k^{GR}$$

we have that $c_k^{(0)^T} r_k^{GR} = (AA^T r_k^{GR})^T r_k^{GR} = \|A^T r_k^{GR}\|_2^2 \neq 0$.

The result $c_k^{(0)^T} r_k^{GR} \neq 0$, together with the fact that $r_k^{GR} \perp span\{c_0, \dots, c_{k-1}\}$, leads to $c_k^{(0)} \notin span\{c_0, \dots, c_{k-1}\}$.

Hence $\|c_k^{(k)}\|_2 \neq 0$, and the method does not break down.

(b) The minimization property of GMRESR follows straight away from the construction of the algorithm. ■

From the proof of Theorem 4.1 it follows that the GMRESR algorithm without LSQR-switch can only breakdown when the inner iteration process stagnates.

From the definition of GMRESR, in section 3, we have straight away the following result.

Lemma 4.1. *For 'GMRESR($\star$, $\star$, 1, m) with LSQR-switch' we have*

1. $r_{k+1}^{GR} = (I - P_k)r_0$, *with* $P_k = \sum_{i=0}^{k} c_i c_i^T$ *the orthogonal projection onto* $span\{c_0, \ldots, c_k\}$.

2. *GMRESR is a finite method, i.e.,* $r_k^{GR} = 0$, *for some* $k \leq n$.

The next lemma says that the c_k vectors in GMRESR are contained in a Krylov subspace. This result will facilitate the comparison between GMRESR and GMRES.

Lemma 4.2. *If GMRESR (without LSQR-switch) does not breakdown within the first k iterations and if the inner iterations are started with initial guess* $u_{k,0}^{(0)} = 0$, *then*

$$r_k^{GR} = r_0 + \sum_{i=1}^{k \cdot m} \alpha_{k,i} A^i r_0$$

and $span\{c_0, \ldots, c_k\} \subset span\{Ar_0, \ldots, A^{(k+1)m} r_0\}$.

Proof The proof is by an induction argument in k. Note that $u_0^{(0)}$ is obtained by m steps GMRES: $u_0^{(0)} \in span\{r_0, \ldots, A^{(m-1)} r_0\}$. Therefore, $c_0 = Au_0^{(0)} \in \{Ar_0, \ldots, A^m r_0\}$, which gives the result for $k = 0$.

Using that $r_{k+1}^{GR} = r_k^{GR} - c_k c_k^T r_k^{GR}$ it follows by induction that

$$r_{k+1}^{GR} = r_0 + \sum_{i=1}^{(k+1)m} \alpha_{k+1,i} A^i r_0$$

Furthermore, we note that $c_{k+1}^{(k+1)} = (I - P_k)A\mathcal{P}_{m,k+1}(A)r_{k+1}^{GR}$, and thus

$$c_{k+1}^{(k+1)} = A\mathcal{P}_{m,k+1}(A)r_{k+1}^{GR} - P_k A\mathcal{P}_{m,k+1}(A)r_{k+1}^{GR}$$

It then follows by induction that

$$c_{k+1} = \frac{c_{k+1}^{(k+1)}}{\|c_{k+1}^{(k+1)}\|_2} \in span\{Ar_0, \ldots, A^{(k+2)m} r_0\}$$

■

From Lemma 4.2 and the well-known property that GMRES minimizes the residual over its associated Krylov subspace, it follows that

$$\|r_k^{GR}\|_2 \geq \|r_{k \cdot m}^{G}\|_2$$

In section 5 we will show that the computation of x_k^{GR} and r_k^{GR} together costs $k \cdot m$ matrix vector products. This shows that GMRESR takes at least as many matrix vector products as full GMRES in order to obtain comparable accuracy. However, as we will see in section 6, it is not always that many more.

Our standard choice in GMRESR is $u_k^{(0)} = u_{k,m}^{(0)} \equiv \mathcal{P}_{m,k}(A)r_k^{GR}$. An obvious disadvantage of this choice is that always m GMRES iterations are applied in the inner iteration and that might lead to a higher accuracy than we actually need in some cases. For example, when r_k^{GR} is close to satisfying the stopping criterion $\|r_k^{GR}\|_2 \leq tol$, then we expect that the choice $u_k^{(0)} = u_{k,j}^{(0)} \equiv \mathcal{P}_{j,k}(A)r_k^{GR}$ with j (much) less than m will be sufficient to have $\|r_{k+1}^{GR}\|_2 < tol$. The following lemma states that it is never necessary to solve the inner iterations more accurately than the outer ones, and it leads to an obvious modification to GMRESR.

Lemma 4.3. *If GMRESR (without LSQR-switch) does not break down and* $\|r_k^{GR} - Au_{k,j}^{(0)}\|_2 < tol$, $j \leq m$, *then with* $u_k^{(0)} = \mathcal{P}_{j,k}(A)r_k^{GR}$ *we have that* $\|r_{k+1}^{GR}\|_2 < tol$.

Proof From Theorem 4.1 we have that

$$\|r_{k+1}^{GR}\|_2 = \min_{z \in span\{c_0,\ldots,c_k\}} \|r_0 - z\|_2 =: \|r_0 - z_{k+1}\|_2$$

Since $z_k + A\mathcal{P}_{j,k}(A)r_k^{GR} \in span\{c_0, \ldots, c_k\}$ we obtain

$$\|r_{k+1}^{GR}\|_2 = \|r_0 - z_{k+1}\|_2 \leq \|r_0 - z_k - A\mathcal{P}_{j,k}(A)r_k^{GR}\|_2 = \|r_k^{GR} - A\mathcal{P}_{j,k}(A)r_k^{GR}\|_2 < tol$$

■

The quantity $\|r_k^{GR} - A\mathcal{P}_{j,k}(A)r_k^{GR}\|_2$ is equal to the norm of the jth residual in the kth GMRES inner iteration, and this norm can be computed with little additional costs in GMRES (see [13]).

It follows from Lemma 4.3 and Theorem 4.1 that if GMRESR does not break down, then the sequence $\{\|r_i^{GR}\|_2\}$ is monotonically decreasing.

The reduction that one may expect at least for a given matrix A is given by α_m:

$$\alpha_m = \sup_{r_0 \in \mathbb{R}^n} \frac{\|r_m^G\|_2}{\|r_0\|_2}$$

(note that the optimality property of GMRES gives $\alpha_m \in [0, 1]$).

In the following lemma we compare the convergence behavior of GMRES(m) and GMRESR($\star$, $\star$, 1, m).

Lemma 4.4. *If* $\alpha_m < 1$ *then*

$$\frac{\|r_{k+1}^{GR}\|_2}{\|r_k^{GR}\|_2} \leq \alpha_m$$

Proof We apply m iteration steps of GMRES to $Au_{k,\star} = r_k^{GR}$ with $u_{k,0} = 0$. After this the residual is equal to $r_k^{GR} - A\mathcal{P}_{m,k}(A)r_k^{GR}$.

Using the definition of α_m it follows that $\|r_k^{GR} - A\mathcal{P}_{m,k}(A)r_k^{GR}\|_2 \leq \alpha_m \|r_k^{GR}\|_2$. Hence, since $\alpha_m < 1$ it follows from the proof for Theorem 4.1 that GMRESR(m) does not switch to LSQR (and does not break down). From Lemma 4.3 it then follows that $\|r_{k+1}^{GR}\|_2 \leq \alpha_m \|r_k^{GR}\|_2$.

■

Corollary 4.1. *If GMRES does not stagnate in m iteration steps(which means $\alpha_m < 1$) for a given matrix A then GMRESR($\star$, $\star$, 1, m) does not switch to LSQR and it converges at least as fast as GMRES(m).*

In [20] it is shown that the original method of Eirola and Nevanlinna [6] is not scaling invariant. Since the idea of GMRESR originates from that method, we investigate the convergence behavior of GMRESR with respect to scaling.

Definition 4.1. *The quantities associated with GMRESR, when applied to $\rho Ax = \rho b$ with $\rho > 0$ are denoted by the accent* $\hat{}$, *e.g.,* $\hat{A} = \rho A$. $\hat{b} = \rho b$, *etc.*

Lemma 4.5. *GMRESR is scaling invariant:*

$$\hat{x}_k^{GR} = x_k^{GR}$$

Proof We prove the lemma by an induction argument in k. The induction hypothesis is:

$$\hat{x}_k^{GR} = x_k^{GR},\ \hat{r}_k^{GR} = \rho r_k^{GR},\ \hat{u}_k = \frac{1}{\rho}u_k,\ \text{ and } \hat{c}_k = c_k$$

For $k = 0$ we have $\hat{x}_0 = x_0$ and $\hat{r}_0^{GR} = \rho r_0$. It is easy to show that $\hat{u}_0^{(0)} = \mathcal{P}_{m,0}(\hat{A})\hat{r}_0 = u_0^{(0)}$ thus $\hat{c}_0^{(0)} = \hat{A}\hat{u}_0^{(0)} = \rho c_0^{(0)}$. This implies that $\hat{u}_0 = \frac{1}{\rho}u_0$ and $\hat{c}_0 = c_0$. Since $\hat{x}_1^{GR} = \hat{x}_0 + \hat{u}_0\hat{c}_0^T\hat{r}_0$ we obtain $\hat{x}_1^{GR} = x_0 + \frac{1}{\rho}u_0c_0^T\rho r_0 = x_0^{GR}$ and $\hat{r}_1^{GR} = \rho r_1^{GR}$ (scaling invariance in case of an LSQR-switch is easily verified).

By similar arguments it follows that $\hat{u}_k^{(0)} = u_k^{(0)}$ and $\hat{c}_k^{(0)} = \rho c_k^{(0)}$. From the GMRESR scheme and the induction hypothesis it follows that $\hat{u}_k = \frac{1}{\rho}u_k$. $\hat{c}_k = c_k$. $\hat{x}_{k+1}^{GR} = x_{k+1}^{GR}$, and $\hat{r}_{k+1}^{GR} = \rho r_{k+1}^{GR}$. ■

For the invariance property we have used the fact that the inner iterations are started with $u_{k,0}^{(0)} = 0$, which is necessary in order to avoid shifts in the Krylov subspaces. Other starts do not necessarily ensure the scaling invariance of the process.

5. Implementation details

In this section we use results from sections 3 and 4, and [13] to obtain a cheaper implementation for GMRES in the inner iteration. Then we compare the amount of work and required memory for full GMRES and GMRESR($\star$, $\star$, 1, m) (this will be referred to as GMRESR(m)). Furthermore we will derive expressions for m that lead to optimal choices with respect to work and memory requirements. We conclude this section with some indications for situations when GMRESR may be preferred over GMRES. In this section we will assume that the LSQR-switch has not been activated. This facilitates the performance analysis.

5.1. *The inner iteration process*

In the GMRESR scheme we do the inner iteration by GMRES for the calculation of $u_k^{(0)} = \mathcal{P}_{m,k}(A)r_k^{GR}$. Since the inner iteration has some special properties we modify GMRES slightly in order to obtain a cheaper variant.

First of all we note that the inner iteration starts with $u_{k,0}^{(0)} = 0$, which implies that the initial residual $r_k^{GR} - Au_{k,0}^{(0)}$ is equal to r_k^{GR}. So the matrix vector product to calculate $Au_{k,0}^{(0)}$ is not necessary in the inner iteration. Second, it follows from Lemma 4.3 that we can stop

Table 2. Amount of work and memory for GMRES and GMRESR(m)

Method	GMRES	GMRESR(m)
steps	m_g	m_{gr}
matvec	m_g	$m_{gr} \cdot m$
vector updates	$\frac{1}{2}m_g^2$	$m_{gr} \cdot (\frac{m^2}{2} + m_{gr})$
inner products	$\frac{1}{2}m_g^2$	$m_{gr} \cdot (\frac{m^2}{2} + \frac{m_{gr}}{2})$
memory vectors	m_g	$2m_{gr} + m$

the inner iteration if the residual is less than tol. Finally, in [13]: p. 863, it is shown that the residual can be calculated with $m + 1$ vector updates instead of using a matrix vector product as in the expression $r_m = b - Ax_m$. In most applications m will be small, e.g. $m < 10$, which implies that $m + 1$ vector updates cost much less than a matrix vector product, so we use a similar idea to calculate $c_k^{(0)} = Au_k^{(0)}$.

We denote the Krylov subspace basisvectors, generated by GMRES, by v_j; V_m is the matrix with columns $v_1, \ldots, v_m$ and $\bar{H}_m$ is the $m+1$ by m upper Hessenberg matrix generated by GMRES (e.g., see [13]). Then we have $u_k^{(0)} = V_m y_m$, and hence $c_k^{(0)} = Au_k^{(0)} = AV_m y_m$. Since $AV_m = V_{m+1}\bar{H}_m$ it follows that $c_k^{(0)} = V_{m+1}\bar{H}_m y_m$. With these modifications we obtain the following algorithm:

Algorithm for the computation of $u_k^{(0)}$ and $c_k^{(0)}$

```
1.  Start:      Take $tol$ as in the outer iteration,
                $r_0 = r_k^{GR}$ and $v_1 = r_0/\|r_0\|_2$,
2.  Iterate: for $j = 1, \ldots, i$ (where $i$ is such that
                $i = m$ or $\|r_i\|_2 < tol$)
                do
                    $v_{j+1} = Av_j$;
                    for $t = 1, \ldots, j$ do
                        $h_{tj} = v_{j+1}^T v_t$; $v_{j+1} = v_{j+1} - h_{tj} v_t$;
                    $h_{j+1,j} = \|v_{j+1}\|_2$; $v_{j+1} = v_{j+1}/h_{j+1,j}$;
3.  $u_k^{(0)} = V_i y_i$
    $c_k^{(0)} = V_{i+1}\bar{H}_i y_i$, where $y_i$
                minimizes $\|\beta e_1 - \bar{H}_i y_i\|_2$
                with $\beta = \|r_k^{GR}\|_2$ and $e_1, y_i \in \mathbb{R}^i$.
```

5.2. *The choice of m*

In order to compare the efficiency of GMRES and GMRESR(m), estimates for the amount of work and the required memory of both methods are listed in Table 2. From these estimates we derive optimal choices for m with respect to work and required memory. To that end we assume that $m_{gr} \cdot m \cong m_g$. We have already seen in section 4 that $m_{gr} \cdot m \geq m_g$. In situations where $m_{gr} \cdot m$ is considerably larger than m_g we expect that GMRESR is far less efficient with respect to CPU time and memory than GMRES. Hence, it is only attractive to use GMRESR(m) when $m_{gr} \cdot m$ is not too far from m_g.

If $m_{gr} \cdot m \cong m_g$ then the number of required matrix vector products is about the same for both methods; however the numbers of vector updates and inner products can be different. Assuming that a vector update costs as much as an inner product, the amount of work w (in suitable units), with respect to vector updates and inner products, is given by:

GMRES: $w_g(m_g) = m_g^2$,
GMRESR(m): $w_{gr}(m_{gr}, m) = 1.5m_{gr}^2 + m_{gr} \cdot m^2$.

Using $m_{gr} = m_g/m$, the amount of work w_{gr} as a function of m is given by

$$w_{gr}(m) = \frac{1.5m_g^2}{m^2} + m_g \cdot m$$

The minimum is attained for $m = \sqrt[3]{3m_g}$ and is equal to $\frac{4.5}{\sqrt[3]{9}} \cdot m_g^{4/3}$. Note that if m_g grows the amount of work in GMRES increases as m_g^2, whereas the increase of work in GMRESR(m) is equal to 2.5 $m_g^{4/3}$ which has a much smaller increase than m_g^2. With respect to the optimal value of $m = \sqrt[3]{3m_g}$ we remark that it is a slow varying function of m_g. Thus a given m is near-optimal for a wide range of values of m_g. For numerical experiments with this choice of m we refer to section 6.

In order to optimize m with respect to memory requirements, we denote the amount of memory by:

GMRES: $mem(m_g) = m_g$
GMRESR(m): $mem(m_{gr}, m) = 2m_{gr} + m$

Assuming again that $m_{gr} = m_g/m$ we obtain $mem(m_{gr}, m) = \frac{2m_g}{m} + m$. The optimal value of m in this case is equal to $m = \sqrt{2m_g}$, which implies that the amount of memory is equal to $2\sqrt{2m_g}$. So the increase in required memory in GMRESR(m) as function of m_g is much less than for GMRES.

Note that the optimal m with respect to work is in general less than the optimal m with respect to memory. It depends on the problem and the available computer, which value is preferred. However, in our experiments we observe that for both choices the amount of work and required memory is much less than for GMRES.

In order to obtain an optimal choice of m it is necessary to estimate m_g. If the system of equations is solved once the only possibility is to get an upper-bound of m_g from an analysis of the problem. If the system of equations is solved many times, e.g. a time dependent problem, a nonlinear problem or many right-hand side vectors, then there are other possibilities:

— the first time the system is solved with full GMRES. We then assume that the value of m_g does not change much in other problems. Since many systems are solved, the extra costs of full GMRES are negligible.
— the first time the system is solved with, e.g., GMRESR(5) and we use $5 \cdot m_{gr}$ as an approximation for m_g in the remaining systems.

In this paragraph we give three conditions, under which GMRESR(m) is more efficient than GMRES or GMRES(m).

Table 3. The results for $\beta = 1$

Method	Iterations	matvec	CPU time
GMRES(32)	1355	1355	69.0
CGS	288	576	5.8
Bi-CGSTAB	252	504	4.7
GMRESR(10)	36	360	12.0

— m_g is relatively large, because if m_g is small, e.g. less than 20, then the gain obtained from GMRESR(m) is negligible,
— $m_{gr} \cdot m$ is approximately equal to m_g,
— full GMRES has a superlinear convergence behavior, which implies that GMRES (m) shows slow convergence for $m \ll m_g$.

For a class of problems where these conditions holds we refer to [19].

Finally it is possible to use other iterative methods in the inner iteration. Possibilities are: GMRES(m) where m may be different for every step of the outer iteration, or GMRESR(m) itself, other implementations of GMRES (see [3] and [5]) or Bi-CG-methods: CGS [15], Bi-CGSTAB [17]. Note that, by following the approaches suggested in [4] and [5], the inner iteration process is well suited for parallel computation.

6. Numerical experiments

In this section GMRESR(m) is tested and compared with other iterative methods. In our problems the LSQR switch was never activated, except for the very last example (which was designed to obtain that effect). We start with an artificial problem: a convection diffusion equation on a unit square. It appears that GMRESR(10) is a robust method for this class of problems. Thereafter we specify some results for a practical problem, obtained from a discretization of the Navier–Stokes equations. In these experiments we see the theoretical properties of GMRESR(m), as discussed in sections 4 and 5, confirmed.

We describe some numerical experiments with a linear system obtained from a discretization of the following PDE

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\beta\left(\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}\right)=f \text{ on } \Omega$$

$$u|_{\partial\Omega}=0$$

where Ω is the unit square.

The exact solution u is given by $u(x,y) = \sin(\pi x)\sin(\pi y)$. In the discretization we use the standard five-point central finite difference approximation. The step sizes in x- and y-direction are equal to 1/100. We use the following iterative methods: GMRES(m), CGS, Bi-CGSTAB and GMRESR(m). We use a more or less optimal choice of m to obtain results using GMRES(m). We start with $x_0 = 0$ and stop if $\|r_k\|_2/\|r_0\|_2 \leq 10^{-12}$, see Tables 3–5. Using $\beta = 100$ the updated residual of Bi-CGSTAB satisfies $\|r_{210}\|_2 \leq 10^{-13}$ whereas the norm of the exact residual is equal to 10^{-9}.

Note that in these examples GMRESR(10) is a robust method because it converges for all

Table 4. The results for $\beta = 100$

Method	Iterations	matvec	CPU time
GMRES(4)	256	256	4.8
CGS	n.c.		
Bi-CGSTAB	210	420	4.6
GMRESR(10)	35	350	11.0

Table 5. The results for $\beta = 500$

Method	Iterations	matvec	CPU time
GMRES(4)	302	302	5.6
CGS	n.c.		
Bi-CGSTAB	n.c.		
GMRESR(10)	36	360	13.0

our choices of β. CGS and Bi-CGSTAB fail for β large, whereas for β small the restarted version of GMRES has a slow convergence behavior. Contrary to GMRESR, where $m = 10$ is a good choice for a wide range of β, the optimal value of m, used in GMRES(m), changes considerable for different values of β.

Finally we take β as a function of x and y as follows (and see Table 6):

$$\beta(x, y) = \begin{cases} 1 & \text{for } x, y \in [\frac{1}{2}, \frac{3}{5}]^2 \\ 1000 & \text{for } x, y \in [0, 1]^2 \setminus [\frac{1}{2}, \frac{3}{5}]^2 \end{cases} \tag{6.1}$$

For CGS the updated residual is such that $\|r_{583}\|_2 \leq 10^{-10}$ whereas the norm of the exact residual is larger than 10^{-4}; we consider this as a case of nonconvergence. Note that in this problem GMRESR(10) is the best method.

The following examples come from a discretization of the incompressible Navier–Stokes equations. This discretization leads to two different linear systems, the momentum equations and the pressure equation (for a further description we refer to [19]). Here we consider a specific test problem, which describes the flow through a curved channel.

In the first example the problem is discretized with 16×64 finite volumes. The pressure equations are solved with GMRES(m) and GMRESR(m). We start with $x_0 = 0$ and stop when $\|r_k\|_2/\|r_0\|_2 \leq 10^{-6}$. This is essentially the same problem as the one for which results

Table 6. The results, where β is given in (6.1)

Method	Iterations	matvec	CPU time
GMRES(32)	1418	1418	70
CGS	n.c.		
Bi-CGSTAB	n.c.		
GMRESR(10)	56	560	19

Table 7. GMRESR(m) applied to the pressure equations

m	4	8	12	16	18	20	22
iterations	45	23	16	12	11	10	9
CPU time	1.15	0.89	0.90	1.05	1.09	1.23	1.30
memory vectors	94	54	44	40	40	40	40

Table 8. Iterative methods applied to the momentum equations

Method	Iterations	CPU time	Memory vectors
full GMRES	31	0.61	31
GMRESR(5)	7	0.35	19
GMRESR(8)	4	0.37	16

were reported in section 3, only the discretization is slightly different.

Full GMRES converges in 177 iterations and used 5.8 s CPU time. Restarting GMRES is a bad idea, e.g. GMRES(50) takes 1323 iterates and 16 s CPU time to converge. In Table 7 we specify the results for GMRESR(m), where m is chosen near the optimal value with respect to work ($m = 8$) and with respect to memory ($m = 20$). Note that in this example $m \cdot m_{gr}$ is approximately equal to m_g for $4 \leq m \leq 22$.

We observe a good correspondence between the predicted and the real optimal values of m. Note that the optimal value of m with respect to memory is larger than that with respect to work.

For this problem we also solve the momentum equations (see Table 8) with full GMRES and GMRESR(m). The choices for m are $m = 5$, optimal with respect to work, and $m = 8$, optimal with respect to memory. For both choices of m we observe a considerable gain in computing time and memory requirements.

We have solved the pressure equations with a combination of GMRESR(m) with a (M)ILU preconditioner (see [10], [16] and [8]).

In Table 9 we show results, using an average of an ILU and a MILU preconditioner with $\alpha = 0.975$ [1], [19]: p.8. For large problems (32×128) GMRESR(m) is much better than full GMRES. Since GMRES(m) converges very slowly for these examples we have not included results for GMRES(m) in Table 9.

Our final example has been included in order to demonstrate the effect of the 'LSQR switch'. The matrix A has as its columns $e_2, e_3, \ldots, e_{10000}, e_1$, where e_i is the ith canonical basis vector in $\mathbb{R}^{10000}$. For the right-hand side we take $b = e_1$. It is well-known that, with the start $x_0 = 0$, GMRES produces the iterands $x_1 = \ldots = x_{9999} = 0$ and $x_{10000} = e_{10000}$. In this case we may expect stagnation in the inner iterations for any reasonable choice of m. With the LSQR switch, however, GMRESR converges in only one iteration, owing to the fact that LSQR converges in one iteration for this specific case.

In order to make the situation less trivial, we select a different right-hand side. The vector b is chosen such that it leads to the solution x with

$$x_{(i-1)\cdot 100+j} = \sin(\pi i/100)\sin(\pi j/100)$$

Table 9. Iterative method applied to the pressure equation using a MILU preconditioner

Method	Finite volume	Iterations	CPU time	Memory vectors
full GMRES	16×64	28	0.31	28
GMRESR(4)	16×64	9	0.27	22
full GMRES	32×128	47	2.19	47
GMRESR(5)	32×128	10	1.21	25

Table 10. Results for relaxed LSQR switch

s	Iterations
0.9	2
$1 - 10^{-7}$	4
$1 - 10^{-8}$	> 100

$i, j = 1, \dots, 100$.

Furthermore, we change the switch criterion a little bit. Instead of switching only when $\|Au_{k,m}^{(0)} - r_k\|_2 = \|r_k\|_2$, which seems not quite practical in actual computing, we switch when $\|Au_{k,m}^{(0)} - r_k\|_2 \geq s\|r_k\|_2$, for some suitable s close to 1.

GMRESR(10) is started with x_0, as above, and for different values of s we have listed the number of GMRESR iteration steps in Table 10.

This experiment indicates that it might be better to take s in practice slightly less than 1.

7. Conclusions

We propose a class of new iterative methods, GMRESR(k, j, $n(\varepsilon)$, m), for the iterative solution of a linear system $Ax = b$ with unsymmetric nonsingular matrix A. These methods are shown to be robust when an LSQR-switch is included.

It appears that the increase of vector updates, inner products and required memory, as function of the amount of iterations, is much less than the increase of this quantities using full GMRES. From our numerical experiments we conclude that GMRESR($\star, \star, 1, m$), even without activating the LSQR switch, is a robust method.

Optimal choices for the parameter m are easily obtained and do not change very much for different problems. In most experiments we observe for GMRESR($\star, \star, 1, m$) a considerable improvement, in computing time and memory requirements, in comparison with more familiar GMRES variants.

Though we have only analysed one specific GMRESR variant, it is clear from our presentation that there is an overwhelming freedom in variants. Some of these are currently investigated and will be reported separately.

REFERENCES

1. O. Axelsson and G. Lindskog. *On the eigenvalue distribution of a class of preconditioning*

methods. Numer. Math., 48, 479–498, 1986.

2. 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.
3. Z. Bai, D. Hu, and L. Reichel. *A Newton basis GMRES implementation.* Tech. Report 91-03, University of Kentucky, 1991.
4. A. T. Chronopoulos and S. K. Kim. *s-Step Orthomin and GMRES implemented on parallel computers.* Tech. Report 90/43R, UMSI, Minneapolis, 1990.
5. E. de Sturler. *A parallel variant of GMRES(m).* In *Proc. of the 13th IMACS World Congress on Computation and Applied Math.*, J. Miller and R. Vichnevetsky, editors, pages 682–683. Criterion Press, Dublin, 1991.
6. T. Eirola and O. Nevanlinna. *Accelerating with rank-one updates. Lin. Alg. and its Appl.*, 121, 511–520, 1989.
7. 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.
8. I. Gustafsson. *A class of first order factorization methods. BIT*, 18, 142–156, 1978.
9. Y. Huang and H. A. van der Vorst. *Some observations on the convergence behavior of GMRES.* Tech. Report 89-09, Delft University of Technology, Faculty of Tech. Math., 1989.
10. J. A. Meijerink and H. A. van der Vorst. *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math.Comp.*, 31, 148–162, 1977.
11. N. M. Nachtigal, L. Reichel and L. N. Trefethen. *A hybrid GMRES algorithm for nonsymmetric matrix iterations. SIAM J. Matrix Anal. Appl.*, 13, 796–825, 1992.
12. C. C. Paige and M. A. Saunders. *LSQR: An algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Soft.*, 8, 43–71, 1982.
13. Y. Saad and M. H. Schultz. *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Statist. Comput.*, 7, 856–869, 1986.
14. Y. Saad. *A flexible Inner-Outer preconditioned GMRES algorithm. SIAM J. Sci. Stat. Comp.*, 14, 461–469, 1993.
15. P. Sonneveld. *CGS: a fast Lanczos-type solver for nonsymmetric linear systems. SIAM J. Sci. Statist. Comput.*, 10, 36–52, 1989.
16. H. A. van der Vorst. *High performance preconditioning. SIAM J. Sci. Statist. Comput.*, 10, 1174–1185, 1989.
17. H. A. van der Vorst. *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. SIAM J. Sci. Statist. Comput.*, 13, 631-644, 1992.
18. H. A. van der Vorst and C. Vuik. *The superlinear convergence behaviour of GMRES. J. Comput. Appl. Math.*, 48, 327-341, 1993.
19. C. Vuik. *Solution of the discretized incompressible Navier–Stokes equations with the GMRES method. Int. J. for Num. Meth. in Fluids*, 16, 507–523, 1993.
20. C. Vuik and H. A. van der Vorst. *A comparison of some GMRES-like methods. Lin. Alg. and its Appl.*, 160, 131-162, 1992.