CMRH: A new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm

H. Sadok

Université du Littoral, zone universitaire de la Mi-voix, Bâtiment H. Poincaré, 50 rue F. Buisson, BP 699, F-62228 Calais Cedex, France
E-mail: sadok@lma.univ-littoral.fr

Received 30 June 1998; revised 21 January 1999
Communicated by G. Meurant

The Generalized Minimal Residual (GMRES) method and the Quasi-Minimal Residual (QMR) method are two Krylov methods for solving linear systems. The main difference between these methods is the generation of the basis vectors for the Krylov subspace. The GMRES method uses the Arnoldi process while QMR uses the Lanczos algorithm for constructing a basis of the Krylov subspace.

In this paper we give a new method similar to QMR but based on the Hessenberg process instead of the Lanczos process. We call the new method the CMRH method. The CMRH method is less expensive and requires slightly less storage than GMRES. Numerical experiments suggest that it has behaviour similar to GMRES.

Keywords: linear systems, iterative methods, Hessenberg's method, GMRES, QMR

AMS subject classification: 65F10

1. Introduction and notations

Many large linear systems, in particular those arising from discretization by finite differences or by finite elements, can be solved efficiently by iterative methods, especially those based on a Krylov subspace; for a review of such methods see, for example, [11]. Among the most widely used method is the Generalized Minimum Residual algorithm (GMRES) due to Saad and Schultz [25], whose work and storage requirements grow linearly with iterations. In general the restarted algorithm GMRES(m) is used. Another competitive method is the Quasi-Minimal Residual method (QMR) which has low storage (in general) and constant work per iteration. It was introduced by Freund [10] for the case of complex symmetric linear systems and extended by Freund and Nachtigal [13] for the case of nonsymmetric matrices. These two methods are based on the reduction of a Krylov matrix [32, chapter 6] to have a Krylov basis and an upper Hessenberg matrix. GMRES uses the Arnoldi process while QMR uses the Lanczos algorithm. When using the Lanczos algorithm we may have a possible breakdown or

a near-breakdown and one must use a look-ahead Lanczos process for avoiding such breakdowns [5,6,12,17,21,23].

Another possibility for constructing a basis for a Krylov subspace and an upper Hessenberg matrix is the Hessenberg method; see [18; 32, p. 379]. This method requires less work and storage than Arnoldi's method since it uses, at iteration k, the basis vectors $\{l_1, \ldots, l_k\}$ such that l_i has $i-1$ components equal to zero and one component equal to one. Moreover, as in Arnoldi's method, we have to compute a matrix by vector multiplication Al_i which has a lower cost ($n(n-i)$ for a dense matrix) and hence is less expensive than a matrix by vector multiplication (n^2) as in the Arnoldi process and two matrix by vector multiplications for the Lanczos method.

By replacing in QMR the Lanczos algorithm by the Hessenberg process we obtain a new method, which minimizes the residual in a norm that changes with the iterations. This new method will be called CMRH (Changing Minimal Residual method based on the Hessenberg process).

The outline of the paper is as follows. In section 2, we present the Hessenberg method and we briefly recall its properties and its implementation. The new method is presented and its theoretical properties are analyzed in section 3. Finally we present some numerical experiments.

Throughout the paper, all vectors and matrices are assumed to be real. We define the Krylov subspace by

$$K_k(v, A) = \text{span}\{v, Av, \ldots, A^{k-1}v\}$$

for any given vector $v \in \mathbb{R}^n$ and $n \times n$ matrix A. Furthermore, for a vector v, $\|v\|$ always denotes the Euclidean norm $\|v\| = \sqrt{(v^T v)}$ and $\|v\|_\infty$ denotes the maximum norm $\|v\|_\infty = \max_{i=1,\ldots,n} |(v)_i|$, where $(v)_i$ is the ith component of the vector v. For a matrix A, $\|A\|$ denotes the 2-norm and $\chi(A)$ denotes the 2-condition number of A.

Moreover, we denote by $I_k^{(n)}$ the $n \times k$ matrix whose columns are $e_k^{(n)}$, the kth canonical vector of $\mathbb{R}^n$:

$$e_k^{(n)} = (0, \ldots, 0, 1, 0, \ldots, 0)^{\mathrm{T}}.$$

We have $I_k^{(k)} = I_k$, the identity matrix.

2. The Hessenberg method

We shall now describe an algorithm normally only used for computing explicitly the characteristic polynomial. This method is due to Hessenberg [18].

Let v be a given vector of $\mathbb{R}^n$ and A an $n \times n$ matrix. The Hessenberg process consists in constructing a basis $\{l_1, l_2, \ldots, l_k\}$ of the subspace $K_k(v, A)$. The $n \times k$ matrix L_k whose columns are $l_1, \ldots, l_k$, is a trapezoidal matrix. Before computing the vectors l_k we need some definitions.

Let Z_k be the $n \times k$ matrix whose columns are $z_1, \ldots, z_k$. We partition this matrix as follows:

$$Z_k = \begin{pmatrix} Z1_k \\ Z2_k \end{pmatrix},$$

where $Z1_k$ is a $k \times k$ square matrix.

If the matrix $Z1_k$ is nonsingular we define Z_k^L a left inverse of Z_k by

$$Z_k^L = \left(I_k^{(n)\mathrm{T}} Z_k\right)^{-1} I_k^{(n)\mathrm{T}} = \left(Z1_k^{-1}, 0\right). \tag{2.1}$$

Now let V_k be the Krylov matrix whose columns are $v_1, \ldots, v_k$ with $v_1 = v$ and $v_i = A^{i-1}v$, for $i = 2, \ldots, k$.

Since V_k is a rectangular $n \times k$ matrix, its LU factorization [15, p. 102] when it exists, can be written as $V_k = L_k U_k$, where L_k is a trapezoidal (triangular when $k = n$) $n \times k$ matrix and U_k is a unit upper triangular $k \times k$ matrix.

Let us write L_k and V_k in the partitioned form

$$L_k = \begin{pmatrix} L1_k \\ L2_k \end{pmatrix} \quad \text{and} \quad V_k = \begin{pmatrix} V1_k \\ V2_k \end{pmatrix}.$$

If $\det(V1_j) \neq 0$ for $j = 1, \ldots, k$, then the LU factorization exists and $L1_k$ is a lower triangular matrix with $\det(L1_k) = \det(V1_k)$, $V1_k = L1_k U_k$ and $V2_k = L2_k U_k$.

We have the following fundamental result:

Theorem 1. Let V_k be a Krylov matrix such that $\det(V1_j) \neq 0$ for $j = 1, \ldots, k$. Then

1. $\mathrm{span}\{l_1, \ldots, l_k\} = K_k(v, A)$.
2. The vector l_k can be written as

$$l_k = v_k - V_{k-1} V_{k-1}^L v_k = Al_{k-1} - L_{k-1} L_{k-1}^L Al_{k-1}, \quad \forall k > 1 \quad \text{and} \quad l_1 = v.$$

3. $AL_{k-1} = L_k H_{k-1}$, where H_{k-1} is an upper Hessenberg $k \times (k-1)$ matrix such that

$$H_{k-1} = \begin{pmatrix} h_1 & \ldots & h_{k-1} \\ 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \quad \text{and} \quad h_i = L_i^L Al_i \in \mathbb{R}^i, \text{ for } i = 1, \ldots, k-1.$$

Proof. 1. Obvious.

2. The elements of the matrix L_k can be written as a determinantal expression [19, formula 5, p. 11], and by using the generalization of Schur's formula given in [4], we obtain the kth column l_k of the matrix L_k under the compact form

$$l_k = v_k - V_{k-1} V_{k-1}^L v_k.$$

The last relation gives $v_k = l_k + V_{k-1}d_k$ with $d_k = V_{k-1}^L v_k$, and since $v_k = Av_{k-1}$ we have $v_k = Al_{k-1} + AV_{k-2}d_{k-1} = Al_{k-1} + V_{k-1}g_k$ with the first component of g_k equal to zero. By using the fact that $V_k V_k^L V_k = V_k$, we obtain

$$\begin{aligned} l_k &= v_k - V_{k-1}V_{k-1}^L v_k \\ &= Al_{k-1} + V_{k-1}g_k - V_{k-1}V_{k-1}^L(Al_{k-1} + V_{k-1}g_k) \\ &= Al_{k-1} - V_{k-1}V_{k-1}^L Al_{k-1}. \end{aligned}$$

Now, by definition, $V_k^L = (V1_k^{-1}, 0)$ and since $V1_k = L1_k U_k$ it follows that

$$V_k V_k^L = L_k U_k \big(V1_k{}^{-1}, 0\big) = L_k U_k \big(U_k^{-1} L1_k{}^{-1}, 0\big) = L_k \big(L1_k{}^{-1}, 0\big) = L_k L_k^L.$$

Hence, $l_k = Al_{k-1} - L_{k-1}L_{k-1}^L Al_{k-1}$.

3. We proceed by induction. For $k = 2$ we have

$$l_2 = Al_1 - h_1 l_1 \quad \text{with } h_1 = \frac{(Al_1)_1}{(l_1)_1}.$$

Hence,

$$Al_1 = h_1 l_1 + l_2 = (l_1, l_2)\begin{pmatrix} h_1 \\ 1 \end{pmatrix} = L_2 H_1.$$

Assuming that the relation holds for $k-1$, let us prove it for k. We have $L_{k-1} = (L_{k-2}, l_{k-1})$, then $AL_{k-1} = (AL_{k-2}, Al_{k-1})$. Now, since $Al_{k-1} = l_k + L_{k-1}h_{k-1}$, where

$$h_{k-1} = L_{k-1}^L Al_{k-1},$$

we get

$$\begin{aligned} AL_{k-1} &= (AL_{k-2},\, l_k + L_{k-1}h_{k-1}) = (L_{k-1}H_{k-2},\, l_k + L_{k-1}h_{k-1}) \\ &= (L_{k-1},\, l_k)\begin{pmatrix} H_{k-2} & h_{k-1} \\ 0 & 1 \end{pmatrix} = L_k H_{k-1}, \end{aligned}$$

which ends the proof. □

By using theorem 1, we are now able to describe the Hessenberg process.

Algorithm 1 (Hessenberg process).

$l_1 = v$
for $k = 1, \ldots, n$
  $u = Al_k$
  for $j = 1, \ldots, k$
    $\beta_j = u_j/(l_j)_j;\ h_{j,k} = \beta_j;\ u_j = 0$
    for $l = j+1, \ldots, n$
      $u_l = u_l - \beta_j * (l_j)_l$

```
    end
  end
  h_{k+1,k} = 1; l_{k+1} = u
end.
```

The Hessenberg process can break down if $(l_j)_j$ is zero. We can avoid such a breakdown and also ensure numerical stability if we use a pivoting strategy such as in the Gaussian elimination method (see, for example, [15]). This is equivalent to replacing the left inverse of the matrix Z_k by

$$Z_k^L = \left(Y_k^{\mathrm{T}} Z_k\right)^{-1} Y_k^{\mathrm{T}}, \tag{2.2}$$

where Y_k is the $n \times k$ matrix whose columns are $y_1, \ldots, y_k$, with $y_k = e_{p(k)}^{(n)}$ and $p(k) \in \{1, \ldots, n\}$.

Now we shall see how to determine the $p(k)$. We follow the practical procedure given in [32, p. 390].

Suppose that $p(1), \ldots, p(k)$ have already been obtained and that we want to have $p(k+1)$. First, we compute $u = Al_k$ and subtract multiples of $l_1, \ldots, l_k$ in order to annihilate the k components $p(1), \ldots, p(k)$ of the vector u, to obtain a vector w. Now if $\|w\|_\infty = |(w)_{i0}|$, then set $p(k+1) = i0$ and define the vector l_{k+1} as

$$l_{k+1} = \frac{w}{(w)_{i0}}.$$

With this procedure the matrix L_k is not trapezoidal, but there exists a permutation matrix such that PL_k is unit trapezoidal. It is important to note that the vector l_{k+1} is normalized ($\|l_{k+1}\|_\infty = 1$), has k zero components and one component equal to one. We note that algorithm 1 produces a unit upper Hessenberg H_k and a non-unit lower trapezoidal L_k. Since now the vector l_{k+1} is normalized ($\|l_{k+1}\|_\infty = 1$) for avoiding overflow, the following algorithm produces a non-unit upper Hessenberg and a unit lower trapezoidal PL_k.

The Hessenberg process with pivoting is as follows:

Algorithm 2 (Hessenberg process with pivoting).

```
Set p = (1, ..., n)^T.
Determine i_0 so |(v)_{i_0}| = ||v||_∞;  l_1 = v/(v)_{i_0};  p(1) ↔ p(i_0)
for k = 1, ..., n
  u = Al_k
  for j = 1, ..., k
    c = (u)_{p(j)};  h_{j,k} = c;  (u)_{p(j)} = 0
    for l = j + 1, ..., n
      (u)_{p(l)} = (u)_{p(l)} − c * (l_j)_{p(l)}
    end
  end
```

If $k < n$
determine i_0 so $|(u)_{i_0}| = \|u\|_\infty$; $p(k+1) \leftrightarrow p(i_0)$
$h_{k+1,k} = (u)_{i_0}$; $l_{k+1} = u/(u)_{i_0}$
end
end.

The "$\leftrightarrow$" notations means as in [15] "swap contents":

$$\alpha \leftrightarrow \beta \quad \Longleftrightarrow \quad \gamma = \alpha;\ \alpha = \beta;\ \beta = \gamma.$$

Remarks. 1. Note that if $\|u\|_\infty = 0$ at step k, then, in exact arithmetic, the minimal polynomial with respect to the vector v has degree k and the process must be stopped.

2. For iteration k, the algorithm involves $\sum_{j=1}^{k}(n-j) = nk - k(k+1)/2$ multiplications and a matrix by vector multiplication which requires nb_k multiplications, where $nb_k = n(n-k)$ for a dense matrix and $nb_k < NZ$, the number of nonzero elements in A, for a sparse matrix. The algorithm requires for m iterations about $nm^2/2 - m^3/6 + \sum_{k=1}^{m}(nb_k)$ multiplications. For comparison, the Arnoldi algorithm requires $2nk$ multiplications for iteration k and a matrix by vector multiplication which involves nb multiplications, where $nb = n^2$ for a general matrix and $nb = NZ$ for a sparse matrix. For m iterations, Arnoldi's algorithm requires about $nm^2 + mnb$ multiplications. Hence, the Hessenberg process is less expensive than Arnoldi's algorithm.

3. The zero elements of $(l_k)_{p(j)}$ could be overwritten by $h_{j,k}$, hence, the Hessenberg algorithm needs slightly less storage than Arnoldi's algorithm.

For algorithm 2, we have the following results.

Theorem 2. If the Krylov matrix V_k is of full rank then the vectors l_i, for $i = 1, \ldots, k$, generated by the Hessenberg process with pivoting are such that:

1. span $\{l_1, \ldots, l_k\} = K_k(v, A)$.
2. If we set $\tilde{l}_1 = v$ and $\tilde{l}_k = Al_{k-1} - L_{k-1}L_{k-1}^L Al_{k-1}$, where L_i^L is the left inverse of L_i defined by (2.2) with $p(1), \ldots, p(k-1)$ obtained by algorithm 2, then
$$l_k = \tilde{l}_k/(\tilde{l}_k)_{i_k}, \quad \text{where } \left|(\tilde{l}_k)_{i_k}\right| = \left\|\tilde{l}_k\right\|_\infty.$$
3. $AL_{k-1} = L_k H_{k-1}$, where H_{k-1} is an upper Hessenberg $k \times (k-1)$ matrix such that
$$H_{k-1} = \begin{pmatrix} h_1 & \ldots & h_{k-1} \\ (\tilde{l}_2)_{i_2} & & \\ & \ddots & \\ & & (\tilde{l}_k)_{i_k} \end{pmatrix} \quad \text{and} \quad h_i = L_i^L Al_i \in \mathbb{R}^i, \quad \text{for } i = 1, \ldots, k-1.$$

3. The CMRH method

3.1. A description of the GMRES and the QMR methods

In this section, we shall begin with a short description of the GMRES method. For a full description of this method and its standard implementation, see [25]. Other implementations are also possible, see [2,22,26,27,30,31]. For the convergence behaviour see [7,25,28,29].

Consider the linear system of equations

$$Ax = b, \tag{3.1}$$

where A is a nonsingular matrix of order n. Let x_0 be an initial approximation to x and $r_0 = b - Ax_0$. Consider the problem of finding x_k such that

$$x_k = x_0 + z_k, \tag{3.2}$$

where z_k is the solution of

$$\min_{z \in K_k(r_0, A)} \big\|b - A(x_0 + z)\big\|. \tag{3.3}$$

The iterates $\{x_k\}$ can be obtained by the Orthodir algorithm of Young and Jea [33], or by Axelsson's method [1] or by the Generalized Conjugate Residual method [8,9]. But the most efficient algorithm for generating such iterates is the GMRES method.

Saad and Schultz [25] solved the problem (3.2) by first computing an orthonormal basis $\{q_1, \ldots, q_{k+1}\}$ of $K_{k+1}(r_0, A)$ with $q_1 = r_0/\|r_0\|$ by the Arnoldi process, such that

$$AQ_k = Q_{k+1}\widetilde{H}_k,$$

where Q_k is an $n \times k$ matrix whose columns are $q_1, \ldots, q_k$ and $\widetilde{H}_k$ is a $(k+1) \times k$ upper Hessenberg matrix. Setting $z_k = Q_k y_k$, then y_k is the solution of

$$\min_{y \in \mathbb{R}^k} \big\|b - A(x_0 + Q_k y)\big\|. \tag{3.4}$$

On the other hand, we have

$$\min_{y \in \mathbb{R}^k} \big\|b - A(x_0 + Q_k y)\big\| = \min_{y \in \mathbb{R}^k} \|r_0 - AQ_k y\| = \min_{y \in \mathbb{R}^k} \big\|\|r_0\| e_1^{(k+1)} - \widetilde{H}_k y\big\|.$$

This least squares problem is solved by using the QR factorization of the matrix $\widetilde{H}_k$ with Givens rotations. Hence, (3.2) could be written as

$$x_k = x_0 + Q_k \widetilde{H}_k^{+} \|r_0\| e_1^{(k+1)}, \tag{3.5}$$

where Z^{+} denotes the pseudo-inverse of the matrix Z, see, for example, [15, p. 243].

Consider now the QMR method [13]. For a full decription and its implementation, see [13,14]. This method is based on the Lanczos process which constructs two bases

$\{u_1, \ldots, u_k\}$ and $\{\widetilde{u}_1, \ldots, \widetilde{u}_k\}$ which are in the regular case (without look-ahead) such that

1. span$\{u_1, \ldots, u_k\} = K_k(r_0, A)$ and span$\{\widetilde{u}_1, \ldots, \widetilde{u}_k\} = K_k(y, A^{\mathrm{T}})$, where y is a given vector.
2. If we set $U_k = (u_1, \ldots, u_k)$ and $\widetilde{U}_k = (\widetilde{u}_1, \ldots, \widetilde{u}_k)$, then $\widetilde{U}_k^{\mathrm{T}} U_k = D_k$ is a diagonal matrix.
3. $AU_k = U_{k+1} H_k^{(e)}$, where $H_k^{(e)}$ is a tridiagonal matrix.

The kth iterate of QMR is defined by

$$x_k = x_0 + U_k y_k,$$

where y_k is the solution of

$$\min_{y \in \mathbb{R}^k} \left\| \|r_0\| e_1^{(k+1)} - H_k^{(e)} y \right\|.$$

This iterate can also be written as $x_k = x_0 + z_k$, where z_k is the solution of the problem

$$\min_{w \in \mathbb{R}^{k+1}, z \in K_k(r_0, A)} \|w\|, \quad \text{subject to } Az = r_0 + U_{k+1} w. \tag{3.6}$$

We shall now propose a new method which is similar to QMR but uses another basis for the Krylov subspace. This basis is constructed by the Hessenberg process with pivoting as shown in the preceding section.

3.2. Description and implementation of the CMRH method

Replacing the matrix U_{k+1} in the problem (3.6) by the matrix $L_{k+1} = (l_1, \ldots, l_{k+1})$, where the vectors l_i for $i = 1, \ldots, k+1$ are obtained by algorithm 2, we get

$$\min_{w \in \mathbb{R}^{k+1}, z \in K_k(r_0, A)} \|w\|, \quad \text{subject to } Az = r_0 + L_{k+1} w. \tag{3.7}$$

This formulation is used to define the CMRH method in which the iterates x_k can be written as

$$x_k = x_0 + L_k H_k^{+} (r_0)_{i_0} e_1^{(k+1)}, \tag{3.8}$$

where i_0 is such that $|(r_0)_{i_0}| = \|r_0\|_\infty$.

Hence, iteration k in the CMRH method is based on one iteration of the Hessenberg process with pivoting (algorithm 2), the update factorization of the matrix H_k and only when (an estimate of) the residual $\|r_k\|$ is sufficiently small, a matrix by vector multiplication $z_k = L_k y_k$, where $y_k = (r_0)_{i_0} H_k^{+} e_1^{(k+1)}$. An estimation of the residual norms will be given in the sequel (formula (3.9)). We note that the matrix L_k has $(k(k-1))/2$ zero elements. We will now give the outline of the CMRH method.

Algorithm 3 (CMRH method).

Choose x_0, compute $r_0 = b - Ax_0$ and set $p(i) = i,$ for $i = 1, \ldots, n$.
Determine i_0 so $|(r_0)_{i_0}| = \|r_0\|_\infty;\ l_1 = r_0/(r_0)_{i_0};\ p(1) \leftrightarrow p(i_0)$
for $k = 1, \ldots,$ until satisfied do:
  $u = Al_k$
  for $j = 1, \ldots, k$
    $c = (u)_{p(j)};\ h_{j,k} = c;\ (u)_{p(j)} = 0$
    for $l = j + 1, \ldots, n$
      $(u)_{p(l)} = (u)_{p(l)} - c * (l_j)_{p(l)}$
    end
  end
  If $k < n$
    determine i_0 so $|(u)_{i_0}| = \|u\|_\infty;\ p(k + 1) \leftrightarrow p(i_0)$
    $h_{k+1,k} = (u)_{i_0};\ l_{k+1} = u/(u)_{i_0}$
  end.
  If (an estimate of) $\|b - Ax_k\|$ is small enough or $k = n$ then
    $x_k = x_0 + (l_1, \ldots, l_k) * y_k$, where y_k minimizes $\|H_k y - (r_0)_{i_0} e_1^{(k+1)}\|$,
    $y \in \mathbb{R}^k$.
    stop iteration
  end
end.

Remarks. 1. Computing $u = Al_k$ requires nb_k multiplications, where $nb_k = n(n - k)$ for a dense matrix and $nb_k < NZ$, the number of nonzero elements in A, for a sparse matrix. Furthermore, for each iteration k the computation of L_{k+1} by algorithm 2 requires $(k + 1)(2n - k)/2$ multiplications.

Now, if we neglect the cost of computing y_k as for the GMRES algorithm, then the CMRH method requires for step k about $k \times n + nb_k$ multiplications with $nb_k < NZ$, while the GMRES algorithm requires $2n \times k + NZ$. Hence the CMRH method is less expensive than the GMRES method.

2. The CMRH method uses a permutation matrix, hence it involves some movement for access to the data.

Although the new method is less expensive and needs slightly less storage than the GMRES method per iteration, the number of vectors requiring storage increases as k and hence the method must be restarted. The restarted version of the CMRH will be called CMRH(m) and can be described as follows.

Algorithm 4 (CMRH(m) method).

Choose m.
start: For x_0, compute $r_0 = b - Ax_0$ and set $p(i) = i,$ for $i = 1, \ldots, n$.

Determine i_0 so $|(r_0)_{i_0}| = \|r_0\|_\infty$; $l_1 = r_0/(r_0)_{i_0}$; $p(1) \leftrightarrow p(i_0)$
iterate: for $k = 1, \ldots, m$
  $u = Al_k$
  for $j = 1, \ldots, k$
    $c = (u)_{p(j)}$; $h_{j,k} = c$; $(u)_{p(j)} = 0$
    for $l = j + 1, \ldots, n$
      $(u)_{p(l)} = (u)_{p(l)} - c * (l_j)_{p(l)}$
    end
  end
  If $k < n$
    determine i_0 so $|(u)_{i_0}| = \|u\|_\infty$; $p(k + 1) \leftrightarrow p(i_0)$
    $h_{k+1,k} = (u)_{i_0}$; $l_{k+1} = u/(u)_{i_0}$
  end
  if (an estimate of) $\|b - Ax_k\|$ is small enough or $k = n$ then
    $x_k = x_0 + L_k y_k$, where y_k minimizes $\|H_k y - (r_0)_{i_0} e_1^{(k+1)}\|$
    stop iteration
  end
end
$x_m = x_0 + L_m y_m$, where y_m minimizes $\|H_m y - (r_0)_{i_0} e_1^{(m+1)}\|$, $y \in \mathbb{R}^m$.
$x_0 := x_m$, go to start.

For this algorithm and algorithm 3, we need a stopping criterion. We stop the iteration if the residual satisfies $\|r_k\| \leqslant$ tol. This residual can be obtained by using the formula $r_k = r_0 - AL_k y_k = r_0 - L_{k+1} H_k y_k$. As this formula involves an extra matrix by vector multiplication or a linear combination of the columns of L_{k+1}, instead of computing $\|r_k\|$, we check an upper bound as in the QMR implementation [13]. Since $r_k \in K_{k+1}(r_0, A)$, we have

$$r_k = L_{k+1} s_k.$$

Consequently,

$$\|r_k\| \leqslant \|L_{k+1}\| \|s_k\| \leqslant \sqrt{(n - k/2)(k + 1)}\|s_k\|. \tag{3.9}$$

The number $\|s_k\|$ is obtained from the QR factorization of the matrix H_k. Another alternative is to use $\|s_k\|$ for the convergence criterion, since it is a norm of r_k in $K_{k+1}(r_0, A)$.

3.3. Theoretical results for CMRH

We shall first show that CMRH cannot break down and that it gives the solution of the system (3.1) at the same iteration as GMRES.

Theorem 3. If the degree of the minimal polynomial of the matrix A for the vector r_0 is k, then the iterates x_j in the CMRH method are well defined for $j = 1, \ldots, k$ and x_k is the exact solution of (3.1).

Proof. By hypothesis, we have $\dim(K_k(r_0, A)) = \dim(K_{k+1}(r_0, A)) = k$. Let $v_1 = r_0$ and $v_i = Av_{i-1}$ for $i = 1, \ldots, k+1$, then the matrix $V_k = (v_1, \ldots, v_k)$ is of full rank. Next, by using theorem 2, the matrices L_k and H_{k-1} are of full rank and consequently, the iterates $\{x_j\}$ for $j = 1, \ldots, k-1$ are well defined.

Now, the vector l_k can be written as $l_k = V_k d_1$, and then $Al_k = AV_k d_1$. But the rank of the matrix $V_{k+1} = (r_0, AV_k)$ is k, hence, $Al_k = V_k d_2 = L_k d_3$. Therefore,

$$\tilde{l}_{k+1} = Al_k - L_k L_k^L Al_k = L_k d_3 - L_k L_k^L L_k d_3 = L_k d_3 - L_k d_3 = 0$$

and

$$|h_{k+1,k}| = \left\|\tilde{l}_{k+1}\right\|_\infty = 0.$$

Furthermore, if we partition H_k as

$$H_k = \begin{pmatrix} \widehat{H}_k \\ 0^{\mathrm{T}} \end{pmatrix},$$

then we have $AL_k = L_k \widehat{H}_k$, where $\widehat{H}_k$ is a square nonsingular matrix (the proof of this result is the same as for Arnoldi's algorithm [24]). Consequently,

$$H_k^+ = \left(\widehat{H}_k^{-1}, 0\right)$$

and, thus, $H_k^+ e_1^{(k+1)} = \widehat{H}_k^{-1} e_1^{(k)}$.

Using (3.8) and the last equality, we next have

$$\begin{aligned} r_k &= b - Ax_k = r_0 - (r_0)_{i_0} AL_k \widehat{H}_k^{-1} e_1^{(k)} \\ &= r_0 - (r_0)_{i_0} L_k \widehat{H}_k \widehat{H}_k^{-1} e_1^{(k)} = r_0 - (r_0)_{i_0} l_1 = 0, \end{aligned}$$

which proves the theorem. □

We will now give a result which relates the norm of the residual vector r_k^{CMRH} for CMRH to the residual r_k^{GMRES} for GMRES.

Theorem 4. If $r_0^{\mathrm{CMRH}} = r_0^{\mathrm{GMRES}} = r_0$, then

$$\left\|r_k^{\mathrm{CMRH}}\right\| \leqslant \chi(L_{k+1}) \left\|r_k^{\mathrm{GMRES}}\right\|.$$

Proof. The two vectors r_k^{CMRH} and r_k^{GMRES} are in $K_{k+1}(r_0, A)$ and can be written as

$$r_k^{\mathrm{CMRH}} = L_{k+1} s_k^{\mathrm{CMRH}} \quad \text{and} \quad r_k^{\mathrm{GMRES}} = L_{k+1} s_k^{\mathrm{GMRES}}.$$

Thus,

$$\left\|r_k^{\mathrm{CMRH}}\right\| \leqslant \|L_{k+1}\| \left\|s_k^{\mathrm{CMRH}}\right\|.$$

But (3.7) implies that

$$\left\|s_k^{\mathrm{CMRH}}\right\| \leqslant \left\|s_k^{\mathrm{GMRES}}\right\| = \left\|L_{k+1}^{+} L_{k+1} s_k^{\mathrm{GMRES}}\right\|.$$

Hence,

$$\left\|r_k^{\mathrm{CMRH}}\right\| \leqslant \left\|L_{k+1}^{+}\right\| \left\|L_{k+1}\right\| \left\|r_k^{\mathrm{GMRES}}\right\|,$$

which ends the proof. □

Remarks. 1. For the QMR we have a similar bound [11]:

$$\left\|r_k^{\mathrm{QMR}}\right\| \leqslant \chi(U_{k+1}) \left\|r_k^{\mathrm{GMRES}}\right\|.$$

This result can also be proved by using (3.6) and a proof similar to that of theorem 4.

2. The bound given in theorem 4 depends on $\chi(L_{k+1})$. Unfortunately, this number may be very large and we can construct some examples for which $\chi(L_{k+1}) \geqslant 2^{k-1}$, but we believe that such examples would seldom occur in practice, just as analogous examples very seldom occur for Gaussian elimination with partial pivoting.

On the other hand, for GMRES we have $\chi(H_k) \leqslant \chi(A)$, but for CMRH we do not have such a bound. We found some examples where $\chi(H_k) \leqslant \chi(A)$ and others where $\chi(H_k) \geqslant \chi(A)$. However, in the latter case, $\chi(H_k)$ and $\chi(A)$ were comparable.

3. $\chi(L_k)$ and $\chi(H_k)$ are also important to the numerical soundness of the CMRH method, since $\chi(H_k)$ influences the accuracy in y_k and $\chi(L_k)$ influences the accuracy in the correction $L_k y_k$.

4. Numerical experiments

In this section we give some numerical experiments comparing the performance of the CMRH with GMRES and QMR.

4.1. Full algorithm

For these comparisons we used the implementation of the full GMRES method as described in [25,27], while for QMR we used algorithm 7.1 (QMR based on coupled recurrences without look-ahead) given in [14] without preconditioner.

The following tests were run using MATLAB on a Sun SparcStation 10 with machine precision equal to $2.220 \cdot 10^{-16}$.

The system $Ax = b$ is solved for a random right-hand side. In these experiments we take the initial vector $x_0 = (0, \ldots, 0)^{\mathrm{T}}$ and use the inequality $\|r_k\|_2 / \|b\|_2 \leqslant 10^{-12}$ as stopping criterion. For QMR, we used a random vector y for avoiding breakdown but not the near-breakdown for which one must use a look-ahead implementation. In assessing the comparative performance, we monitored the true residual norms $\|b - Ax_k\|$ in a logarithmic scale, generated for the three methods.

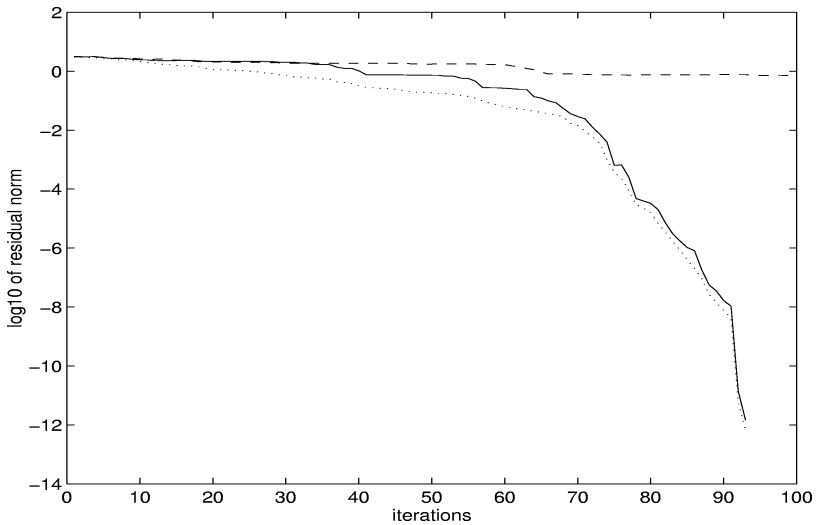

Figure 1. Example 1: $\varepsilon =$ 1e–2, $n = 100$.

In all plots, the solid line is the curve for the CMRH, the dashed line is the curve for the QMR and the dotted line is the curve for the full GMRES.

Example 1. The first example is taken from [16]. Let us consider the $n \times n$ matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 & 1 \\ a_1 & 1 & 1 & \dots & 1 & 1 \\ a_1 & a_2 & 1 & \dots & 1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_1 & a_2 & a_3 & \dots & a_{n-1} & 1 \end{pmatrix}$$

with $a_i = 1 + i\varepsilon$.

We choose $\varepsilon = 10^{-2}$ and $n = 100$. In figure 1 we have plotted the convergence behaviour for the three methods. As the plot indicates, the convergence curves for CMRH and GMRES are similar. CMRH requires 93 steps to converge and gives $\|r_{93}^{\mathrm{CMRH}}\| = 1.4 \cdot 10^{-12}$, GMRES requires 93 steps and gives $\|r_{93}^{\mathrm{GMRES}}\| = 7.0 \cdot 10^{-13}$ and for QMR we obtained $\|r_{100}^{\mathrm{QMR}}\| = 7.1 \cdot 10^{-1}$. After this iteration we have stagnation, and the QMR fails to converge.

In order to compare the number of operations, we used the Matlab function flops, which returns the cumulative number of floating point operations. The flop count for CMRH was $4.4 \cdot 10^6$, for QMR it was $6.4 \cdot 10^6$ and $6.8 \cdot 10^6$ for GMRES.

Example 2. This example is taken from [20]. We consider the $n \times n$ matrix $A = SBS^{-1}$, where

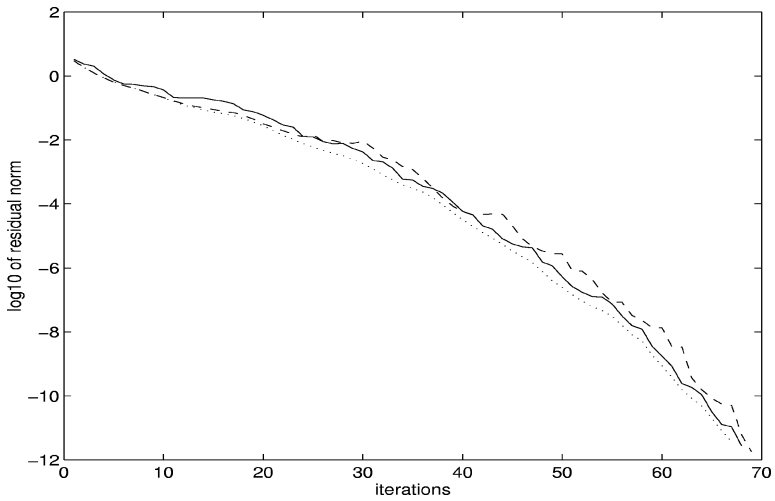

Figure 2. Example 2: $\beta = 0.9$, $\alpha = 1$, $n = 100$.

$$S = \begin{pmatrix} 1 & \beta & & & \\ & 1 & \beta & 0 & \\ & & \ddots & \ddots & \\ & 0 & & \ddots & \beta \\ & & & & 1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1 & & & \\ & 1+\alpha & & 0 \\ & & 3 & \\ & 0 & & \ddots \\ & & & & n \end{pmatrix}.$$

With the variable β, we can control the condition number of the similarity transformation S and make A more non-normal, see [20]. We used two choices of β:

1. $\beta = 0.9$, $\alpha = 1$, $n = 100$ (figure 2). The three methods give a similar behaviour. GMRES is the best and CMRH is slightly better than QMR.
2. $\beta = 1.1$, $\alpha = 0$, $n = 100$ (figure 3). GMRES and CMRH have a similar convergence behaviour. QMR is the best for this example.

Example 3. This example was given by Brown [7] to illustrate the stagnation for the GMRES. We now consider the $n \times n$ matrix

$$A = \begin{pmatrix} \varepsilon & 1 & & & \\ -1 & \varepsilon & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & \varepsilon & 1 \\ & & & -1 & \varepsilon \end{pmatrix}.$$

We chose two values of ε when $n = 40$.

1. $\varepsilon = 0.1$, $n = 40$ (figure 4). The GMRES and CMRH methods are better than QMR.

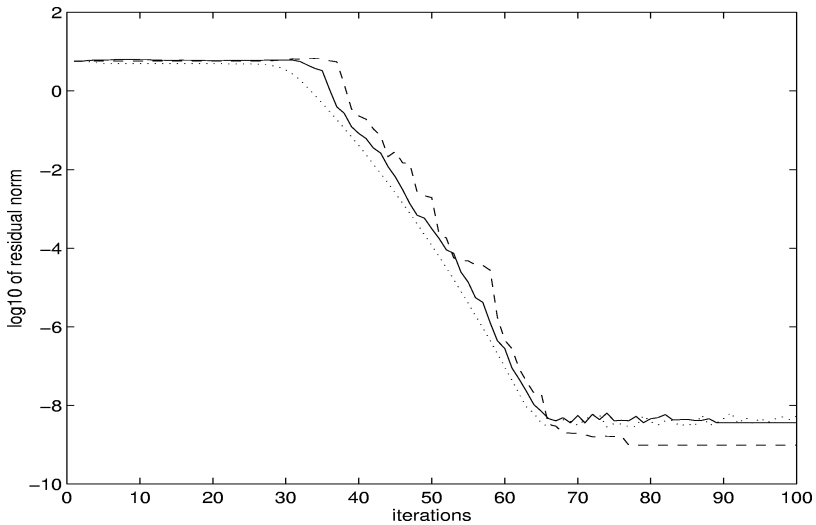

Figure 3. Example 2: $\beta = 1.1$, $\alpha = 0$, $n = 100$.

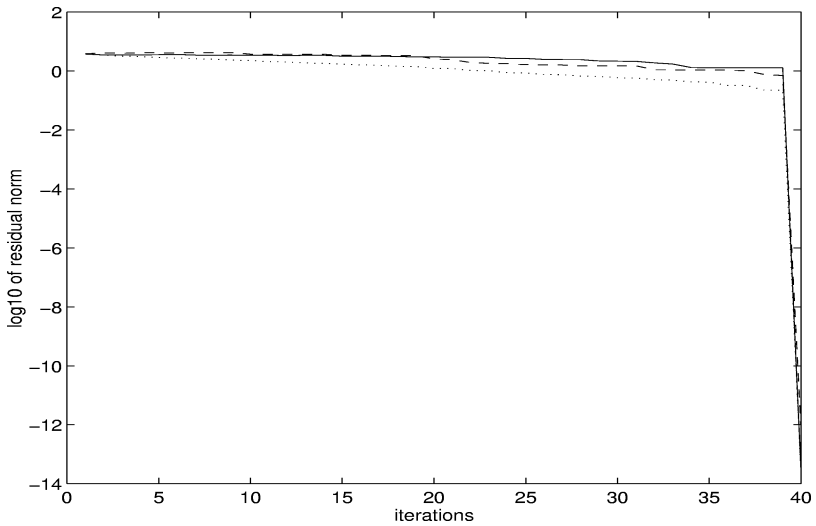

Figure 4. Example 3: $\varepsilon = 0.1$, $n = 40$.

2. $\varepsilon = 10^{-15}$, $n = 40$ (figure 5). The GMRES and CMRH methods have the same convegence curve and the three methods have poor performance. As a final residual, we obtained

$$\left\|r_{40}^{\text{CMRH}}\right\| = 5.5 \cdot 10^{-14}, \quad \left\|r_{40}^{\text{GMRES}}\right\| = 2.9 \cdot 0^{-14} \quad \text{and} \quad \left\|r_{40}^{\text{QMR}}\right\| = 2.2 \cdot 10^{-11}.$$

We also see in the plot that for this example we have stagnation for the three methods.

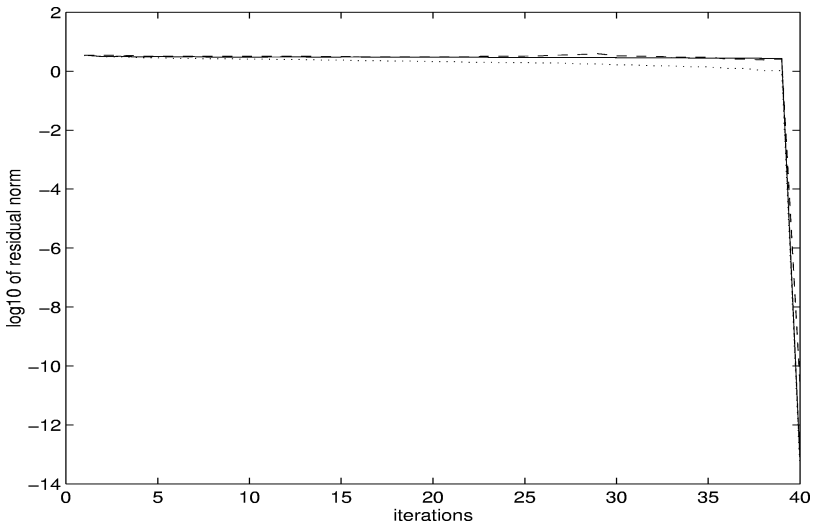

Figure 5. Example 3: $\varepsilon =$ 1e–15, $n = 40$.

4.2. Restarted algorithm

We now present the results of numerical experiments with the restarted algorithm CMRH(m) and compare it with the restarted GMRES(m) method as described in [25,27] and with QMR (algorithm 6.1, QMR based on coupled recurrences, given in [14]) without preconditioner. For QMR the tests were done with the code given in the QMRPACK which is available from Netlib. The experiments are run using the standard f77 compiler on a Sun SparcStation 20.

The stopping test $\|r_k\|/\|r_0\| < 10^{-8}$ is used for GMRES and QMR. The residual norm $\|r_k\|$ for GMRES is obtained from the QR-factorization of the matrix H_k. For CMRH we used the inequality (3.9)

$$\frac{\sqrt{(n-k/2)(k+1)}\|s_k\|}{\|r_0\|} < 10^{-8}$$

as stopping criterion.

Example 4. We derived our test matrix problem by discretizing the boundary value problem [2]

$$-u_{xx}(x,y) - u_{yy}(x,y) + 2p_1u_x(x,y) + 2p_2u_y(x,y) - p_3u(x,y) = G(x,y) \quad \text{on } \Omega,$$
$$u(x,y) = 1 + xy \quad \text{on } \partial\Omega,$$

by finite differences, where Ω is the unit square $\{(x,y) \in \mathbb{R}^2,\ 0 \leqslant x, y \leqslant 1\}$ and p_1, p_2, p_3 are positive constants. The right-hand-side function $G(x,y)$ was chosen so that the true solution is $u(x,y) = 1 + xy$ on Ω. We utilise central differencing to discretize this problem on a uniform $(n+2)*(n+2)$ grid (including grid points on the boundary). Letting $h = 1/(n+1)$, we get a matrix of size $N = n^2$.

Table 1

	GMRES(30)	CMRH(30)	GMRES(50)	CMRH(50)
CPU time	14	16	18	22
Residual norms	$1.9 \cdot 10^{-7}$	$2.9 \cdot 10^{-8}$	$2.1 \cdot 10^{-7}$	$3.3 \cdot 10^{-8}$

Table 2

	GMRES	CMRH	QMR
CPU time	30	31	28
Residual norms	$2.0 \cdot 10^{-7}$	$2.2 \cdot 10^{-9}$	$1.4 \cdot 10^{-7}$

Table 3

	GMRES	CMRH	QMR
CPU time	57	51	39
Residual norms	$1.6 \cdot 10^{-7}$	$3.4 \cdot 10^{-9}$	$1.9 \cdot 10^{-7}$

1. In this example, we let $n = 63$ and choose $p_1 = 1$, $p_2 = 1$ and $p_3 = 10$.

As can be seen from table 1, the CPU time obtained for GMRES is smaller than the one for CMRH, this fact can be explained by the overestimation of the residual norm for CMRH.

For comparison, we give in table 2 the results obtained by GMRES, CMRH and QMR.

2. We keep $n = 63$, $p_1 = 1$ and $p_2 = 1$ but choose $p_3 = 100$.

For this example GMRES(m) and CMRH(m) fail to converge for $m < 120$. Thus for large n, QMR is the only method (among the three methods) to be used for such a problem. We give in table 3 the results obtained by QMR, CMRH and GMRES. These results are obtained after 284 steps for GMRES, 308 steps for CMRH and 326 steps for QMR. We notice again that the estimation of the residual norm used for CMRH is pessimist. Very often $\|s_k\|$ tends to give enough accuracy as an estimate for the residual norm.

5. Conclusion

In this paper we proposed a new iterative method, CMRH, for the iterative solution of a linear system $Ax = b$ with a nonsymmetric nonsingular matrix. This method can be implemented such as to require only about half of the arithmetical operations per iteration and slightly less storage than GMRES.

The main difficulty with this method comes from the condition numbers $\chi(L_k)$ and $\chi(H_k)$, which may be large in theory. However, in practice, we have observed that $\chi(L_k)$ is small and $\chi(H_k)$ is, in general, smaller or comparable to $\chi(A)$.

From our experiments, we saw that CMRH performs accurately and reduces the residual norm about as fast as GMRES.

The CMRH method is interesting for dense matrices and for large sparse matrices with long restart.

We think that CMRH may find some use in the context of parallel computing.

References

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

[2] Z. Bai, D. Hu and L. Reichel, A Newton basis GMRES implementation, IMA J. Numer. Anal. 14 (1994) 563–581.

[3] A. Björck, Least squares methods, in: *Handbook of Numerical Analysis, Vol. I*: *Finite Difference Methods – Solution of Equations in* $\mathbb{R}^n$, eds. P.G. Ciarlet and J.L. Lions (Elsevier/North-Holland, Amsterdam, 1990).

[4] C. Brezinski, Other manifestations of the Schur complement, Linear Algebra Appl. 111 (1988) 231–247.

[5] C. Brezinski, M. Redivo-Zaglia and H. Sadok, Avoiding breakdown and near breakdown in Lanczos type algorithms, Numer. Algorithms 1 (1991) 199–206.

[6] C. Brezinski, M. Redivo-Zaglia and H. Sadok, A breakdown-free Lanczos type algorithm for solving linear systems, Numer. Math. 45 (1992) 361–376.

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

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

[9] H.C. Elman, Iterative methods for large sparse nonsymmetric systems of linear equations, Ph.D. thesis, Computer Science Dept., Yale University, New Haven, CT (1982).

[10] R. Freund, On Conjugate Gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices, Numer. Math. 57 (1990) 285–312.

[11] R. Freund, G.H. Golub and N.M. Nachtigal, Iterative solution of linear systems, Acta Numerica 1 (1992) 57–100.

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

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

[14] R. Freund and N.M. Nachtigal, An implementation of the QMR method based on coupled two-term recurrences, SIAM J. Sci. Statist. Comput. 15 (1994) 313–337.

[15] G. Golub and C.F. van Loan, *Matrix Computations*, 2nd ed. (Johns Hopkins Univ. Press, Baltimore, MD, 1989).

[16] R.T. Gregory and D.L. Karney, *A Collection of Matrices for Testing Computational Algorithms* (Wiley, New York, 1969).

[17] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms Part I, SIAM J. Matrix Anal. Appl. 13 (1992) 594–639.

[18] K. Hessenberg, Behandlung der linearen Eigenwert-Aufgaben mit Hilfe der Hamilton–Cayleychen Gleichung, Darmstadt dissertation (1940).

[19] A.S. Householder, *The Theory of Matrices in Numerical Analysis* (Dover, New York, 1974).

[20] Y. Huang and H.A. van der Vorst, Some observations on the convergence behaviour of GMRES, Delft University of Technology, Report 89-09 (1989).

[21] W.D. Joubert, Lanczos methods for the solution of nonsymmetric systems of linear equations, SIAM J. Matrix Anal. Appl. 13 (1992) 926–943.

[22] W.D. Joubert and G.F. Carey, Parallelizable restarted iterative methods for nonsymmetric linear systems, Internat. J. Comput. Math. 44 (1992) 243–267.
[23] B.N. Parlett, D.R. Taylor and Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, Math. Comp. 44 (1985) 105–124.
[24] Y. Saad, Krylov subspace methods for solving large unsymmetric linear systems, Math. Comp. 37 (1981) 105–126.
[25] Y. Saad and M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
[26] K. Turner and H.F. Walker, Efficient hight accuracy solutions with GMRES(m), SIAM J. Sci. Statist. Comput. 13 (1992) 815–825.
[27] H.A. van der Vorst, The convergence behaviour of some iterative solution methods, Delft University of Technology, Report 89-19 (1989).
[28] H.A. van der Vorst and C. Vuik, The superlinear convergence of GMRES, J. Comput. Appl. Math. 48 (1993) 327–341.
[29] C. Vuik and H.A. van der Vorst, A comparison of some GMRES-like methods, Linear Algebra Appl. 160 (1992) 131–162.
[30] H.F. Walker, Implementation of the GMRES method using Householder transformations, SIAM J. Sci. Statist. Comput. 9 (1988) 152–163.
[31] H.F. Walker and L. Zhou, A simpler GMRES, Utah State University, Report 1/92/54 (1992).
[32] J.H. Wilkinson, *The Algebraic Eigenvalue Problem* (Clarendon Press, Oxford, UK, 1965).
[33] D.M. Young and K.C. Jea, Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods, Linear Algebra Appl. 34 (1980) 159–194.