ELSEVIER

Applied Numerical Mathematics 27 (1998) 123–140

APPLIED NUMERICAL MATHEMATICS

On the QMR approach for iterative methods including coupled three-term recurrences for solving nonsymmetric linear systems ☆

Zhi-Hao Cao

Laboratory of Mathematics for Nonlinear Sciences and Department of Mathematics, Fudan University, Shanghai 200433, People's Republic of China

Abstract

Recently many simple Quasi-Minimal Residual (QMR) approaches have been proposed to improve the convergence behavior of the BI-CG algorithm and its variants (see Chan et al., 1994; Freund, 1993; Freund and Nachtigal, 1991; Freund and Szeto, 1991; Tong, 1994). For using them to obtain improved approximate solutions one needs only to change a few lines in the original algorithms. In most of these approaches the underlying iterative methods to be improved include only two-term recurrences. An exception is the approach of Freund and Nachtigal (1991). In this paper we present a simple but universal QMR approach for constructing QMR variants of any iterative method which includes three-term recurrences. Unified formulas for obtaining improved approximate solutions are derived. The resulting QMR variants can be implemented in a unified manner by adding only a few lines to the original algorithms. Applications of this QMR method to the BICGSTAB2 algorithm of Gutknecht (1993) and the BICGSTAB3 algorithm, which is presented in this paper and is an algorithm with full three-term recurrences, are described. In addition, our QMR approach can also be applied easily to lookhead (i.e., breakdown avoiding) algorithms. Finally, numerical experiments are reported. © 1998 IMACS/Elsevier Science B.V.

Keywords: BI-CG; Nonsymmetric linear systems; Quasi-minimal residual approach; Iterative methods

1. Introduction

It is well known that the Biconjugate Gradient (or nonsymmetric Lanczos) algorithm and its variants (cf. [8,9,12,14]) are powerful algorithms for solving large sparse nonsymmetric systems of linear equations. Recently, many Quasi-Minimal Residual (QMR) approaches have been proposed to improve the convergence behavior of the BI-CG algorithm and most of its variants (cf. [4–7,13]). These approaches are very simple. For using them to obtain improved approximate solutions one needs only to change a few lines in the original algorithm. However, in most of these approaches the underlying iterative algorithms to be improved include only two-term recurrences for the approximate

☆ Supported by the State Major Key Project for Basic Researches and the Doctorial Point Foundation of China.

PII S0168-9274(98)00004-X

solutions. An exception is [6]. As a result, these simple QMR approaches cannot deal with those iterative algorithms, such as BICGSTAB2 [8] and BIORES [9] etc., which include coupled three-term recurrences. Of course, the QMR approach proposed in [6] can deal with, in principle, two-term, three-term and even long-term recurrences; however, the implementation of the resulting algorithm is rather complicated.

In this paper, we first present a simple but universal approach for constructing QMR variants of any iterative algorithm which includes two- or three- or alternating two- and three-term recurrences. The resulting QMR algorithms can be implemented very easily by adding only a few lines to the original iterative algorithm. Then we apply our QMR approach to the BICGSTAB2 [8] and the BICGSTAB3 algorithms. The former is an iterative algorithm with coupled alternating two- and three-term recurrences, the latter is proposed in this paper and is one with coupled three-term recurrences. In addition, we also apply this QMR approach to a breakdown avoiding algorithm [3] which is an iterative algorithm with formal alternating two- and three-term recurrences (cf. Section 2). Finally, numerical examples are reported.

2. QMR approach for iterative algorithm with alternating two- and three-term recurrences

Consider the system of linear equations

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{2.1}$$

where A is an $n \times n$ matrix, $\boldsymbol{x}$ and $\boldsymbol{b}$ are n-dimensional vectors. For an iterative algorithm with alternating two- and three-term recurrences (for example, algorithm BICGSTAB2 [8]) its iterative vectors (i.e., approximate solutions) can be written in the following form (cf. (2.52)):

$$\begin{cases} \boldsymbol{x}_{2m+1} = \boldsymbol{x}_{2m} + \boldsymbol{y}_{2m+1}, \\ \boldsymbol{x}_{2m+2} = (1-\xi_m)\boldsymbol{x}_{2m} + \xi_m \boldsymbol{x}_{2m+1} + \boldsymbol{y}_{2m+2}, \end{cases} \quad m = 0, 1, 2, \ldots. \tag{2.2}$$

Let $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ be the residual vector of the iterate $\boldsymbol{x}_k$, then

$$\begin{cases} \boldsymbol{r}_{2m+1} = \boldsymbol{r}_{2m} - A\boldsymbol{y}_{2m+1}, \\ \boldsymbol{r}_{2m+2} = (1-\xi_m)\boldsymbol{r}_{2m} + \xi_m \boldsymbol{r}_{2m+1} - A\boldsymbol{y}_{2m+2}, \end{cases} \quad m = 0, 1, 2, \ldots. \tag{2.3}$$

Remark 1. Since we assume that an original algorithm with alternating two- and three-term recurrences described by (2.2) and (2.3) is given, vectors $\boldsymbol{y}_{2m+1}$, $\boldsymbol{y}_{2m+2}$ and scalars ξ_m in (2.2) are well known (i.e., they have been defined) and can be computed by the original algorithm (cf. (2.52) and Remark 2). Then vectors $\boldsymbol{r}_{2m+1}$, $\boldsymbol{r}_{2m+2}$ can be computed by (2.3).

If

$$Y_k = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k], \qquad R_k = [\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k-1}], \tag{2.4}$$

then the matrix form of (2.3) is

$$AY_k = R_{k+1} T_k^{(e)}, \tag{2.5}$$

where $T_k^{(e)}$ is a $(k+1) \times k$ matrix.

If $k = 2m+1$, then $T^{(e)}_{2m+1}$ is as follows:

$$T^{(e)}_{2m+1} = \begin{pmatrix} 1 & 1-\xi_0 & & & & & & & \\ -1 & \xi_0 & & & & & & & \\ & -1 & 1 & 1-\xi_1 & & & & & \\ & & -1 & \xi_1 & & & & & \\ & & & \ddots & \ddots & & & & \\ & & & & -1 & 1 & 1-\xi_{m-1} & & \\ & & & & & -1 & \xi_{m-1} & & \\ & & & & & & -1 & 1 & \\ & & & & & & & -1 & \end{pmatrix}. \tag{2.6}$$

If $k = 2m+2$, then

$$T^{(e)}_{2m+2} = \begin{pmatrix} 1 & 1-\xi_0 & & & & & & & & \\ -1 & \xi_0 & & & & & & & & \\ & -1 & 1 & 1-\xi_1 & & & & & & \\ & & -1 & \xi_1 & & & & & & \\ & & & \ddots & \ddots & & & & & \\ & & & & -1 & 1 & 1-\xi_{m-1} & & & \\ & & & & & -1 & \xi_{m-1} & & & \\ & & & & & & -1 & 1 & 1-\xi_m & \\ & & & & & & & -1 & \xi_m & \\ & & & & & & & & -1 & \end{pmatrix}. \tag{2.7}$$

Obviously,

$$T^{(e)}_k = \begin{pmatrix} T_k \\ -\boldsymbol{e}^{(k)\mathrm{T}}_k \end{pmatrix}, \tag{2.8}$$

where T_k is a $k \times k$ matrix and $\boldsymbol{e}^{(k)}_k = [0, \dots, 0, 1]^{\mathrm{T}} \in \mathbb{R}^k$.

Let us begin the derivation of our QMR approach. Assume that the improved iterate (i.e., QMR iterate) $\widehat{\boldsymbol{x}}_k$ is written in the form

$$\widehat{\boldsymbol{x}}_k = \boldsymbol{x}_0 + Y_k \boldsymbol{z},$$

where $\boldsymbol{z} \in \mathcal{C}^k$ will be determined later. By (2.5) the corresponding residual vector satisfies

$$\widehat{\boldsymbol{r}}_k = \boldsymbol{r}_0 - AY_k\boldsymbol{z} = R_{k+1}\big(\boldsymbol{e}^{(k+1)}_1 - T^{(e)}_k \boldsymbol{z}\big). \tag{2.9}$$

Let Ω_{k+1} be a nonsingular scaling matrix and rewrite (2.9) as

$$\widehat{\boldsymbol{r}}_k = R_{k+1}\Omega^{-1}_{k+1}\big(\boldsymbol{f}_{k+1} - H^{(e)}_k \boldsymbol{z}\big), \tag{2.10}$$

where

$$\boldsymbol{f}_{k+1} = \Omega_{k+1}\boldsymbol{e}^{(k+1)}_1 \quad \text{and} \quad H^{(e)}_k = \Omega_{k+1}T^{(e)}_k. \tag{2.11}$$

Then we define the kth QMR iterate $\widehat{\boldsymbol{x}}_k$ as follows:

$$\widehat{\boldsymbol{x}}_k = \boldsymbol{x}_0 + Y_k\boldsymbol{z}_k, \tag{2.12}$$

where $z_k \in C^k$ minimizes the Euclidean norm of the bracketed term in (2.10), i.e., z_k is the solution of the least-squares problem

$$\tau_k \equiv \left\| f_{k+1} - H_k^{(e)} z_k \right\|_2 = \min_{z \in C^k} \left\| f_{k+1} - H_k^{(e)} z \right\|_2. \tag{2.13}$$

Obviously, by (2.6), (2.7) and (2.11), the matrix $H_k^{(e)}$ has full column rank k, thus z_k is uniquely defined by (2.13). If the scaling matrix Ω_{k+1} is upper triangular, then the solution z_k of (2.13) can be updated easily from step to step. This can be derived by using a lemma due to Freund [5]:

Lemma (Freund). *Let* $\delta_1 > 0,\ k \geqslant 1$, *and*

$$H_k^{(e)} = \begin{pmatrix} H_k \\ h_{k+1,k} e_k^{(k)\mathrm{T}} \end{pmatrix} = \begin{pmatrix} H_{k-1}^{(e)} & h_k \\ 0 & h_{k+1,k} \end{pmatrix}, \tag{2.14}$$

be a $(k+1) \times k$ *upper Hessenberg matrix of full column rank* k, *where* $h_k \in C^k$. *For* $m = k-1, k$, *let* z_m *denote the solution of the least-squares problem*

$$\tau_m \equiv \min \left\| f_{m+1} - H_m^{(e)} z \right\|_2, \quad \text{where } f_{m+1} = \delta_1 e_1^{(m+1)} \in \mathbb{R}^{m+1}. \tag{2.15}$$

Moreover, assume that the $k \times k$ *matrix* H_k *in* (2.14) *is nonsingular, and set* $\tilde{z}_k = H_k^{-1} f_k$. *Then*

$$z_k = \left(1 - \zeta_k^2\right) \begin{pmatrix} z_{k-1} \\ 0 \end{pmatrix} + \zeta_k^2 \tilde{z}_k, \tag{2.16}$$

$$\tau_k = \tau_{k-1} \theta_k \zeta_k, \tag{2.17}$$

where

$$\theta_k = \frac{1}{\tau_{k-1}} \left\| f_{k+1} - H_k^{(e)} \tilde{z}_k \right\|_2, \qquad \zeta_k = 1 \Big/ \sqrt{1 + \theta_k^2}.$$

Let us now apply this lemma to the particular least-squares problem (2.13). We will show that when the scaling matrix Ω_{k+1} is taken as a diagonal matrix or a block diagonal matrix, where each block is a 2×2 upper triangular submatrix, then the algorithm for obtaining the improved QMR solution can be implemented very easily by adding a few lines in the original algorithm.

Let us first note that if we eliminate the subdiagonal of T_k (cf. (2.8)), then the resulting triangular matrix $\widehat{T}_k$ is as follows.

If $k = 2m+1$, then

$$\widehat{T}_{2m+1} = \begin{pmatrix} 1 & 1-\xi_0 & & & & & & \\ & 1 & & & & & & \\ & & 1 & 1-\xi_1 & & & & \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & & 1 & 1-\xi_{m-1} & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix}. \tag{2.18}$$

If $k = 2m+2$, then

$$\widehat{T}_{2m+2} = \begin{pmatrix} 1 & 1-\xi_0 & & & & & \\ & 1 & & & & & \\ & & 1 & 1-\xi_1 & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & 1-\xi_m \\ & & & & & & 1 \end{pmatrix}. \tag{2.19}$$

2.1. Case 1. Diagonal scaling matrix

Let

$$\Omega_k = \operatorname{diag}(\delta_1, \ldots, \delta_k), \tag{2.20}$$

where

$$\delta_j = \|\boldsymbol{r}_{j-1}\|_2, \quad j = 1, \ldots, k. \tag{2.21}$$

Then

$$H_k = \Omega_k T_k, \qquad \boldsymbol{f}_k = \delta_1 \boldsymbol{e}_1^{(k)}. \tag{2.22}$$

In order to derive the solution $\boldsymbol{z}_k$ of the least-squares problem (2.13), we need to derive the solution $\widetilde{\boldsymbol{z}}_k$ of the following system of linear equations:

$$H_k \widetilde{\boldsymbol{z}}_k = \boldsymbol{f}_k. \tag{2.23}$$

By (2.8), (2.11) and (2.20) this system is equivalent to the following system of equations:

$$T_k \widetilde{\boldsymbol{z}}_k = \boldsymbol{e}_1^{(k)}. \tag{2.24}$$

By elimination of the subdiagonal of T_k, the above system turns to be equivalent to the system

$$\widehat{T}_k \widetilde{\boldsymbol{z}}_k = \boldsymbol{e}^{(k)}, \tag{2.25}$$

where $\boldsymbol{e}^{(k)} = [1, \ldots, 1]^{\mathrm{T}} \in \mathbb{R}^k$. From (2.18) and (2.19) we have

$$\widehat{T}_{2m+1}^{-1} = \begin{pmatrix} \widehat{T}_{2m}^{-1} & \\ & 1 \end{pmatrix}, \tag{2.26}$$

$$\widehat{T}_{2m+2}^{-1} = \begin{pmatrix} \widehat{T}_{2m+1}^{-1} & -(1-\xi_m)\widehat{T}_{2m+1}^{-1}\boldsymbol{e}_{2m+1}^{(2m+1)} \\ & 1 \end{pmatrix}. \tag{2.27}$$

Therefore, we can derive the solution $\widetilde{\boldsymbol{z}}_k$ as follows:

$$\widetilde{\boldsymbol{z}}_k = \begin{cases} \begin{pmatrix} \widetilde{\boldsymbol{z}}_{2m} \\ 1 \end{pmatrix}, & \text{if } k = 2m+1, \\ \begin{pmatrix} \widetilde{\boldsymbol{z}}_{2m+1} - (1-\xi_m)\boldsymbol{e}_{2m+1}^{(2m+1)} \\ 1 \end{pmatrix}, & \text{if } k = 2m+2. \end{cases} \tag{2.28}$$

By (2.14), (2.20), (2.21) and (2.28), we have

$$\left\|\boldsymbol{f}_{k+1} - H_k^{(e)}\widetilde{\boldsymbol{z}}_k\right\|_2 = |-\delta_{k+1}| = \|\boldsymbol{r}_k\|_2. \tag{2.29}$$

Let

$$\widetilde{\boldsymbol{x}}_k = \boldsymbol{x}_0 + Y_k\widetilde{\boldsymbol{z}}_k, \tag{2.30}$$

from (2.4) and (2.28), we have

$$\widetilde{\boldsymbol{x}}_k = \begin{cases} \widetilde{\boldsymbol{x}}_{2m} + \boldsymbol{y}_{2m+1}, & \text{if } k = 2m+1, \\ \widetilde{\boldsymbol{x}}_{2m+1} - (1-\xi_m)\boldsymbol{y}_{2m+1} + \boldsymbol{y}_{2m+2}, & \text{if } k = 2m+2. \end{cases} \tag{2.31}$$

Then, it follows from (2.12), (2.16) and (2.31) that

$$\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_{k-1} + \zeta_k^2\left(\widetilde{\boldsymbol{x}}_k - \widehat{\boldsymbol{x}}_{k-1}\right). \tag{2.32}$$

If we let

$$\boldsymbol{d}_k = \widetilde{\boldsymbol{x}}_k - \widehat{\boldsymbol{x}}_{k-1}, \tag{2.33}$$

then (2.32) can be rewritten as follows:

$$\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_{k-1} + \zeta_k^2\boldsymbol{d}_k. \tag{2.34}$$

By (2.31) and (2.33) we have

$$\boldsymbol{d}_k = \begin{cases} \boldsymbol{y}_{2m+1} + (1-\zeta_{2m}^2)\boldsymbol{d}_{2m}, & \text{if } k = 2m+1, \\ -(1-\xi_m)\boldsymbol{y}_{2m+1} + \boldsymbol{y}_{2m+2} + (1-\zeta_{2m+1}^2)\boldsymbol{d}_{2m+1}, & \text{if } k = 2m+2. \end{cases}$$

To summarize, $\widehat{\boldsymbol{x}}_k$ *can be computed recursively as follows:*

$$\begin{aligned}
&\widehat{\boldsymbol{x}}_0 = \boldsymbol{x}_0\ (\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0), \qquad \tau_0 = \|\boldsymbol{r}_0\|_2, \qquad \zeta_0 = 1, \qquad \boldsymbol{d}_0 = \boldsymbol{0}. \\
&\textbf{For } k = 1,2,\ldots, \textbf{ do} \\
&\quad \text{(Compute } \boldsymbol{y}_k,\ \boldsymbol{r}_k \text{ from the original algorithm.)} \\
&\quad \theta_k = \frac{\|\boldsymbol{r}_k\|_2}{\tau_{k-1}}, \qquad \zeta_k^2 = \frac{1}{1+\theta_k^2}, \qquad \tau_k = \|\boldsymbol{r}_k\|_2\zeta_k. \\
&\quad \text{(Compute } \xi_m \text{ from the original algorithm if } k = 2m+2.\text{)} \\
&\quad \boldsymbol{d}_k = \begin{cases} \boldsymbol{y}_{2m+1} + (1-\zeta_{2m}^2)\boldsymbol{d}_{2m}, & \text{if } k = 2m+1, \\ \boldsymbol{y}_{2m+2} - (1-\xi_m)\boldsymbol{y}_{2m+1} + (1-\zeta_{2m+1}^2)\boldsymbol{d}_{2m+1}, & \text{if } k = 2m+2, \end{cases} \\
&\quad \widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_{k-1} + \zeta_k^2\boldsymbol{d}_k.
\end{aligned} \tag{2.35}$$

2.2. *Case 2. Block diagonal matrix with* 2×2 *upper triangular submatrices*

In this case, the scaling matrix Ω_{k+1} has the form

$$\Omega_{2m+1} = \begin{pmatrix} \delta_1 & \varepsilon_1 & & & & \\ & \delta_2 & & & & \\ & & \ddots & & & \\ & & & \delta_{2m-1} & \varepsilon_{2m-1} & \\ & & & & \delta_{2m} & \\ & & & & & \delta_{2m+1} \end{pmatrix}, \tag{2.36}$$

$$\Omega_{2m+2} = \begin{pmatrix} \delta_1 & \varepsilon_1 & & & \\ & \delta_2 & & & \\ & & \ddots & & \\ & & & \delta_{2m+1} & \varepsilon_{2m+1} \\ & & & & \delta_{2m+2} \end{pmatrix}, \tag{2.37}$$

where each 2×2 upper triangular submatrix is determined by orthogonormalizing consecutive pairs of $\boldsymbol{r}_i' s$. Therefore, we have

$$\delta_{2j+1} = \|\boldsymbol{r}_{2j}\|_2, \qquad j = 0, 1, 2, \ldots . \tag{2.38}$$

Let

$$\begin{pmatrix} \delta_{2m+1} & \varepsilon_{2m+1} \\ & \delta_{2m+2} \end{pmatrix}^{-1} = \begin{pmatrix} \delta_{2m+1}^{-1} & -\varepsilon_{2m+1}\delta_{2m+1}^{-1}\delta_{2m+2}^{-1} \\ & \delta_{2m+2}^{-1} \end{pmatrix} \equiv \begin{pmatrix} \delta_{2m+1}^{-1} & \widetilde{\varepsilon}_{2m+1} \\ & \widetilde{\delta}_{2m+2} \end{pmatrix} \tag{2.39}$$

and $\widetilde{\delta}_{2m+2}, \widetilde{\varepsilon}_{2m+1}$ $(m = 0, 1, \ldots)$ are determined by the following equations:

$$\begin{cases} \boldsymbol{r}_{2m}^H \big(\widetilde{\varepsilon}_{2m+1}\boldsymbol{r}_{2m} + \widetilde{\delta}_{2m+2}\boldsymbol{r}_{2m+1}\big) = 0, \\ \big\|\widetilde{\varepsilon}_{2m+1}\boldsymbol{r}_{2m} + \widetilde{\delta}_{2m+2}\boldsymbol{r}_{2m+1}\big\|_2 = 1. \end{cases} \tag{2.40}$$

From (2.40) we have

$$\begin{cases} \widetilde{\delta}_{2m+2} = \left(\left\|\boldsymbol{r}_{2m+1} - \dfrac{\boldsymbol{r}_{2m}^H\boldsymbol{r}_{2m+1}}{\|\boldsymbol{r}_{2m+1}\|_2^2}\boldsymbol{r}_{2m}\right\|_2\right)^{-1}, \\ \widetilde{\varepsilon}_{2m+1} = -\dfrac{\boldsymbol{r}_{2m}^H\boldsymbol{r}_{2m+1}}{\|\boldsymbol{r}_{2m}\|_2^2}\widetilde{\delta}_{2m+2}. \end{cases} \tag{2.41}$$

(2.39) and (2.41) imply

$$\begin{cases} \delta_{2m+2} = \left\|\boldsymbol{r}_{2m+1} - \dfrac{\boldsymbol{r}_{2m}^H\boldsymbol{r}_{2m+1}}{\|\boldsymbol{r}_{2m}\|_2^2}\boldsymbol{r}_{2m}\right\|_2, \\ \varepsilon_{2m+1} = \dfrac{\boldsymbol{r}_{2m}^H\boldsymbol{r}_{2m+1}}{\|\boldsymbol{r}_{2m}\|_2}. \end{cases} \tag{2.42}$$

When $k = 2m + 1$ we have

$$H_{2m+1}^{(e)} = \begin{pmatrix} \Omega_{2m+1} & \varepsilon_{2m+1}\boldsymbol{e}_{2m+1}^{(2m+1)} \\ & \delta_{2m+2} \end{pmatrix} \begin{pmatrix} T_{2m+1} \\ -\boldsymbol{e}_{2m+1}^{(2m+1)\mathrm{T}} \end{pmatrix}. \tag{2.43}$$

Thus

$$H_{2m+1} = \Omega_{2m+1}T_{2m+1} - \varepsilon_{2m+1}\boldsymbol{e}_{2m+1}^{(2m+1)}\boldsymbol{e}_{2m+1}^{(2m+1)\mathrm{T}}, \tag{2.44}$$

and the linear system $H_{2m+1}\widetilde{\boldsymbol{z}}_{2m+1} = \boldsymbol{f}_{2m+1}$ is equivalent to the system

$$\left(\widehat{T}_{2m+1} - (\varepsilon_{2m+1}/\delta_{2m+1})\boldsymbol{e}_{2m+1}^{(2m+1)}\boldsymbol{e}_{2m+1}^{(2m+1)\mathrm{T}}\right)\widetilde{\boldsymbol{z}}_{2m+1} = \boldsymbol{e}^{(2m+1)}. \tag{2.45}$$

By (2.18) we have

$$\widetilde{\boldsymbol{z}}_{2m+1} = \widehat{T}_{2m+1}^{-1}\boldsymbol{e}^{(2m+1)} + \eta_{2m+1}\boldsymbol{e}_{2m+1}^{(2m+1)} = \begin{pmatrix} \widetilde{\boldsymbol{z}}_{2m} \\ 1+\eta_{2m+1} \end{pmatrix}, \tag{2.46}$$

where

$$\eta_{2m+1} = \frac{\varepsilon_{2m+1}/\delta_{2m+1}}{1-\varepsilon_{2m+1}/\delta_{2m+1}}. \tag{2.47}$$

When $k = 2m+2$ we have $H_{2m+2} = \Omega_{2m+2}T_{2m+2}$. Thus, the linear system is equivalent to $\widehat{T}_{2m+2}\widetilde{\boldsymbol{z}}_{2m+2} = \boldsymbol{e}^{(2m+2)}$. By (2.27) and (2.46) we have

$$\begin{aligned}\widetilde{\boldsymbol{z}}_{2m+2} &= \begin{pmatrix} \widehat{T}_{2m+1}^{-1}\boldsymbol{e}^{(2m+1)} - (1-\xi_m)\boldsymbol{e}_{2m+1}^{(2m+1)} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \widetilde{\boldsymbol{z}}_{2m+1} - [\eta_{2m+1} + (1-\xi_m)]\boldsymbol{e}_{2m+1}^{(2m+1)} \\ 1 \end{pmatrix}.\end{aligned} \tag{2.48}$$

From (2.46) and (2.48) we have

$$\left\|\boldsymbol{f}_{k+1} - H_k^{(e)}\widetilde{\boldsymbol{z}}_k\right\|_2 = \left|\delta_{k+1}\boldsymbol{e}_k^{(k)\mathrm{T}}\widetilde{\boldsymbol{z}}_k\right| = \begin{cases} |\delta_{2m+2}|\,|1+\eta_{2m+1}|, & \text{if } k = 2m+1, \\ |\delta_{2m+3}|, & \text{if } k = 2m+2, \end{cases} \tag{2.49}$$

and

$$\widetilde{\boldsymbol{x}}_k = \begin{cases} \widetilde{\boldsymbol{x}}_{2m} + (1+\eta_{2m+1})\boldsymbol{y}_{2m+1}, & \text{if } k = 2m+1, \\ \widetilde{\boldsymbol{x}}_{2m+1} - (\eta_{2m+1} + 1 - \xi_m)\boldsymbol{y}_{2m+1} + \boldsymbol{y}_{2m+2}, & \text{if } k = 2m+2. \end{cases} \tag{2.50}$$

It is easy to see that (2.32)–(2.34) are also valid in these case.

To summarize, $\widehat{\boldsymbol{x}}_k$ can be computed recursively as follows:

$\widehat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ $(\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0)$, $\tau_0 = \|\boldsymbol{r}_0\|_2$, $\boldsymbol{d}_0 = \mathbf{0}$, $\zeta_0 = 1$.

For $k = 1, 2, \ldots,$ **do**

if $k = 2m+1$ **then**

(Compute $\boldsymbol{y}_{2m+1}$, $\boldsymbol{r}_{2m+1}$ from the original algorithm.)

$$\delta_k = \|\boldsymbol{r}_{2m}\|_2, \qquad \delta_{k+1} = \left\|\boldsymbol{r}_{2m+1} - \frac{\boldsymbol{r}_{2m}^H\boldsymbol{r}_{2m+1}}{\|\boldsymbol{r}_{2m}\|_2^2}\boldsymbol{r}_{2m}\right\|_2,$$

$$\varepsilon_{2m+1} = \frac{\boldsymbol{r}_{2m}^H\boldsymbol{r}_{2m+1}}{\|\boldsymbol{r}_{2m}\|_2}, \qquad \eta_{2m+1} = \frac{\varepsilon_{2m+1}/\delta_{2m+1}}{1-\varepsilon_{2m+1}/\delta_{2m+1}},$$

$$\theta_k = \frac{|\delta_{2m+2}|}{\tau_{2m}}|1+\eta_{2m+1}|$$

else $(k = 2m+2)$

(Compute ξ_m, $\boldsymbol{y}_{2m+2}$, $\boldsymbol{r}_{2m+2}$ from the original algorithm.)

$$\theta_k = \frac{\|\boldsymbol{r}_{2m+2}\|_2}{\tau_{2m+1}} \tag{2.51}$$

end if

$$\zeta_k^2 = 1/\left(1+\theta_k^2\right), \qquad \tau_k = \tau_{k-1}\theta_k\zeta_k \tag{2.51'}$$

if $k = 2m+1$ **then**

$$\boldsymbol{d}_k = (1+\eta_{2m+1})\boldsymbol{y}_{2m+1} + \left(1-\zeta_{2m}^2\right)\boldsymbol{d}_{2m}$$

else $(k = 2m+2)$

$$\boldsymbol{d}_k = \boldsymbol{y}_{2m+2} - (\eta_{2m+1}+1-\xi_m)\boldsymbol{y}_{2m+1} + \left(1-\zeta_{2m+1}^2\right)\boldsymbol{d}_{2m+1}$$

end if

$$\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_{k-1} + \zeta_k^2\boldsymbol{d}_k.$$

For constructing the QMR variants of BICGSTAB2 [8], we first find out from this algorithm the lines which include the alternating two- and three-term recurrences for the approximate solutions [8, (40e) and (40j)] and compare them with (2.2), we obtain

$$\boldsymbol{y}_{k+1} = \begin{cases} \widetilde{\boldsymbol{s}}_k\omega_k + \widetilde{\boldsymbol{w}}_{k+1}\chi_m, & k = 2m, \\ (\widetilde{\boldsymbol{s}}_{k-1}\omega_{k-1} + \widetilde{\boldsymbol{t}}_k\omega_k)(1-\xi_m) + \widetilde{\boldsymbol{s}}_k\omega_k\xi_m - \widetilde{\boldsymbol{w}}_{k+1}\eta_m, & k = 2m+1. \end{cases} \tag{2.52}$$

Then we add (2.35) and (2.51) to BICGSTAB2 and obtain two QMR variants QBICGSTAB2 and Q2BICGSTAB2, respectively.

Remark 2. If $\xi_m \equiv 1$ in (2.2), we are given an original algorithm with two-term recurrences for the approximate solutions. In this case, if we apply formulas (2.35) to the CGS algorithm [12] and the BICGSTAB algorithm [14], we obtain two variants QTFCGS and QTFSTAB which are slightly different from TFQRM [5] and QMRCGSTAB [4], respectively. Numerical experiments show that they have similar convergence behavior.

The QMR approach discussed above can also be applied to the breakdown avoiding algorithms provided these algorithms include formal two-term or formal alternating two- and three-term recurrences for the approximate solutions. As an example, we take algorithm GBICGSTAB2 [3] which is a breakdown avoiding variant of BICGSTAB2 [8] and was derived by using the theory of formal orthogonal polynomials (FOP) (cf. [1–3,9–11]).

The approximate solutions in GBICGSTAB2 have the following formal alternating two- and three-term recurrences (cf. [3]):

$$\boldsymbol{x}_{n_{2k+1}} = \boldsymbol{x}_{n_{2k}} - w_{2k}(A)\boldsymbol{z}_{2k} - g_{2k}(A)\left(\boldsymbol{r}_{n_{2k}} - Aw_{2k}(A)\boldsymbol{z}_{2k}\right),$$

$$\boldsymbol{x}_{n_{2k+2}} = \begin{cases} \boldsymbol{x}_{n_{2k}} - w_{2k}(A)\boldsymbol{z}_{2k} - w_{2k+1}(A)\left(q_{2k}(A)\boldsymbol{z}_{2k} - c_{2k+1}\boldsymbol{h}_k\right) \\ \quad + \boldsymbol{\delta}_0 + A\boldsymbol{\delta}_1 + \cdots + A^{\widehat{m}_{2k}-1}\boldsymbol{\delta}_{\widehat{m}_{2k}-1}, \\ \quad \text{if } \widehat{m}_{2k} \equiv \frac{1}{2}(m_{2k}+m_{2k+1}) \text{ is integer}, \\ \boldsymbol{x}_{n_{2k}} - w_{2k}(A)\boldsymbol{z}_{2k} - w_{2k+1}(A)\left(q_{2k}(A)\boldsymbol{z}_{2k} - c_{2k+1}\boldsymbol{h}_k\right) \\ \quad + (I + \eta_{\widetilde{m}_{2k+1}+\widehat{m}_{2k}}A)\left(\boldsymbol{\delta}_0 + A\boldsymbol{\delta}_1 + \cdots + A^{\widehat{m}_{2k}-1}\boldsymbol{\delta}_{\widehat{m}_{2k}-1}\right) + \eta_{\widetilde{m}_{2k+1}+\widehat{m}_{2k}}\widetilde{\boldsymbol{r}}_{2k}, \\ \quad \text{if } \widehat{m}_{2k} \equiv \frac{1}{2}(m_{2k}+m_{2k+1}-1) \text{ is integer}. \end{cases} \tag{2.53}$$

These recurrences are formal two- and three-term ones, because from $\boldsymbol{x}_{n_m}$ to $\boldsymbol{x}_{n_{m+1}}$ several breakdown steps have been jumped over. Comparing (2.53) with (2.2) we have

$$\boldsymbol{y}_{2k+1} = -w_{2k}(A)\boldsymbol{z}_{2k} - g_{2k}(A)\big(\boldsymbol{r}_{n_{2k}} - Aw_{2k}(A)\boldsymbol{z}_{2k}\big),$$

$$\boldsymbol{y}_{2k+2} = \begin{cases} -w_{2k}(A)\boldsymbol{z}_{2k} - w_{2k+1}(A)\big(q_{2k}(A)\boldsymbol{z}_{2k} - c_{2k+1}\boldsymbol{h}_k\big) \\ \quad + \boldsymbol{\delta}_0 + A\boldsymbol{\delta}_1 + \cdots + A^{\widehat{m}_{2k}-1}\boldsymbol{\delta}_{\widehat{m}_{2k}-1}, \\ \quad \text{if } \widehat{m}_{2k} \equiv \frac{1}{2}(m_{2k} + m_{2k+1}) \text{ is integer}, \\ -w_{2k}(A)\boldsymbol{z}_{2k} - w_{2k+1}(A)\big(q_{2k}(A)\boldsymbol{z}_{2k} - c_{2k+1}\boldsymbol{h}_k\big) \\ \quad + (I + \eta_{\widetilde{m}_{2k+1}+\widehat{m}_{2k}}A)\big(\boldsymbol{\delta}_0 + A\boldsymbol{\delta}_1 + \cdots + A^{\widehat{m}_{2k}-1}\boldsymbol{\delta}_{\widehat{m}_{2k}-1}\big) + \eta_{\widetilde{m}_{2k+1}+\widehat{m}_{2k}}\widetilde{\boldsymbol{r}}_{2k}, \\ \quad \text{if } \widehat{m}_{2k} \equiv \frac{1}{2}(m_{2k} + m_{2k+1} - 1) \text{ is integer} \end{cases} \tag{2.54}$$

and $\xi_k \equiv 0$. Adding formulas (2.35) to GBICGSTAB2 we obtain its variant QGBICGSTAB2. We will give a a numerical example for this algorithm (cf. Section 4).

3. QMR approach for iterative algorithm with three-term recurrences

The iterative vectors and the residual vectors of this class of algorithm can be written as follows:

$$\boldsymbol{x}_k = (1 - \xi_{k-1})\boldsymbol{x}_{k-2} + \xi_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{y}_k, \tag{3.1}$$

$$\boldsymbol{r}_k = (1 - \xi_{k-1})\boldsymbol{r}_{k-2} + \xi_{k-1}\boldsymbol{r}_{k-1} - A\boldsymbol{y}_k, \tag{3.2}$$

$$k = 1, 2, \ldots \quad (\xi_0 = 1).$$

The matrix $T_k^{(e)}$ in the matrix form of (3.2) is now of the following form:

$$T_k(e) = \begin{pmatrix} 1 & 1-\xi_1 & & & \\ -1 & \xi_1 & & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & & \\ & & & & 1-\xi_{k-1} \\ & & & -1 & \xi_{k-1} \\ & & & & -1 \end{pmatrix} \equiv \begin{pmatrix} T_k \\ -\boldsymbol{e}_k^{(k)\mathrm{T}} \end{pmatrix}. \tag{3.3}$$

If the subdiagonal of $k \times k$ matrix T_k is eliminated, then the resulting upper bidiagonal matrix is as follows:

$$\widehat{T}_k = \begin{pmatrix} 1 & 1-\xi_1 & & & \\ & 1 & 1-\xi_2 & & \\ & & \ddots & \ddots & \\ & & & & 1-\xi_{k-1} \\ & & & & 1 \end{pmatrix}. \tag{3.4}$$

Obviously, we have

$$\widehat{T}_k = \begin{pmatrix} \widehat{T}_{k-1} & (1 - \xi_{k-1}\boldsymbol{e}_{k-1}^{(k-1)}) \\ & 1 \end{pmatrix}, \qquad \widehat{T}_k^{-1} = \begin{pmatrix} \widehat{T}_{k-1}^{-1} & -(1 - \xi_{k-1})\widehat{T}_{k-1}^{-1}\boldsymbol{e}_{k-1}^{(k-1)} \\ & 1 \end{pmatrix}. \tag{3.5}$$

3.1. Case 1. Diagonal scaling matrix

By (2.20)–(2.25) and (3.5) we can derive the solution of the linear system $H_k\widetilde{\boldsymbol{z}}_k = \boldsymbol{f}_k$ as follows:

$$\widetilde{\boldsymbol{z}}_k = \widehat{T}_k^{-1}\boldsymbol{e}^{(k)} = \begin{pmatrix} \widetilde{\boldsymbol{z}}_{k-1} - (1-\xi_{k-1})\widehat{\boldsymbol{z}}_{k-1} \\ 1 \end{pmatrix}, \qquad \widetilde{\boldsymbol{z}}_1 = 1, \tag{3.6}$$

where

$$\widehat{\boldsymbol{z}}_k = \widehat{T}_k^{-1}\boldsymbol{e}_k^{(k)} = \begin{pmatrix} -(1-\xi_{k-1})\widehat{\boldsymbol{z}}_{k-1} \\ 1 \end{pmatrix}, \qquad \widehat{\boldsymbol{z}}_1 = 1. \tag{3.7}$$

It is easy to see that (2.29) and (2.30) are still valid. By (2.4) and (3.6)–(3.7) we have

$$\widetilde{\boldsymbol{x}}_k = \widetilde{\boldsymbol{x}}_{k-1} + \boldsymbol{y}_k - (1-\xi_{k-1})\widetilde{\boldsymbol{y}}_{k-1}, \tag{3.8}$$

where $\widetilde{\boldsymbol{y}}_k \equiv Y_k\widehat{\boldsymbol{z}}_k$ satisfies the following recursive formula:

$$\widetilde{\boldsymbol{y}}_k = \boldsymbol{y}_k - (1-\xi_{k-1})\widetilde{\boldsymbol{y}}_{k-1}. \tag{3.9}$$

(2.32)–(2.34) are also valid.

To summarize, $\widehat{\boldsymbol{x}}_k$ can be computed recursively as follows:

$$\widehat{\boldsymbol{x}}_0 = \boldsymbol{x}_0 \ (\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0), \qquad \tau_0 = \|\boldsymbol{r}_0\|_2, \qquad \boldsymbol{d}_0 = \boldsymbol{0}, \qquad \zeta_0 = 1, \qquad \widetilde{\boldsymbol{y}}_0 = 0. \tag{3.10$'$}$$

For $k = 1, 2, \ldots$ **do**

(Compute ξ_{k-1}, $\boldsymbol{y}_k$, $\boldsymbol{r}_k$ from the original algorithm.)

$$\begin{aligned} &\theta_k = \frac{\|\boldsymbol{r}_k\|_2}{\tau_{k-1}}, \qquad \zeta_k^2 = \frac{1}{1+\theta_k^2}, \qquad \tau_k = \|\boldsymbol{r}_k\|_2\zeta_k, \\ &\boldsymbol{d}_k = \boldsymbol{y}_k - (1-\xi_{k-1})\widetilde{\boldsymbol{y}}_{k-1} + \left(1-\zeta_{k-1}^2\right)\boldsymbol{d}_{k-1}, \\ &\widetilde{\boldsymbol{y}}_k = \boldsymbol{y}_k - (1-\xi_{k-1})\widetilde{\boldsymbol{y}}_{k-1}, \\ &\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_{k-1} + \zeta_k^2\boldsymbol{d}_k. \end{aligned} \tag{3.10}$$

3.2. Case 2. Block diagonal scaling matrix with 2×2 upper triangular submatrices

Obviously, (2.38)–(2.42) are still valid.

When $k = 2m+1$, (2.43)–(2.45) are also valid. By (2.45), (3.5) and the Sherman–Morrison formula we can derive the solution $\widetilde{z}_{2m+1}$ of the system

$$\left(\widehat{T}_{2m+1} - \varepsilon_{2m+1}/\delta_{2m+1}\boldsymbol{e}_{2m+1}^{(2m+1)}\boldsymbol{e}_{2m+1}^{(2m+1)\mathrm{T}}\right)\widetilde{\boldsymbol{z}}_{2m+1} = \boldsymbol{e}^{(2m+1)}$$

as follows:

$$\widetilde{\boldsymbol{z}}_{2m+1} = \begin{pmatrix} \widetilde{\boldsymbol{z}}_{2m} - (1-\xi_{2m})(1+\eta_{2m+1})\widehat{\boldsymbol{z}}_{2m} \\ 1+\eta_{2m+1} \end{pmatrix}. \tag{3.11}$$

Finally, (2.47) is also valid.

When $k = 2m+2$ we have $H_{2m+2} = \Omega_{2m+2} T_{2m+2}$. Thus, the linear system is equivalent to $\widehat{T}_{2m+2} \widetilde{\boldsymbol{z}}_{2m+2} = \boldsymbol{e}^{(2m+2)}$. By (3.5) and (3.7) we have

$$\widetilde{\boldsymbol{z}}_{2m+2} = \begin{pmatrix} \widetilde{\boldsymbol{z}}_{2m+1} + \eta_{2m+1}(1-\xi_{2m})\widehat{\boldsymbol{z}}_{2m} - \eta_{2m+1}\boldsymbol{e}_{2m+1}^{(2m+1)} - (1-\xi_{2m+1})\widehat{\boldsymbol{z}}_{2m+1} \\ 1 \end{pmatrix}. \tag{3.12}$$

From (3.11) and (3.12) we see that (2.49) is still valid and

$$\widetilde{\boldsymbol{x}}_k = \begin{cases} \widetilde{\boldsymbol{x}}_{2m} + (1+\eta_{2m+1})\boldsymbol{y}_{2m+1} - (1+\eta_{2m+1})(1-\xi_{2m})\widetilde{\boldsymbol{y}}_{2m}, & \text{if } k = 2m+1, \\ \widetilde{\boldsymbol{x}}_{2m+1} + \boldsymbol{y}_{2m+2} - \eta_{2m+1}\boldsymbol{y}_{2m+1} + \eta_{2m+1}(1-\xi_{2m})\widetilde{\boldsymbol{y}}_{2m} & \\ \quad -(1-\xi_{2m+1})\widetilde{\boldsymbol{y}}_{2m+1}, & \text{if } k = 2m+2. \end{cases} \tag{3.13}$$

Finally, (2.32)–(2.34) are also valid.

To summarize, $\widehat{\boldsymbol{x}}_k$ *can be computed recursively as follows*:

$\widehat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ $(\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0)$, $\quad \tau_0 = \|\boldsymbol{r}_0\|_2$, $\quad \boldsymbol{d}_0 = \boldsymbol{0}$, $\quad \widetilde{\boldsymbol{y}}_0 = \boldsymbol{0}$, $\quad \zeta_0 = 1$.

For $k = 0, 1, \ldots$ **do**

(Compute ξ_k, $\boldsymbol{y}_{k+1}$, $\boldsymbol{r}_{k+1}$ from the original algorithm.)

if $k = 2m$ **then**

$$\delta_{k+1} = \|\boldsymbol{r}_k\|_2, \qquad \delta_{k+2} = \left\| \boldsymbol{r}_{k+1} - \frac{\boldsymbol{r}_k^H \boldsymbol{r}_{k+1}}{\|\boldsymbol{r}_k\|_2^2}\boldsymbol{r}_k \right\|_2,$$

$$\varepsilon_{2m+1} = \boldsymbol{r}_{2m}^H \boldsymbol{r}_{2m+1} / \|\boldsymbol{r}_{2m}\|_2, \qquad \eta_{2m+1} = \frac{\varepsilon_{2m+1}/\delta_{2m+1}}{1 - \varepsilon_{2m+1}/\delta_{2m+1}},$$

$$\theta_{k+1} = \frac{|\delta_{2m+2}|}{\tau_{2m}} |1 + \eta_{2m+1}|$$

else $(k = 2m+1)$

$$\theta_{k+1} = \|\boldsymbol{r}_{k+1}\|_2 / \tau_k \tag{3.14}$$

end if

$$\zeta_{k+1}^2 = 1/\left(1 + \theta_{k+1}^2\right), \qquad \tau_{k+1} = \tau_k \theta_{k+1} \zeta_{k+1} \tag{3.14'}$$

if $k = 2m$ **then**

$$\boldsymbol{d}_{k+1} = (1+\eta_{2m+1})\boldsymbol{y}_{2m+1} - (1+\eta_{2m+1})(1-\xi_{2m})\widetilde{\boldsymbol{y}}_{2m} + \left(1 - \zeta_{2m}^2\right)\boldsymbol{d}_{2m}$$

else $(k = 2m+1)$

$$\boldsymbol{d}_{k+1} = \boldsymbol{y}_{2m+2} - \eta_{2m+1}\boldsymbol{y}_{2m+1} + \eta_{2m+1}(1-\xi_{2m})\widetilde{\boldsymbol{y}}_{2m} - (1-\xi_{2m+1})\widetilde{\boldsymbol{y}}_{2m+1} + \left(1 - \zeta_{2m+1}^2\right)\boldsymbol{d}_{2m+1}$$

end if

$$\widetilde{\boldsymbol{y}}_{k+1} = \boldsymbol{y}_{k+1} - (1-\xi_k)\widetilde{\boldsymbol{y}}_k,$$

$$\widehat{\boldsymbol{x}}_{k+1} = \widehat{\boldsymbol{x}}_k + \zeta_{k+1}^2 \boldsymbol{d}_{k+1}.$$

Let us now derive an algorithm with coupled three-term recurrences. As we know, the BICGSTAB2 algorithm is derived by alternating one-dimensional and two-dimensional local minimization of the residual norms. If we use in all iterative steps, except for the first one, the two-dimensional local minimization, then we obtain an algorithm with coupled three-term recurrences as follows.

Algorithm BICGSTAB3

1. Start:
   (a) Choose initial guess $\boldsymbol{x}_0$;
   (b) Set $\boldsymbol{p}_0 = \boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$;
   (c) Choose $\widetilde{\boldsymbol{r}}_0$ such that $\rho_0 \equiv \widetilde{\boldsymbol{r}}_0^h \boldsymbol{r}_0 \neq 0$ and $\widetilde{\boldsymbol{r}}_0^H A\boldsymbol{p}_0 \neq 0$.
2. **For** $k = 0, 1, 2, \ldots$ **do**
   (a) $\boldsymbol{v} = A\boldsymbol{p}_k,\ \alpha_k = \rho_k / \widetilde{\boldsymbol{r}}_0^H \boldsymbol{v}$

   if $k \geqslant 1$ **then**

   $\widehat{\boldsymbol{s}}_{k+1} = \boldsymbol{s}_k - \alpha_k \boldsymbol{q}$

   end if

   $\boldsymbol{s}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{v}$

   $\boldsymbol{t} = A\boldsymbol{s}_{k+1}$

   (b) **if** $k = 0$ **then**

   $\omega = \boldsymbol{s}_1^H \boldsymbol{t} / \boldsymbol{t}^H \boldsymbol{t}$

   $\boldsymbol{r}_1 = \boldsymbol{s}_1 - \omega \boldsymbol{t}$

   $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha_0 \boldsymbol{p}_0 + \omega \boldsymbol{s}_1$

   $\rho_1 = \widetilde{\boldsymbol{r}}_0^H \boldsymbol{r}_1,\ \beta_1 = -(\alpha_0/\omega)(\rho_1/\rho_0)$

   $\widehat{\boldsymbol{p}}_1 = \boldsymbol{s}_1 - \beta_1 \boldsymbol{p}_0$

   $\boldsymbol{q} = \boldsymbol{t} - \beta_1 \boldsymbol{v}$

   $\boldsymbol{p}_1 = \boldsymbol{r}_1 - \beta_1(\boldsymbol{p}_0 - \omega \boldsymbol{v})$

   else

   Compute ξ_k and γ_k by solving second order linear system

$$\begin{pmatrix} \|\boldsymbol{s}_{k+1} - \widehat{\boldsymbol{s}}_{k+1}\|_2^2 & (\boldsymbol{s}_{k+1} - \widehat{\boldsymbol{s}}_{k+1})^H A\boldsymbol{s}_{k+1} \\ (A\boldsymbol{s}_{k+1})^H (\boldsymbol{s}_{k+1} - \widehat{\boldsymbol{s}}_{k+1}) & \|A\boldsymbol{s}_{k+1}\|_2^2 \end{pmatrix} \begin{pmatrix} \xi_k \\ \gamma_k \end{pmatrix} = -\begin{pmatrix} (\boldsymbol{s}_{k+1} - \widehat{\boldsymbol{s}}_{k+1})^H \widehat{\boldsymbol{s}}_{k+1} \\ (A\boldsymbol{s}_{k+1})^H \widehat{\boldsymbol{s}}_{k+1} \end{pmatrix}$$

   $\boldsymbol{r}_{k+1} = (1 - \xi_k)\widehat{\boldsymbol{s}}_{k+1} + \xi_k \boldsymbol{s}_{k+1} + \gamma_k \boldsymbol{t}$

   $\boldsymbol{x}_{k+1} = (1 - \xi_k)(\boldsymbol{x}_{k-1} + \alpha_{k-1}\boldsymbol{p}_{k-1} + \alpha_k \widehat{\boldsymbol{p}}_k) + \xi_k(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) - \gamma_k \boldsymbol{s}_{k+1}$

   $\rho_{k+1} = \widetilde{\boldsymbol{r}}_0^H \boldsymbol{r}_{k+1},\ \beta_{k+1} = (\alpha_k/\gamma_k)(\rho_{k+1}/\rho_k)$

$$\widehat{\boldsymbol{p}}_{k+1} = \boldsymbol{s}_{k+1} - \beta_{k+1}\boldsymbol{p}_k$$
$$\boldsymbol{q} = \boldsymbol{t} - \beta_{k+1}\boldsymbol{v}$$
$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \beta_{k+1}\big((1-\xi_k)\widehat{\boldsymbol{p}}_k + \xi_k\boldsymbol{p}_k - \gamma_k\boldsymbol{v}\big)$$

end if.

We shall see from the numerical examples that BICGSTAB3 often has better convergence behavior than that of BICGSTAB2.

Let us now construct the QMR variants of BICGSTAB3. First, we find out the lines which include the three-term recurrences for the approximate solutions and compare them with (3.1), we have

$$\boldsymbol{y}_1 = \alpha_0\boldsymbol{p}_0 + \omega\boldsymbol{s}_1,$$
$$\boldsymbol{y}_{k+1} = (1-\xi_k)(\alpha_{k-1}\boldsymbol{p}_{k-1} + \alpha_k\widehat{\boldsymbol{p}}_k) + \xi_k\alpha_k\boldsymbol{p}_k - \gamma_k\boldsymbol{s}_{k+1}, \quad k \geqslant 1.$$

Then, we add (3.10) and (3.14) to BICGSTAB3 and obtain two QMR variants QBICGSTAB3 and Q2BICGSTAB3, respectively.

4. Numerical examples

Example 1. This example comes from the discretization of the convection–diffusion equation (cf. [5,7,13])

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + \gamma\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y}\right) + \beta u = f \quad \text{on } (0,1)\times(0,1) \tag{4.1}$$

with Dirichlet boundary condition, where $\gamma = 100$, $\beta = -100$.

We discretize (4.1) using centered differences on a uniform 100×100 grid. The right-hand side was chosen such that the vector of all ones is the exact solution of the linear system. The initial guess was $\boldsymbol{x}_0 = 0$, and $\widetilde{\boldsymbol{r}}_0$ was chosen as a vector with random entries distributed uniformly in $[-1,1]$. As stopping criterion, we used

$$\frac{\|\widehat{\boldsymbol{r}}_k\|_2}{\|\boldsymbol{r}_0\|_2} \leqslant 10^{-6}. \tag{4.2}$$

The results of the computation are shown in Figs. 1–3. From Fig. 1 we can see clearly that for this example the BICGSTAB2 algorithm has better convergence behavior than that of the BICGSTAB algorithm and the BICGSTAB3 algorithm has better convergence behavior than that of BICGSTAB2. However, the BICGSTAB0 algorithm, which is a variation of BICGSTAB and consists of replacing $\omega_i = (\boldsymbol{t}_i, \boldsymbol{s}_i)/(\boldsymbol{t}_i, \boldsymbol{t}_i)$ in algorithm BICGSTAB [14] with $\omega_i = (\boldsymbol{s}_i, \boldsymbol{s}_i)/(\boldsymbol{t}_i, \boldsymbol{s}_i)$ (cf. [4,14]), has the best convergence behavior.

Example 2 [2]. The $n \times n$ matrix A is the one with all its elements equal to zero except

$$a_{1,n} = -1,$$
$$a_{i,i-1} = 1, \quad i = 2, \ldots, n.$$

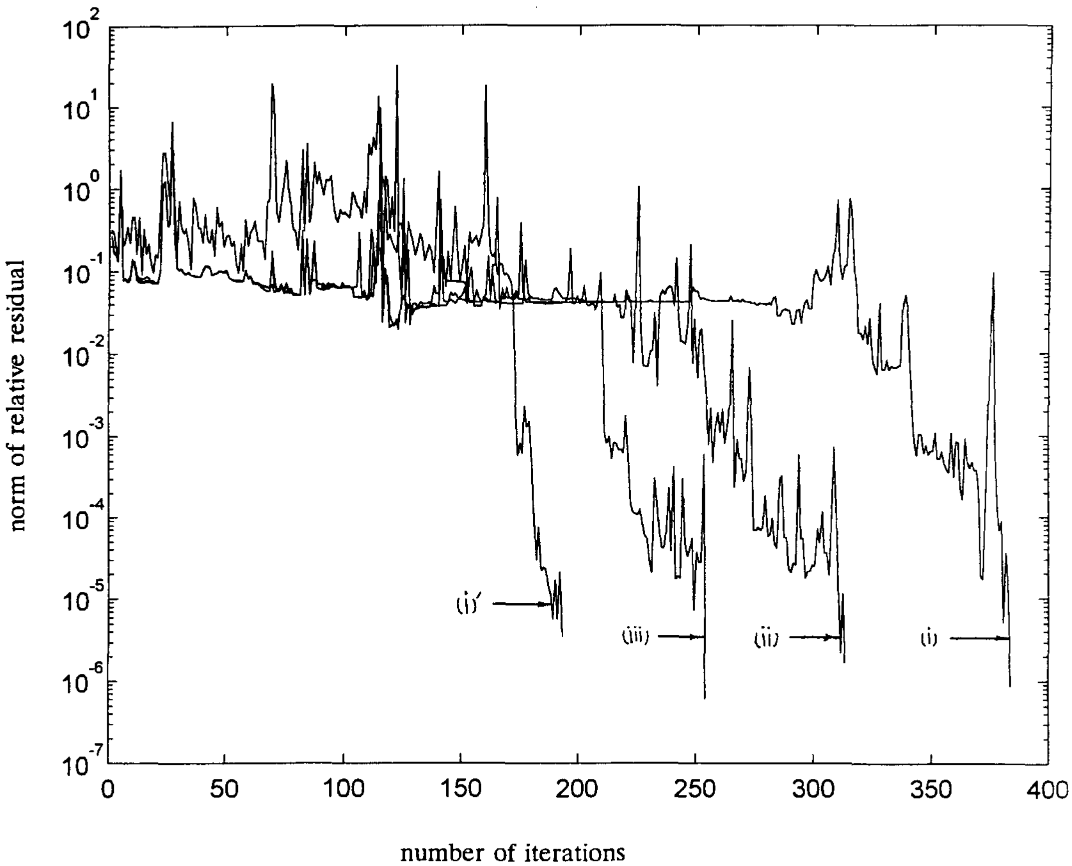

Fig. 1. (i) Algorithm BICGSTAB. (i)′ Algorithm BICGSTAB0. (ii) Algorithm BICGSTAB2. (iii) Algorithm BICGSTAB3.

Table 1

			QGBICGSTAB2		GBICGSTAB2	
k	m_k	n_k	$\|r^{(m)}_{n_k+1}\|/\|r_0\|$	$\|b - Ax^{(m)}_{n_k+1}\|/\|r_0\|$	$\|r_{n_k+1}\|/\|r_0\|$	$\|b - Ax_{n_k+1}\|/\|r_0\|$
0	1	0	0.19898769	0.19898769	0.20313290	0.20313290
1	1	1	0.20486082	0.20486082	0.23094011	0.23094011
2	96	2	0.21040157	0.21040157	0.23011142	0.23011143
3	1	98	0.11354824	0.11354824	0.11301727	0.11308238
4	1	99	2.28511E−4	2.28511E−4	4.80533E−10	2.28511E−4

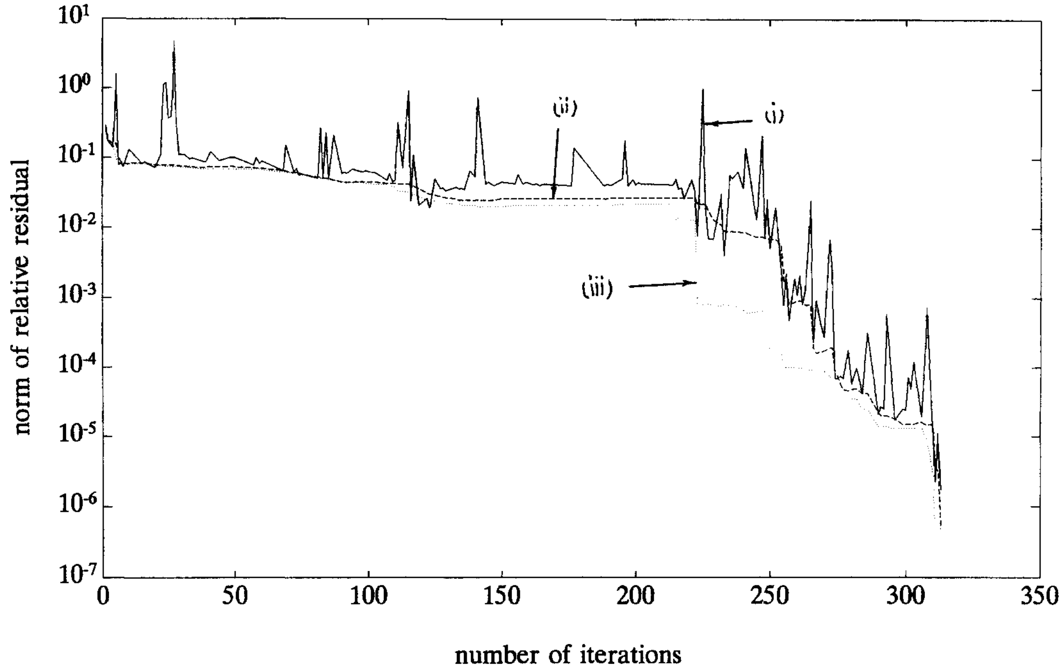

Fig. 2. (i) Algorithm BICGSTAB2. (ii) Algorithm QBICGSTAB2. (iii) Algorithm Q2BICGSTAB2.

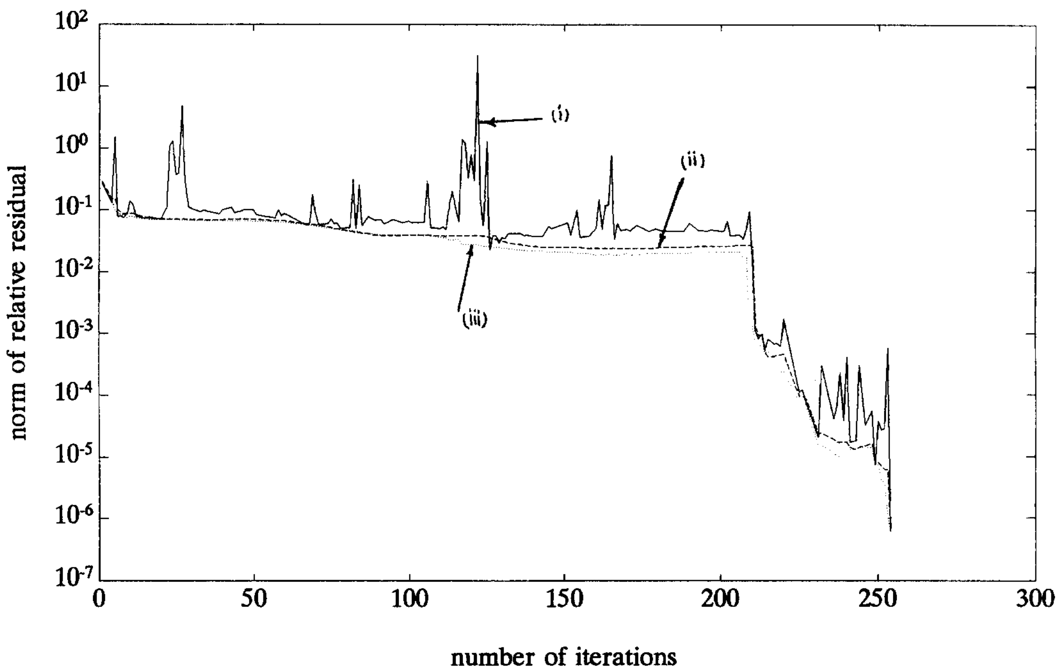

Fig. 3. (i) Algorithm BICGSTAB3. (ii) Algorithm QBICGSTAB3. (iii) Algorithm Q2BICGSTAB3.

For $\boldsymbol{b} = [-1, 1, \ldots, 1]^{\mathrm{T}}$, the solution is $\boldsymbol{x} = [1, \ldots, 1]^{\mathrm{T}}$. Set $\boldsymbol{x}_0 = 0$ and $\widetilde{\boldsymbol{r}}_0 = [1, \ldots, 1]^{\mathrm{T}}$. For $n = 100$, the results are shown in Table 1.

In this example both QGBISTAB2 and GBISTAB2 avoid 95 breakdowns, i.e., the residual polynomials from degree 4 to degree 98 do not exist. The two algorithms are stagnating, i.e., they are not convergent until $n_k + 1 = n$. We note that the residual norms $\|\boldsymbol{r}_{n_k+1}\|_2$ computed recursively by the GBICGSTAB2 are misleading compared to the true residual norms $\|\boldsymbol{b} - A\boldsymbol{x}_{n_k+1}\|_2$. On the other hand, the residual norms $\|\boldsymbol{r}^{(m)}_{n_k+1}\|_2$ computed recursively by the QGBICGSTAB2 are rather accurate.

5. Conclusions

In this paper, we propose a simple but universal QMR approach to construct smoothly convergent variants of any iterative algorithm for solving nonsymmetric systems, provided the iterative algorithm includes two-term or three-term or alternating two- and three-term recurrences for the approximate solutions. The construction is unified and very simple. In order to obtain a QMR variant one only needs to take the two-term or three-term or alternating two- and three-term recurrences for the approximate solutions of the form (2.2) or (3.1) to obtain $\boldsymbol{y}_m$ or ξ_m, $\boldsymbol{y}_{2m+1}$, $\boldsymbol{y}_{2m+2}$, or ξ_k, $\boldsymbol{y}_k$, respectively. Then one has to incorporate (2.35) (or (2.51)), or (3.10) (or (3.14)) into the original algorithm.

Besides its simplicity and unification, our QMR approach can also be applied to the breakdown avoiding algorithms, provided they include formal two-term or three-term or alternating two- and three-term recurrences for the approximate solutions.

Acknowledgements

I would like to thank the referees for their instructive comments and suggestions, which helped to improve the presentation of this paper.

References

[1] C. Brezinski, M. Redivo-Zaglia and H. Sadok, A breakdown-free Lanczos type algorithm for solving linear systems, *Numer. Math.* 63 (1992) 29–38.

[2] C. Brezinski, M. Redivo-Zaglia and H. Sadok, Avoiding breakdown and near-breakdown in Lanczos type algorithm, *Numer. Algorithms* 1 (1991) 261–284.

[3] Z.H. Cao, Avoiding breakdown in variants of the BI-CGSTAB algorithm, *Linear Algebra Appl.* 263 (1997) 113–132.

[4] T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto and C.H. Tong, A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems, *SIAM J. Sci. Comput.* 15 (1994) 338–347.

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

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

[7] R.W. Freund and T. Szeto, A quasi-minimal residual squared algorithm for non-Hermitian linear systems, Technical Report 91.26, RIACS NASA Ames Research Center, Moffett Field, CA (1991).

[8] M.H. Gutknecht, Variants of BICGSTAB for matrices with complex spectrum, *SIAM J. Sci. Comput.* 14 (1993) 1020–1033.
[9] M.H. Gutknecht, The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the *gd* algorithm, in: *Proceedings of the Copper Moutain Conference on Iterative Methods* (1990).
[10] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, *SIAM J. Matrix Anal. Appl.* 13 (1992) 594–634.
[11] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part II, *SIAM J. Matrix Anal. Appl.* 15 (1994) 15–58.
[12] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Comput.* 10 (1989) 36–52.
[13] C.H. Tong, A family of quasi-minimal residual methods for nonsymmetric linear systems, *SIAM J. Sci. Comput.* 15 (1994) 89–105.
[14] H.A. van der Vorst, Bi-CGSTAB: a fast and smoothing converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Comput.* 13 (1992) 631–644.
[15] L. Zhou and H.F. Walker, Residual smoothing techniques for iterative methods, *SIAM J. Sci. Comput.* 15 (1994) 297–312.