ELSEVIER

Journal of Computational and Applied Mathematics 54 (1994) 65–78

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

On the squared unsymmetric Lanczos method

A.T. Chronopoulos *,[1]

Computer Science Department, University of Minnesota, Minneapolis, MN 55455, United States

Received 7 June 1992; revised 5 November 1992

Abstract

The biorthogonal Lanczos and the biconjugate gradient methods have been proposed as iterative methods to approximate the solution of nonsymmetric and indefinite linear systems. Sonneveld (1989) obtained the conjugate gradient squared by squaring the matrix polynomials of the biconjugate gradient method. Here we square the unsymmetric (or biorthogonal) Lanczos method for computing the eigenvalues of nonsymmetric matrices. Three forms of restarted squared Lanczos methods for solving unsymmetric linear systems of equations were derived. Numerical experiments with unsymmetric (in)definite linear systems of equations comparing these methods to a restarted (orthogonal) Krylov subspace iterative method showed that the new methods are competitive and they require that a fixed small number of direction vectors be stored in the main memory.

Keywords: Squared biorthogonal unsymmetric Lanczos method; Restarted methods

1. Introduction

Consider a linear system of equations

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

where A is a real unsymmetric matrix of order N. The transpose of the matrix A will be denoted as A^*. Throughout this article lower-case characters will denote vectors and Greek letters will denote scalars or real functions. Characters with the hat symbol will only denote *matrix polynomials* in A or A^*.

The conjugate gradient or the Lanczos method apply to (1) if A is symmetric and positive definite [8,14]. Paige and Saunders [16] have obtained variants of the Lanczos method (called SYMMLQ and MINRES) for indefinite symmetric systems [14]. Generalizations of the method of conjugate

* Present address: Department of Computer Science, Wayne State University, State Hall 431, 5143 Cass Avenue, Detroit, MI 48202, United States, e-mail: chronos@cs.wayne.edu.

[1] This work was supported by NSF (CCR-8722260).

0377-0427/94/$07.00 © 1994 Elsevier Science B.V. All rights reserved
SSDI 0377-0427(92)00129-Y

gradients to (Krylov subspace based) iterative methods for unsymmetric systems have been derived by several authors (see, for example, [1,3,4,6,19,23]).

Faber and Manteuffel [7] proved that any Krylov subspace based variational method would require to store a number of direction vectors, which may be equal to the dimension N of the linear system, to ensure termination of the process in at most N steps. Thus all the methods described above seem to need storage of an a priori unspecified number of vectors (in addition to the matrix). This number depends on the nonsymmetry and indefiniteness and condition number of the matrix. The biorthogonal Lanczos method for solving linear systems [18], the biconjugate gradients method [8] and the biorthogonal Orthodir(2) methods [5,11,12] do not have this limitation. In the absence of *breakdown*, these methods converge in at most N steps with a modest main memory storage requirement. Several authors have obtained generalizations of biorthogonal methods with fewer *non-breakdown conditions* [11,13,17] than the standard biorthogonal methods.

The conjugate gradient squared method (CGS) [20] was derived from the biconjugate gradients method by simply squaring the residual and direction matrix polynomials. CGS does not need multiplication by the transpose of a matrix. Thus it turns out that CGS is in practice faster than the biconjugate gradients method, though the contrary may occur in some cases [21]. CGS computes exactly the same parameters as the biconjugate gradients method and so it has exactly the same non-breakdown conditions as the biconjugate gradients method. Our recent results include the derivation of squared versions of the biorthogonal Lanczos method for eigenvalues, the biconjugate residual method and biorthogonal Orthodir(2) [5,15]. Some of these results have also been independently obtained in [11,12]. Other authors have derived squared versions of the biorthogonal Lanczos method for linear systems [2]. However, these algorithms are simply transpose-free versions of the biorthogonal Lanczos method for solving linear systems.

In this article, we square the biorthogonal Lanczos iteration for eigenvalues. The squared Lanczos method forms the same tridiagonal matrix T_m as the biorthogonal Lanczos method. The need for multiplication by the matrix transpose has been eliminated. We then obtain squared forms the restarted biorthogonal Lanczos method for linear systems. We compare the restarted squared Lanczos methods to the restarted Generalized Minimal Residual Method (GMRES) [19].

In Section 2 we describe the biorthogonal Lanczos method for eigenvalues and discuss convergence conditions. In Section 3 we review the biorthogonal Lanczos method for unsymmetric linear systems and derive more robust variants of it. In Section 4 we derive the squared Lanczos method for eigenvalues of unsymmetric matrices. In Section 5 we derive the restarted squared Lanczos methods for solving unsymmetric linear systems of equations. In Sections 6 and 7 we present numerical tests comparing the new squared methods to GMRES and we draw conclusions.

2. The biorthogonal Lanczos method

Lanczos [14] introduced a biorthogonal vector generation method and used it to approximate the eigenvalues of unsymmetric matrices. This method can also be used to solve unsymmetric and indefinite linear systems of equations. In this section we review the biorthogonal Lanczos method. This method in the absence of breakdown generates a double sequence of vectors v_i, w_i which is biorthogonal. This means that $(v_i, w_j) = 0$, for $i \neq j$.

Algorithm 1 (*Biorthogonal Lanczos method*).
$\beta_1 = \delta_1 = 0$, $v_0 = w_0 = 0$
v_1 and w_1 with $(v_1, w_1) = 1$
For $i = 1, \ldots, m$ **do**
(1) Av_i, A^*w_i
(2) $\alpha_i = (Av_i, w_i)$
(3) $t_{i+1} = Av_i - \alpha_i v_i - \beta_i v_{i-1}$
(4) $s_{i+1} = A^*w_i - \alpha_i w_i - \delta_i w_{i-1}$
(5) $\gamma_{i+1} = (t_{i+1}, s_{i+1})$
(6) Select β_{i+1}, δ_{i+1}: $\beta_{i+1}\delta_{i+1} = \gamma_{i+1}$
(7) $v_{i+1} = t_{i+1}/\delta_{i+1}$
(8) $w_{i+1} = s_{i+1}/\beta_{i+1}$
EndFor

This method requires modest storage and the computational work is ($14N$ Ops $+2$ Mv) per iteration, where Mv stands for matrix–vector product by the matrix A or A^* and Ops denotes the floating-point operations *addition* or *multiplication*.

The method breaks down if for some index the inner product $\gamma_{i+1} = (t_{i+1}, s_{i+1})$ is zero. If the method does not break down, then the vectors v_i, w_i are biorthogonal and $(v_i, w_i) = 1$. A standard selection for β_i, δ_i is

$$\delta_{i+1} = |\gamma_{i+1}|^{1/2}, \qquad \beta_{i+1} = \delta_{i+1}\,\mathrm{sign}(\gamma_{i+1}).$$

In the absence of breakdown, a tridiagonal matrix $T_m = \mathrm{tridiag}[\delta_{i+1}, \alpha_i, \beta_{i+1}]$, with $i = 1, \ldots, m$, is formed. The following block vector equation holds:

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^{\mathrm{T}}, \tag{2}$$

where $V_m = [v_1, \ldots, v_m]$ and $e_m^{\mathrm{T}} = [0, \ldots, 0, 1]$. The matrix T_m is known to have extreme eigenvalues which approximate the extreme eigenvalues of the unsymmetric matrix A.

The *non-breakdown* conditions for the biorthogonal Lanczos method (i.e., $(v_i, w_i) \neq 0$) can be expressed in terms of the matrices of moments of the initial vectors. The following result is proved in [18].

Proposition 1 (Saad [18]). *Let us assume that* $w_1 = v_1$ *in Algorithm* 1. *Let* M_k *be the moment matrices of dimension* k *with entries* $m_{ij} = (A^{i+j-2}v_1, w_1)$. *The first* m *iterations of the biorthogonal Lanczos method can be completed if and only if*
(i) $\det(M_k) \neq 0$, $k = 1, \ldots, m$.

Proof. From [18]. □

We next prove that at least every other matrix T_m is nonsingular.

Proposition 2. *Let us assume that Algorithm* 1 *does not break down. Then the matrices* T_m *are nonsingular for at least every other index* m *for* $1 < m$.

Proof. For $m = 1$ or $m = 2$ the truth of the proposition is easily checked since β_i and δ_i are not zero. Let us assume that T_{m-1} is singular for $2 < m$. We apply the Givens QR method to the matrix T_m to find its rank. We first apply row permutations $P = P_{m,m+1} \cdots P_{1,2}$ to the matrix T_m. This moves the first row of T_m to the bottom of the matrix, and leaves us with a new matrix $H_m = PT_m$ consisting of an $(m-1) \times (m-1)$ upper triangular matrix augmented with a single row. The matrices H_m and T_m have the same rank. So we will apply QR to the matrix H_m. Let $Q_{i,m}$ be the Givens rotation of the ith and mth rows of H_m which annihilates the entry (m, i) of H_m. Then the orthogonal transformation $Q_{(m-1,m)} \cdots Q_{(1,m)}$ reduces H_m to upper triangular.

For illustration purposes we consider the matrix T_m for $m = 5$ and denote by x its nonzero entries:

$$\begin{bmatrix} x & x & 0 & 0 & 0 \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{bmatrix}.$$

The matrix H_m has the following form for $m = 5$:

$$\begin{bmatrix} x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ x & x & 0 & 0 & 0 \end{bmatrix}.$$

We observe that T_m contains T_{m-1} as a leading submatrix. Also, the matrix H_m contains the matrix H_{m-1} as a submatrix. We now consider the application of Givens QR to H_m. From the assumption the rank of matrix H_{m-1} equals $m-2$. This implies that the application of the $(m-2)$th row rotation $(Q_{(m-2,m)} \cdots Q_{(1,m)})$ eliminates all the entries of the last row except for entry (m, m) which equals $\cos(\theta)\beta_m$ (where $\cos(\theta) \neq 0$ is the cosine term in the rotation). After the last rotation has been applied, the matrix becomes

$$\begin{bmatrix} x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{bmatrix}.$$

Now it is clear that H_m has rank m. □

3. The biorthogonal Lanczos method for linear systems

The biorthogonal Lanczos method can be used to solve the linear system of equations (1). Let r_1 denote the initial residual $[b - Ax_1]$, where x_1 is the initial guess solution vector. We select $v_1 = r_0/\|r_0\|$ and $w_1 = v_1$. Let us assume that no breakdown occurs (in Algorithm 1) and the biorthogonal subspaces $V_{m+1} = [v_1, \ldots, v_{m+1}]$, $W_{m+1} = [w_1, \ldots, w_{m+1}]$ are computed. The subspace V_m can be used to construct an approximate solution of (1) as follows [18]:

$$x_{m+1} = x_1 + V_m z_m. \tag{3}$$

Using (2), we write the residual vector as follows:

$$r_{m+1} = \|r_1\| v_1 - V_m T_m z_m + (\delta_{m+1} z_m^m) v_{m+1} = V_{m+1}\left[\|r_1\| e_1 - \bar{T}_m z_m\right], \tag{4}$$

where e_j for $j = 1, 2, \ldots$ are the Euclidean basis vectors, $z_m^m = e_m^{\mathrm{T}} z_m$ and the matrix $\bar{T}_m$ equals the matrix T_m plus the additional row $\delta_{m+1} e_m^{\mathrm{T}}$. In the standard Lanczos method it is required that the residual $r_{m+1} = r_1 - AV_m z_m$ is biorthogonal to W_m. This requirement and (4) lead to the linear system

$$T_m z_m = \|r_1\| e_1. \tag{5}$$

The matrix T_m is assumed nonsingular [14,18]. The solution z_m is then used in (3) to compute x_{m+1}. From (4) and (5) it follows that the residual norm can be obtained from the formula

$$r_{m+1} = \sigma_m v_{m+1}, \tag{6}$$

where $\sigma_m = z_m^m \delta_{m+1}$.

We will now remove the assumption that the matrix T_m is nonsingular and still define a biorthogonal Lanczos method for linear systems. We only assume that Algorithm 1 does not break down. Let us now consider two ways of obtaining z_m.

Method I. This method will be called Biorthogonal Lanczos QR method (BiLQR). Linear system (5) is equivalent to

$$H_m z_m = \|r_1\| e_m.$$

We apply QR decomposition to this linear system. When the matrix A is symmetric, this method is similar to Paige and Saunders' SYMMLQ [16].

Now, let H_m be singular. Then it has rank $m-1$. We propose two approaches for a robust BiLQR method. The first approach checks for singular H_m and makes use of Proposition 2 to guarantee that H_{m+1} is nonsingular. The second approach modifies H_m regardless if it is singular or not.

(a) In the QR decomposition the (m, m) entry of $Q_{(m-2),m} \cdots Q_{1,m} H_m$ is checked and if it is nearly zero, the algorithm moves to form H_{m+1} and solve $H_{m+1} z_{m+1} = \|r_1\| e_{m+1}$. The residual vector can be computed from (6) and its norm can be used to monitor the convergence of the method.

(b) We modify the $(m-1)$th row of H_m to $[0, \ldots, 0, \delta_m, \pm\delta_m]$. This new matrix will be denoted by $\tilde{H}_m$. The sign $\pm$ is chosen equal to $(-1$ *times*$)$ *(product of the signs of the entries* $(m, m-1)$ *and* (m, m) *of* $Q_{(m-2),m} \cdots Q_{1,m} H_m)$. This selection makes the $(m-1)$th row and the last row of $Q_{(m-2),m} \cdots Q_{1,m} H_m$ linearly independent. This is easily checked because the entry (m, m) of $Q_{(m-2),m} \cdots Q_{1,m} H_m$ equals β_{m-1} and it is nonzero by assumption. Thus the only way that rows $m-1$ and m of $Q_{(m-2),m} \cdots Q_{1,m} H_m$ may become linearly dependent occurs by having nearly the same entries $(m-1, m-1)$, $(m-1, m)$ and $(m, m-1)$, (m, m), respectively. This possibility is eliminated by the choice of $\pm$ in forming the entry $(m-1, m)$ of $\tilde{H}_m$. We now rewrite (4) for the residual using this method:

$$r_{m+1} = V_m\left[\|r_1\| e_m - H_m z_m\right] + \sigma_m v_{m+1} = V_m\left[\|r_1\| e_m - \tilde{H}_m z_m\right] + \tilde{\sigma}_m v_m + \sigma_m v_{m+1}, \tag{7}$$

where $\tilde{\sigma}_m = (\mp\delta_m - \alpha_m) z_m^m$. Since the matrix $\tilde{H}_m$ is invertible, we solve

$$\tilde{H}_m z_m = \|r_1\| e_m \tag{8}$$

(via QR decomposition) and obtain the residual vector

$$r_{m+1} = \tilde{\sigma}_m v_m + \sigma_m v_{m+1}. \tag{9}$$

It is clear that r_{m+1} is biorthogonal to W_{m-1}, for $1 < m$.

Method II. This method solves the (linear least-squares) problem:

$$\operatorname*{Min}_{z\in\mathbb{R}^m} \left\| \bar{H}_m z_m - \|r_1\| e_m \right\| \tag{10}$$

to compute z_m. The matrix $\bar{H}_m$ is obtained from the matrix $\bar{T}_m$ by permuting rows so that the first row becomes last (see Proposition 2 for a similar derivation of H_m). It is checked using (4) that

$$\operatorname*{Min}_{z\in\mathbb{R}^m} \|r_{m+1}\| \leqslant \|V_{m+1}^* V_{m+1}\|^{1/2} \operatorname*{Min}_{z\in\mathbb{R}^m} \left\| \bar{H}_m z_m - \|r_1\| e_m \right\|.$$

When the matrix A is symmetric, this method is the Paige and Saunders' MINRES. This method will be called Biorthogonal Lanczos Minimal Residual method (BiLMINRES). For A symmetric, the factor $\|V_{m+1}^* V_{m+1}\|$ equals 1. However, for A unsymmetric, this factor may be very large. Another shortcoming of this method for A unsymmetric is that r_{m+1} is not biorthogonal to W_m. This method can viewed as a special case of the QMR method (by Freund and Nachtigal) without lookahead [10].

Corollary 3. *Let us assume that $w_1 = v_1 = r_1/\|r_1\|$ in BiLMINRES or BiLQR, where r_1 is the initial residual vector. The methods BiLMINRES or BiLQR do not break down and provide the approximate solution x_{m+1} if and only if* (i) *in Proposition* 1 *holds.*

Proof. From Propositions 1 and 2 and the definition of the methods. □

For comparison we mention that to compute x_{m+1} the biconjugate gradients method and CGS in addition to (i) of Proposition 1 the following additional m non-breakdown conditions must be satisfied:

(ii) $\det(M_k') \neq 0$, $1 \leqslant k \leqslant m$,

where M_k' are the moment matrices of dimension k with entries $m_{ij}' = (A^{i+j-1}v_1, w_1)$.

This result was proved in [18]. The TFQMR method [9] (by Freund) has the same non-breakdown conditions as CGS.

4. The squared biorthogonal Lanczos method for eigenvalues

In this section we derive the squared biorthogonal Lanczos method (SBiL) by squaring the biorthogonal Lanczos method matrix polynomials and obtaining a simple recurrence equation for generating them. This derivation was first presented in [5]. We will use characters with hat to denote the polynomials (in variables A or A^*) which, if applied to v_1, yield the corresponding biorthogonal Lanczos vectors.

Notation. In the biorthogonal Lanczos method, let $\hat{v}_i$ and $\hat{w}_i$ be the polynomials of degree i such that $v_i = \hat{v}_i(A)v_1$ and $w_i = \hat{w}_i(A^*)w_1$.

Remark 4. In the biorthogonal Lanczos method the vectors w_i are used only in determining the parameters α_i and β_i.

The parameters in the biorthogonal Lanczos method can be expressed in terms of products of the matrix polynomials $\hat{v}_i$ and $\hat{w}_i$ in the matrix A only. To see this, we write

$$\alpha_i = (A\hat{v}_i(A)v_1, \hat{w}_i(A^*)w_1) = (A\hat{w}_i(A)\hat{v}_i(A)v_1, w_1) \tag{11}$$

and

$$\gamma_i = \beta_i\delta_i = (\hat{v}_i(A)v_1, \hat{w}_i(A^*)w_1) = (\hat{w}_i(A)\hat{v}_i(A)v_1, w_1). \tag{12}$$

Therefore we must find a recursion to compute the polynomials $\hat{v}_i(A)\hat{w}_i(A)$ and $A\hat{v}_i(A)\hat{w}_i(A)$. We note that these polynomials do not depend on the order of their factors. For example, $\hat{v}_i(A)\hat{w}_i(A) = \hat{w}_i(A)\hat{v}_i(A)$.

Multiplication of the polynomials $\hat{t}_{i+1}$ and $\hat{s}_{i+1}$ from (3) and (4) of Algorithm 1 yield

$$\hat{t}_{i+1}\hat{s}_{i+1} = A[(A\hat{v}_i\hat{w}_i - 2\alpha_i\hat{v}_i\hat{w}_i) - (\beta_i\hat{v}_{i-1}\hat{w}_i + \delta_i\hat{v}_i\hat{w}_{i-1})] + \alpha_i^2\hat{v}_i\hat{w}_i \tag{13}$$

$$+ \alpha_i(\beta_i\hat{v}_{i-1}\hat{w}_i + \delta_i\hat{v}_i\hat{w}_{i-1}) + \beta_i\delta_i\hat{v}_{i-1}\hat{w}_{i-1}. \tag{14}$$

In order to be able to compute $\hat{t}_{i+1}\hat{s}_{i+1}$ recursively, we need to compute recursively $\hat{s}_{i+1}\hat{v}_i = \beta_{i+1}\hat{v}_i\hat{w}_{i+1}$ and $\hat{t}_{i+1}\hat{w}_i = \delta_{i+1}\hat{v}_{i+1}\hat{w}_i$. From (3) and (4) of Algorithm 1 we obtain

$$\hat{s}_{i+1}\hat{v}_i = (A\hat{v}_i\hat{w}_i - \alpha_i\hat{v}_i\hat{w}_i) - \delta_i\hat{v}_i\hat{w}_{i-1} = (A\hat{v}_i\hat{w}_i - \alpha_i\hat{v}_i\hat{w}_i) - \hat{t}_i\hat{w}_{i-1}, \tag{15}$$

$$\hat{t}_{i+1}\hat{w}_i = (A\hat{v}_i\hat{w}_i - \alpha_i\hat{v}_i\hat{w}_i) - \beta_i\hat{v}_{i-1}\hat{w}_i = (A\hat{v}_i\hat{w}_i - \alpha_i\hat{v}_i\hat{w}_i) - \hat{s}_i\hat{v}_{i-1}. \tag{16}$$

It can be easily checked by induction that $\hat{s}_{i+1}\hat{v}_i = \hat{t}_{i+1}\hat{w}_i$.

We set $\hat{u}_{i+1} = \hat{t}_{i+1}\hat{s}_{i+1}/\gamma_{i+1}$ and $\hat{p}_{i+1} = \hat{t}_{i+1}\hat{w}_i = \hat{v}_{i+1}\hat{w}_i$. Then we obtain the simplified expressions

$$\hat{u}_{i+1} = A^2\hat{u}_i - 2\alpha_i A\hat{u}_i - 2A\hat{p}_i + \alpha_i{}^2\hat{u}_i + 2\alpha_i\hat{p}_i + \gamma_i\hat{u}_{i-1} \tag{17}$$

and

$$\hat{p}_{i+1} = A\hat{u}_i - \alpha_i\hat{u}_i - \hat{p}_i. \tag{18}$$

We next present the SBiL method in matrix polynomial form. We first need the following notation.

Notation. The inner product $[\hat{v}, \hat{w}]$ of the matrix polynomials (in A) $\hat{v}$ and $\hat{w}$ stands for the inner product $(\hat{v}(A)v_1, \hat{w}(A^*)w_1)$.

Algorithm 2 (*The Squared Biorthogonal Lanczos Method* (*SBiL*)).
$\gamma_1 = 0$, $\hat{u}_0 = 0$ and $\hat{p}_1 = \hat{u}_1 = 1$
For $i = 1, \ldots, m$ **do**
(1) Compute $A\hat{u}_i$
(2) $\alpha_i = [1, A\hat{u}_i]$
(3) $\hat{y}_i = A\hat{u}_i - \alpha_i\hat{u}_i$
(4) Compute $A^2\hat{u}_i$
(5) $A\hat{y}_i = A^2\hat{u}_i - \alpha_i A\hat{u}_i$

(6) $\hat{u}_{i+1} = A\hat{y}_i - \alpha_i\hat{y}_i - 2A\hat{p}_i + 2\alpha_i\hat{p}_i + \gamma_i\hat{u}_{i-1}$
(7) $\gamma_{i+1} = [1, \hat{u}_{i+1}]$
(8) $\hat{u}_{i+1} = \hat{u}_{i+1}/\gamma_{i+1}$
(9) $\hat{p}_{i+1} = \hat{y}_i - \hat{p}_i$
(10) $A\hat{p}_{i+1} = A\hat{y}_i - A\hat{p}_i$
EndFor

Let the characters without hat represent vectors which equal the matrix polynomials applied to v_1 (e.g., $u_i = \hat{u}_i(A)v_1$). Then the vector form of this algorithm can be obtained by removing the hat and replacing the *unit* polynomial by v_1. This method requires computational work equal to (19N Ops + 2Mv) per iteration. Let $T_m = \text{tridiag}[\delta_{i+1}, \alpha_i, \beta_{i+1}]$, where α_i, γ_i are computed in (2) and (7) (respectively) of Algorithm 2 and $\delta_{i+1} = |\gamma_{i+1}|^{1/2}$ and $\beta_{i+1} = \delta_{i+1}\,\text{sign}(\gamma_{i+1})$. Algorithms 1 and 2 compute the same matrix T_m, which can be used to approximate the extreme eigenvalues of A. So, Algorithm 2 is a matrix transpose free unsymmetric Lanczos for eigenvalues.

Let $v_1 = w_1$; then the polynomials $\hat{v}_i$ and $\hat{w}_i$ are equal up to a sign.

Remark 5. Let us assume that Algorithm 1 does not break down and $v_1 = w_1$. Then the polynomials $\hat{v}_i$ and $\hat{w}_i$ satisfy the equalities $\hat{v}_i = (-1)^{i_s}\hat{w}_i$ where i_s is number of sign changes in the sequence β_j for $1 \leqslant j \leqslant i$. To see this, let $\delta_i = \pm\beta_i$, $\hat{w}_{i-1} = (-1)^{i_s-1}\hat{v}_{i-1}$ and $\hat{w}_i = (-1)^{i_s}\hat{v}_i = \pm(-1)^{i_s-1}$. Then $\hat{s}_{i+1} = (A\hat{w}_i - \alpha_i\hat{w}_i) - \delta_i\hat{w}_{i-1} = (-1)^{i_s}[A\hat{v}_i - \alpha_i\hat{v}_i - \beta_i\hat{v}_{i-1}] = (-1)^{i_s}\hat{t}_{i+1}$. Now it follows that $\hat{w}_{i+1} = (-1)^{(i+1)_s}\hat{v}_{i+1}$.

If $v_1 = w_1$, Remark 5 allows us to compute $\hat{v}_i^2$ and $\hat{v}_i\hat{v}_{i-1}$ from Algorithm 2 for little extra work.

Remark 6. We apply Remark 5 to $\hat{u}_i = \hat{v}_i\hat{w}_i$ and $\hat{p}_i = \hat{v}_i\hat{w}_{i-1}$ to compute $\hat{v}_i^2 = (-1)^{i_s}\hat{u}_i$ and $\hat{v}_i\hat{v}_{i-1} = (-1)^{i_s}\hat{p}_i/\beta_i$, respectively. To achieve this, we need a vector of size m to keep track of the occurrences of negative signs in the sequence of β_i, for $i = 1, \ldots, m$.

In the following section we use Algorithm 2 as part of a squared Lanczos for linear systems.

5. The restarted squared Lanczos method for linear systems

In this section we present a squared form of the restarted BiLMINRES and BiLQR methods of *cycle* (consisting of m iterations) which will be called SBiMINRES(m) and SBiLQR(m), respectively. We need the following remark.

Remark 7. Assume that $x_1 = 0$ and $\|r_1\| = 1$. To achieve this, one redefines (1) to become $Ax = \tilde{b}$ where $\tilde{b} = b - Ax_1$; then, scaling this system gives $\|r_1\| = 1$.

Remark 7 implies that (in the BiLMINRES and BiLQR methods) $v_1 = r_1$. Also, from (3) the solution $x_{m+1} = V_m z_m$, where the Lanczos vectors $V_m = [v_1, \ldots, v_m]$ are of the form $\hat{v}_i(A)v_1$, for $i = 1, \ldots, m$. So, the solution x_{m+1} is of the form $\hat{x}_{m+1}(A)v_1$, for a matrix polynomial $\hat{x}_{m+1}$.

We will derive the recurrences for the matrix polynomial form of the algorithms for a complete cycle. Then the vector form of the algorithms can be obtained by removing the hat and replacing the

unit polynomial by v_1. We will use the vector notation wherever it is more concise and it does not lead to confusion.

Notation. The residual and solution vectors in SBiMINRES(m) and SBiLQR(m) (after m iterations) will be denoted by r^s_{m+1} and x^s_{m+1}, respectively.

Let us now consider the residual vector r_{m+1} (in BiLMINRES and BiLQR) updated in a recursive form $q_{i+1} = q_i - z^i_m A v_i$, for $i = 1, \ldots, m$, where $q_1 = r_1$, $r_{m+1} = q_{m+1}$ and z_m is the solution of (8) or (10). The vectors q_i are *intermediate residual* vectors but they do not have the properties of the residual vectors r_{m+1} generated by BiLMINRES or BiLQR. The recursion for the *intermediate residual* matrix polynomials becomes

$$\hat{q}_{i+1} = \hat{q}_i - z^i_m A \hat{v}_i. \tag{19}$$

We next derive a recurrence for the *squared intermediate residual polynomials* $\hat{q}^2_i$, which will be denoted by $\hat{R}_i$, for $i = 1, \ldots, m+1$, where $r^s_{m+1} = R_{m+1}$. We then derive the corresponding *intermediate solution polynomials* $\hat{X}_i$, for $i = 1, \ldots, m+1$, where $x^s_1 = X_1$ and $x^s_{m+1} = X_{m+1}$. Squaring the intermediate residual polynomials in (19), we obtain the squared intermediate residual polynomials

$$\hat{R}_{i+1} = \hat{R}_i - 2z^i_m A\hat{q}_i\hat{v}_i + (z^i_m)^2 A^2 \hat{v}^2_i, \tag{20}$$

for $i = 1, \ldots, m$, where $A\hat{q}_i\hat{v}_i$ is computed by direct matrix times vector multiplication and (from (19) and Algorithm 1) we obtain

$$\hat{q}_i\hat{v}_i = \hat{q}_{i-1}\hat{v}_i - z^{i-1}_m A\hat{v}_i\hat{v}_{i-1}, \tag{21}$$

where

$$\hat{q}_{i-1}\hat{v}_i = \frac{1}{\delta_i}[A\hat{q}_{i-1}\hat{v}_{i-1} - \alpha_{i-1}\hat{q}_{i-1}\hat{v}_{i-1} - \beta_{i-1}\hat{q}_{i-1}\hat{v}_{i-2}] \tag{22}$$

and

$$\hat{q}_{i-1}\hat{v}_{i-2} = \hat{q}_{i-2}\hat{v}_{i-2} - z^{i-2}_m A\hat{v}^2_{i-2}. \tag{23}$$

Now, to derive the solution polynomials for SBiMINRES(m) or SBiLQR(m), we use (20) and the fact that $R_i = b - AX_i$, for $i = 1, \ldots, m$,

$$\hat{X}_{i+1} = \hat{X}_i + 2z^i_m\hat{q}_i\hat{v}_i - (z^i_m)^2 A\hat{v}^2_i. \tag{24}$$

So the solution vector is $x^s_{m+1} = X_{m+1}$.

We use Remark 6 to show that $A\hat{v}^2_i = (-1)^{i_s} A\hat{u}_i$, $A^2\hat{v}^2_i = (-1)^{i_s} A^2\hat{u}_i$ and $A\hat{v}_i\hat{v}_{i-1} = (-1)^{i_s} A\hat{p}_i/\beta_i$. The polynomials $A\hat{u}_i$, $A^2\hat{u}_i$ and $A\hat{p}_i$ are computed in Algorithm 2. We will use $\hat{f}_i$, $\hat{g}_{i-1}$ and $\hat{h}_{i-2}$ to denote $\hat{q}_i\hat{v}_i$, $\hat{q}_{i-1}\hat{v}_i$ and $\hat{q}_{i-1}\hat{v}_{i-2}$, respectively. We summarize the defining equations (20)–(24) (using matrix polynomials) for one complete *cycle* (of m iterations) of SBiMINRES(m) or SBiLQR(m) in the following algorithm.

Algorithm 3 (SBiMINRES(m) *or* SBiLQR(m)).
Compute $A\hat{u}_i$, $A^2\hat{u}_i$, $A\hat{p}_i$ and α_i, β_i, δ_i in Algorithm 2 for $i = 1, \ldots, m$

Compute z_m in (8) (or (10))
Set $\hat{f}_1 = 1$, $\hat{X}_1 = 0$ and $\hat{R}_1 = 1$
For $i = 1, \ldots, m$ **do**
(1) If $(i \leqslant 1)$ goto (5)
(2) If $(2 < i)$: $\hat{h}_{i-2} = \hat{f}_{i-2} - z_m^{i-2}(-1)^{(i-2)_s} A\hat{u}_{i-2}$
(3) $\hat{g}_{i-1} = 1/\delta_i[A\hat{f}_{i-1} - \alpha_{i-1}\hat{f}_{i-1} - \beta_{i-1}\hat{h}_{i-2}]$
(4) $\hat{f}_i = \hat{g}_{i-1} - z_m^{i-1}(-1)^{i_s} A\hat{p}_i/\beta_i$
(5) Compute $A\hat{f}_i$
(6) $\hat{R}_{i+1} = \hat{R}_i - 2z_m^i A\hat{f}_i + (-1)^{i_s}(z_m^i)^2 A^2\hat{u}_i$
(7) $\hat{X}_{i+1} = \hat{X}_i + 2z_m^i\hat{f}_i - (-1)^{i_s}(z_m^i)^2 A\hat{u}_i$
EndFor

The vector form of Algorithm 3 can be obtained by removing the hat and replacing the unit polynomial 1 by v_1. For SBiLQR(m) equation (6) of Algorithm 3 is deleted because the residuals can be computed in a simpler way described in the following remark.

Remark 8. In SBiLQR(m), $\hat{R}_{m+1}$ can be computed from $\hat{u}_m$ and $\hat{u}_{m+1}$. For approach (a) by squaring the matrix polynomials in (6) and using Remark 6, we obtain

$$\hat{R}_{m+1} = \sigma_m^2(-1)^{(m+1)_s}\hat{u}_{m+1}. \tag{25}$$

For approach (b) by squaring the matrix polynomials in (9) and using Remark 7, we obtain

$$\hat{R}_{m+1} = \tilde{\sigma}_m^2(-1)^{m_s}\hat{u}_m + 2\tilde{\sigma}_m\sigma_m\frac{(-1)^{m_s}}{\beta_m}\hat{p}_m + \sigma_m^2(-1)^{(m+1)_s}\hat{u}_{m+1}. \tag{26}$$

The computational cost for SBiMINRES(m) and SBiLQR(m) requires ($19N$ Ops + 2 Mv) per iteration for the Algorithm 2 part. The computation of R_m and X_m requires an extra ($17N$ Ops + 1 Mv) per iteration for SBiMINRES(m) and an extra $((13 + 4/m)N$ Ops + 1 Mv) per iteration for SBiLQR(m) (using (26)). So the total work equals ($36N$ Ops + 3 Mv) per iteration for SBiMINRES(m) and $((32 + 4/m)N$ Ops + 3 Mv) per iteration for SBiLQR(m). It also requires to keep in secondary storage the vectors Au_i, A^2u_i, Ap_i in Algorithm 2 for $i = 1, \ldots, m$ until they are used in Algorithm 3.

Remark 8 allows us to monitor the size of $\|\hat{R}_{m+1}(A)v_1\|$ for little additional work and select dynamically the cycle size for restarting. We implemented SBiLQR to dynamically select the size of a cycle with a maximum allowable size m_0.

The *modified* SBiLQR method is the following implementation of the restarted SBiLQR with varying cycle size. The residual norm is monitored and the cycle ends by applying the following criteria.

(1) Either the norm of the residual (given by (26) for some $m = 1, \ldots, m_0 + 1$) is smaller than the residual norm of the preceding cycle;

(2) or after m_0 steps of Algorithm 3.

In case (2), let m_1: $\|\hat{R}_{m_1}(A)v_1\| = \mathrm{Min}_{2\leqslant m\leqslant m_0+1}\|\hat{R}_m(A)v_1\|$. The solution is computed (in cases (1) and (2)) based on SBiLQR(m_1).

This implementation leads to less oscillatory behavior of the residual error norm and faster convergence.

6. Numerical tests

We have discretized two boundary value problems in partial differential equations on a square region by the method of finite differences. The first problem is a standard elliptic problem which can be found in [19] and the right-hand side function is constructed so that the analytic solution is known. The second problem is taken from [22].

Problem I.

$$-(\rho\psi_{\xi_1})_{\xi_1} - (\sigma\psi_{\xi_2})_{\xi_2} + (\tau\psi)_{\xi_1} + (\zeta\psi)_{\xi_2} + \phi\psi = \chi, \quad \Omega = (0,1) \times (0,1),$$

where

$$\rho(\xi_1,\xi_2) = e^{-\xi_1\xi_2}, \qquad \sigma(\xi_1,\xi_2) = e^{\xi_1\xi_2},$$
$$\tau(\xi_1,\xi_2) = \tilde{\beta} * (\xi_1 + \xi_2), \qquad \zeta(\xi_1,\xi_2) = \tilde{\gamma} * (\xi_1 + \xi_2),$$
$$\phi(\xi_1,\xi_2) = \frac{1}{(1+\xi_1\xi_2)}, \qquad \psi(\xi_1,\xi_2) = \xi_1 e^{\xi_1\xi_2} \sin(\pi\xi_1)\sin(\pi\xi_2),$$

with Dirichlet boundary condition and $\chi(\xi_1,\xi_2)$ the corresponding right-hand side function. By controlling $\tilde{\gamma}$ and $\tilde{\beta}$, we could change the degree of nonsymmetry. We chose $\tilde{\gamma} = 50.0$, $\tilde{\beta} = 1.0$.

Problem II.

$$-(\psi_{\xi_1\xi_1} + \psi_{\xi_2\xi_2}) + \eta\psi_{\xi_1} + \tfrac{1}{2}\eta_{\xi_2}\psi = 0, \quad \Omega = (0,1) \times (0,1),$$

where

$$\eta = \eta(\xi_1,\xi_2) = 20\exp(3.5(\xi_1^2 + \xi_2^2)).$$

We have used the five-point difference operator for the Laplacian, central difference scheme for the first derivative. We placed 200 uniform grid points in each dimension. This yielded unsymmetric (nonsingular) linear systems of 40,000 equations. The initial guess is $x^s = 0$ and the stopping criterion was $\|r_{m+1}^s\| < \epsilon$ with $\epsilon = 10^{-7}$. We have used the standard Incomplete LU preconditioning (ILU(0)). We run SBiLQR(m), SBiMINRES(m) and GMRES(m) all with $m = 15$. We also run *modified* SBiLQR with maximum allowed *cycle* size $m = 15$. We plotted the logarithm (with base 10) of the residual norm versus the number of iterations in Figs. 1 and 2. It is clear from the figures that the GMRES gives a smooth curve and curves for SBiLQR and SBiMINRES oscillate.

In terms of work, GMRES(m) requires $((2m+3m+2/m)N\,\text{Ops} + (1+1/m)\,\text{Mv})$ per iteration while SBiLQR(m) and SBiMINRES(m) require $(36N\,\text{Ops} + 3\,\text{Mv})$ and $((32+4/m)N\,\text{Ops} + 3\,\text{Mv})$ per iteration, respectively. SBiLQR(m) and SBiMINRES(m) require more storage than GMRES(m). The SBiLQR method seems to perform the best in terms of number of iterations and overall work. The modified SBiLQR gives a smoother curve than SBiLQR(15) and SBiMINRES(15). In the *modified* SBiLQR the number of additional iterations of Algorithm 2 that were performed but not used for the solution (in Algorithm 3) were 22 in Problem I and 15 in Problem II. Note that iterations of Algorithm 2 are less expensive than iterations of Algorithm 3 (steps (1)–(8)).

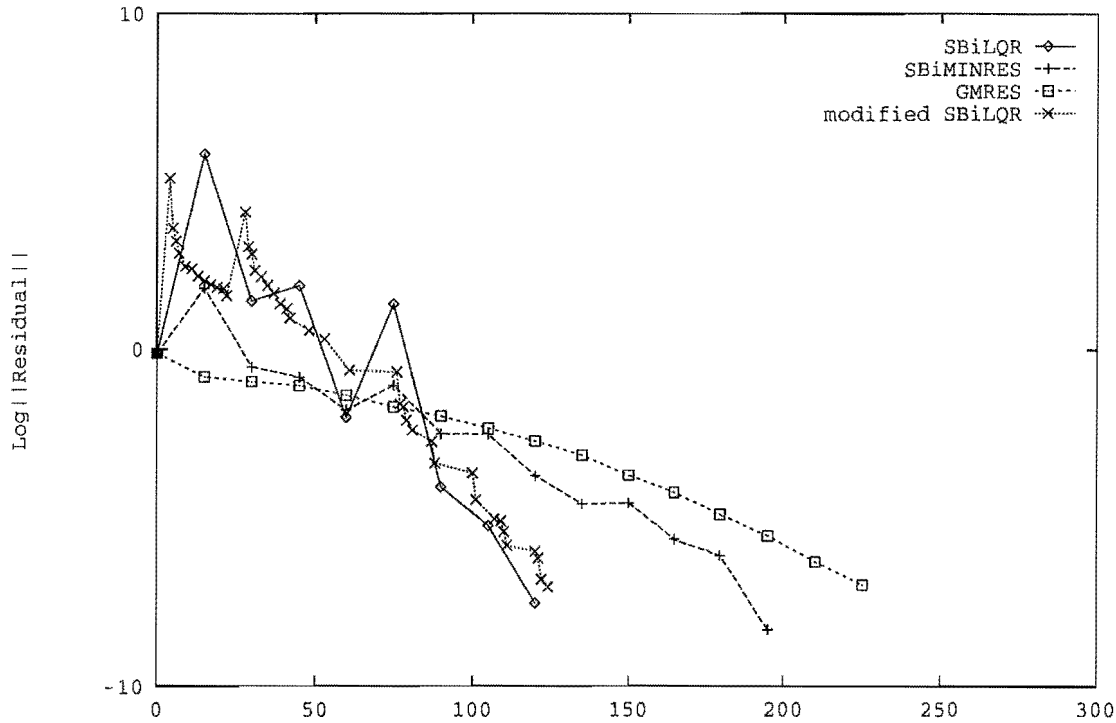

Fig. 1. Problem I, iterations.

7. Conclusions

We derived the squared Lanczos method for eigenvalues of unsymmetric matrices. The squared Lanczos method forms the same tridiagonal matrix T_m as the biorthogonal Lanczos method. The need for multiplication by the matrix transpose has been eliminated. We derived robust biorthogonal Lanczos methods for linear systems. We then obtained restarted squared forms of these methods. We compared the new squared methods to the restarted Generalized Minimal Residual Method (GMRES). The residual norms in the new methods are initially very large and they oscillate. This is expected for two reasons. First, the biorthogonal Lanczos type methods do not minimize the residual norm. Second, if we assume residual norm minimization (e.g., in symmetric problems), the matrix polynomials of squared biorthogonal Lanczos methods are not always of spectral radius smaller than one (especially in the beginning of a cycle). This results in very high residual norms in the first few iterations. The modified SBiLQR exhibits smoother reduction of the residual norms as the iteration proceeds compared to the rest of the squared biorthogonal Lanczos methods. For restarted (orthogonal) Krylov subspace methods all direction vectors being formed are being used in every step and must be stored in the main memory. However, for the squared restarted biorthogonal, only a fixed small number of the direction vectors must be stored in the main memory. All the direction vectors are needed only when the approximate solution is computed.

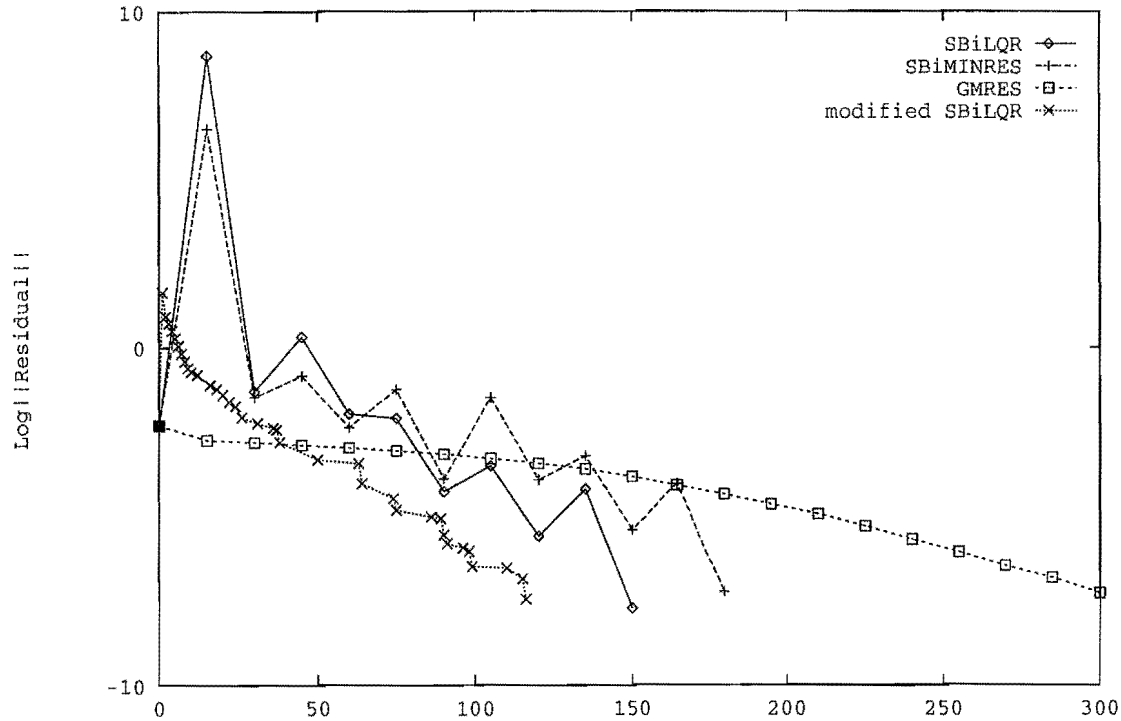

Fig. 2. Problem II, iterations.

Acknowledgements

The author expresses his appreciation to the referees for their helpful suggestions.

References

[1] O. Axelsson, A generalized conjugate gradient, least square method, *Numer. Math.* **51** (1987) 209–227.
[2] T.F. Chan, L. De Pillis and H. van der Vorst, A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems, Tech. Report CAM 91-17, Univ. California, Los Angeles, CA, 1991.
[3] A.T. Chronopoulos, Krylov subspace iterative methods for nonsymmetric indefinite linear systems, Tech. Report 90-21, Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, 1990.
[4] A.T. Chronopoulos, *s*-step iterative methods for (non)symmetric (in)definite linear systems, *SIAM J. Numer. Anal.* **28** (6) (1991) 1776–1789.
[5] A.T. Chronopoulos and S. Ma, On squaring Krylov subspace iterative methods for nonsymmetric linear systems, Tech. Report 89-67, Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, 1989.
[6] 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.
[7] V. Faber and T.A. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, *SIAM J. Numer. Anal.* **21** (2) (1984) 352–362.
[8] R. Fletcher, *Conjugate Gradient Methods for Indefinite Systems*, Lecture Notes in Math. **506** (Springer, New York, 1976).
[9] R.W. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, *SIAM J. Sci. Statist. Comput.* **14** (1993) 470–482.

[10] R.W. Freund and N.M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems, *Numer. Math.* **60** (1991) 315–339.
[11] 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.
[12] W.D. Joubert, Generalized gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations, Report CNA-238, Univ. Texas, Austin, TX, 1990.
[13] S.K. Kim and A.T. Chronopoulos, An efficient nonsymmetric Lanczos method on parallel vector computers, *J. Comput. Appl. Math.* **42** (3) (1992) 357–374.
[14] C. Lanczos, An iteration for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Nat. Bureau Standards* **45** (1950) 255–282.
[15] S. Ma and A.T. Chronopoulos, Implementation of iterative methods for large sparse nonsymmetric linear systems on parallel vector computers, *Internat. J. Supercomput.* **4** (4) (1990) 9–24.
[16] C.C. Paige and M.A. Saunders, Solution of sparse indefinite systems of linear equations, *SIAM J. Numer. Anal.* **12** (1975) 617–624.
[17] B.N. Parlett, D.R. Taylor and Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, *Math. Comp.* **44** (169) (1985) 105–124.
[18] Y. Saad, The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems, *SIAM J. Numer. Anal.* **19** (3) (1982) 485–506.
[19] Y. Saad and M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **7** (1986) 856–869.
[20] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric systems, *SIAM Sci. Statist. Comput.* **10** (1989) 36–52.
[21] H. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13** (2) (1992) 631–644.
[22] O. Widlund, A Lanczos method for a class of nonsymmetric systems of linear equations, *SIAM J. Numer. Anal.* **15** (1978) 801–812.
[23] D.M. Young and K.C. Jea, Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods, *Linear Algebra Appl.* **34** (1980) 159–194.