Computers and Mathematics with Applications 60 (2010) 873–889

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A thick-restarted block Arnoldi algorithm with modified Ritz vectors for large eigenproblems

Wei Jiang, Gang Wu *

School of Mathematical Sciences, Xuzhou Normal University, Xuzhou, 221116, Jiangsu, PR China

ARTICLE INFO

Article history:
Received 20 May 2009
Received in revised form 21 May 2010
Accepted 21 May 2010

Keywords:
Block Arnoldi
Krylov subspace
Thick-restarting
Subspace iteration
Ritz vector
Modified Ritz vector

ABSTRACT

The block Arnoldi method is one of the most commonly used techniques for large eigenproblems. In this paper, we exploit certain modified Ritz vectors to take the place of Ritz vectors in the thick-restarted block Arnoldi algorithm, and propose a modified thick-restarted block Arnoldi algorithm for large eigenproblems. We then consider how to periodically combine the refined subspace iterative method with the modified thick-restarting block Arnoldi algorithm for computing a few dominant eigenpairs of a large matrix. The resulting algorithm is called a Subspace-Block Arnoldi algorithm. Numerical experiments show the efficiency of our new algorithms.

1. Introduction

Suppose we wish to solve the large eigenproblem

$$Ax = \lambda x,$$

where A is an $n \times n$ real matrix, and (λ, x) is referred to as an eigenpair of A with $\|x\|_2 = 1$, here the norm used is the Euclidean norm.

Large scale eigenproblems arise in many applications [1–4], such as computational fluid dynamics, electrical engineering, oceanography, and quantum chemistry, and so on. Arnoldi's method [5,6] and block Arnoldi's method [7,8] are commonly used techniques for solving them.

Block Krylov methods have their own merits compared with the non-block ones [9,7]. For instance, block methods are more suitable to deal with multiple or clustered eigenproblems. Another advantage of a block method over a non-block one is the better use of cache [9,7]. Other merits include the use of Level 3 BLAS matrix–matrix products for fast algorithms. These make the block eigensolvers remain important throughout the development of numerical linear algebra [10,7,11–14, 8,15,16,4,17].

Let V_1 be an n by p orthonormal block vectors, then the m-step block Arnoldi process constructs an orthonormal basis $\mathscr{U}_{m+1} = [V_1, \ldots, V_m, V_{m+1}]$ for the block Krylov subspace

$$\mathscr{K}_{m+1}(A, V_1) = span\{V_1, AV_1, A^2V_1, \ldots, A^mV_1\}.$$

The following relations hold for the block Arnoldi process [3,4]

$$A\mathscr{U}_m = \mathscr{U}_m\mathscr{H}_{mp} + V_{m+1}H_{m+1,m}E_m^H = \mathscr{U}_{m+1}\overline{\mathscr{H}}_{mp}, \tag{1.1}$$

* Corresponding author.
E-mail addresses: veijiang@yahoo.com.cn (W. Jiang), gangwu76@yahoo.com.cn, wugangzy@gmail.com (G. Wu).

0898-1221/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2010.05.034

where $\overline{\mathscr{H}}_{mp}$ is an $(mp+p)\times mp$ band upper-Hessenberg matrix with $p\times p$ matrices H_{ij} as its elements, and E_m is an $mp\times p$ zero matrix except for its last p rows being an identity matrix. The following algorithm provides essential details of the block Arnoldi process, for more details, refer to [3,4].

Algorithm 1 (*The Block Arnoldi Process (with A. Ruhe's Variant)*)**.**

1. Start: Given m, the steps of the block Arnoldi process; p, the block size; and an initial orthonormal block vector V_1 of size n by p;
2. Iterate:

for $j = p, p+1, \ldots, mp-1$
  $k = j-p+1$;
  $w = Av_k$;
  for $i = 1, 2, \ldots, j$
    $h_{i,k} = (w, v_i)$;
    $w = w - h_{i,k}v_i$;
  end
  $h_{j+1,k} = \|w\|_2$; if $h_{j+1,j} = 0$, then stop;
  $v_{j+1} = w/h_{j+1,k}$;
end

Let $(\tilde{\lambda}_i, \tilde{g}_i),\ i = 1, 2, \ldots, mp$, be the eigenpairs of $\mathscr{H}_{mp}$, the block Arnoldi method uses $\tilde{\lambda}_i$ and $\tilde{x}_i = \mathscr{V}_m\tilde{g}_i$ to approximate some eigenpairs of A [3,4]. Here $\tilde{\lambda}_i$ and $\tilde{x}_i$ are called Ritz values and Ritz vectors of A with respect to the block Krylov subspace $\mathscr{K}_m(A, V_1)$, respectively.

However, as the steps of the block Arnoldi process increases, the block Arnoldi method will become impractical because of memory and computational requirements. One remedy is to use restarting strategies [18,19]. Recently, Baglama [7] developed an augmented block Householder Arnoldi (ABHA) method that combines the advantages of a block routine and augmented routine. The foundation of the ABHA method is the use of the Householder process to create an orthonormal basis for the block Krylov subspace. An advantage of the representation is the heavy use of Level 3 BLAS matrix–matrix operations.

In this paper, we focus on the thick-restarting strategy proposed recently [20,13,18,21,22]. We first apply the thick-restarting strategy to the block Arnoldi algorithm, and introduce a thick-restarted block Arnoldi algorithm for large eigenproblems. Unfortunately, the convergence analysis given in [23] indicates that the Ritz vectors obtained by the block Arnoldi method may not be good approximations to the desired eigenvectors. Therefore, it is necessary to seek new approximations to take the place of Ritz vectors and improve the performance of the thick-restarted block Arnoldi algorithm. Therefore, in Section 2 we exploit the modified Ritz vectors [24,23,25,26] to take the place of the Ritz vectors in the thick-restarted block Arnoldi algorithm, and propose a modified thick-restarted block Arnoldi algorithm for large eigenproblems. The rationality of the new algorithm is also discussed.

The subspace iteration is a classical method for computing a few dominant eigenvalues and the associated eigenvectors of a large sparse matrix [3]. In [27], Jia combined the refined strategy [12] and an updating scheme with the subspace iteration, and proposed a refined subspace iteration method. Motivated by the idea proposed in [28], in Section 3 we knit the refined subspace iteration together with the modified thick-restarted block Arnoldi algorithm proposed in Section 2, and present a Subspace-Block Arnoldi algorithm for computing a few dominant eigenpairs of large, sparse matrices. Numerical experiments given in Section 4 illustrate numerical behavior of our new algorithms.

Some notations are listed below. Throughout this paper, we denote by $\mathscr{K}_{m+1}(A, V_1)$ the block Krylov subspace $span\{V_1, AV_1, \ldots, A^mV_1\}$, by l the number of desired eigenpairs, by $\mathbb{C}^p$ the complex space of dimension p, and by "^{H}" the conjugate transpose of a matrix or a vector. We denote by $\sigma_{\max}(X)$ and $\sigma_{\min}(X)$ the largest and the smallest singular value of the matrix X, respectively. Let I be the identity matrix whose order is clear from the context.

2. A modified thick-restarted block Arnoldi algorithm for large eigenproblems

In this section, we first apply the thick-restarting strategy [18,21] to the block Arnoldi algorithm, and present a thick-restarted block Arnoldi algorithm. We then introduce the modified Ritz vector [24,23,25,26], and show why it is superior to the regular Ritz vector. Finally, we apply the modified Ritz vector to the thick-restarted block Arnoldi algorithm, and propose a modified thick-restarted block Arnoldi algorithm for large nonsymmetric eigenproblems. Some properties of the new algorithm are also discussed.

2.1. A thick-restarted block Arnoldi algorithm for large eigenproblems

The thick-restarted Arnoldi algorithm [18,29] is mathematically equivalent to the famous implicitly restarted Arnoldi advocated by Sorensen when exact shifts are used [18,19]. Generally, however, it is not the case for the block Arnoldi method. The thick-restarted Arnoldi algorithm is a little bit similar to the Arnoldi-E method [30], except the approximate eigenvectors are put firstly. Some remarkable merits of this approach are that the entire subspace is still a Krylov subspace, and it includes

smaller Krylov subspaces with Ritz vectors as starting vectors [30,20]. Furthermore, k_0 matrix-vector multiplications can be saved before restarting, where k_0 is the number of Ritz vectors from the previous iteration. For more details, we refer to [20,13,18,21] and the references given therein.

In this subsection, we introduce a thick-restarted block Arnoldi algorithm for large eigenproblems, which is a straightforward generalization of the thick-restarted Arnoldi algorithm. It can also be viewed as an exact analogy of the thick-restarted harmonic block Arnoldi algorithm due to Morgan [13]. The algorithm is described as follows.

Algorithm 2 (*A Thick-restarted Block Arnoldi Algorithm for Large Eigenproblems*)**.**

1. Start: Choose m, the step of block Arnoldi procedure; p, the block size; k_0, the number of approximate eigenvectors which are retained from one cycle to the next; also l, the desired number of eigenpairs (with $k_0 \geq l$). Choose an initial $n \times p$ column orthonormal block vector V_1, as well as a prescribed tolerance tol;
2. The first iteration: Run Algorithm 1 for the computation of $\mathscr{U}_{m+1} = [V_1, \ldots, V_m, V_{m+1}]$ and $\mathscr{H}_{mp}$. Compute all the eigenpairs $(\tilde{\lambda}_i, \tilde{g}_i)$, $i = 1, \ldots, mp$, of $\mathscr{H}_{mp}$, and select l of them as the desired ones; goto **Step** 5;
3. Apply the block Arnoldi procedure from the current point, to complete the computation of new block vectors of $\mathscr{U}_{m+1}$ as well as the new entries of the $(mp + p) \times mp$ matrix $\overline{\mathscr{H}}_{mp}$, where the current point is V_{k_0+p};
4. Compute all the eigenpairs $(\tilde{\lambda}_i, \tilde{g}_i)$, $i = 1, \ldots, mp$, of $\mathscr{H}_{mp}$, and select l of them as the desired ones;
5. If all the desired eigenpairs are accurate enough, then stop, else continue;
6. Orthonormalization of k_0 short vectors: Orthonormalize the $\tilde{g}_i's$, for $i = 1, \ldots, k_0$. Separate $\tilde{g}_i$ into real part and imaginary part if it is complex, in order to form a real $mp \times k_0$ matrix $P_{k_0} = [\tilde{p}_1, \ldots, \tilde{p}_{k_0}]$. Both parts of complex vectors need to be included, so temporarily reduce k_0 by 1 if necessary (or k_0 can be increased by 1);
7. Orthonormalization of the $k_0 + p$ short vectors: Extend $\tilde{p}_1, \ldots, \tilde{p}_{k_0}$ to a length $(mp + p) \times k_0$ matrix $\widehat{P}_{k_0}$ by appending a $p \times k_0$ zeros matrix, and set $P_{k_0+1} = [\widehat{P}_{k_0}, E_{m+1}]$, where E_{m+1} is an $(mp + p) \times p$ zero matrix except for its last p rows being an identity matrix. Note that P_{k_0+1} is an $(mp + p) \times (k_0 + p)$ matrix;
8. Form portions of new $\overline{\mathscr{H}}_{mp}$ and $\mathscr{U}_{m+1}$ using the old $\overline{\mathscr{H}}_{mp}$ and $\mathscr{U}_{m+1}$: Let $\overline{\mathscr{H}}_{k_0}^{new} = P_{k_0+1}^H \overline{\mathscr{H}}_{mp} P_{k_0}$, and $V_{k_0+p}^{new} = \mathscr{U}_{m+1} P_{k_0+1}$, then set $\overline{\mathscr{H}}_{k_0} = \overline{\mathscr{H}}_{k_0}^{new}$ and $V_{k_0+p} = V_{k_0+p}^{new}$; goto **Step** 3.

At each cycle after the first (where the first cycle is the ordinarily block Arnoldi iteration), some recurrences similar to (1.1) are generated by the thick-restarted block Arnoldi algorithm.

Theorem 2.1. *Under the above notation, the following relation holds for the thick-restarted block Arnoldi algorithm*

$$A\mathscr{U}_m = \mathscr{U}_m \mathscr{H}_{mp} + V_{m+1} H_{m+1,m} E_m^H. \tag{2.1}$$

Where $\mathscr{H}_{mp}$ is a block upper-Hessenberg matrix as before, except for its leading $(k_0 + p) \times (k_0 + p)$ portion is a full matrix, and E_m is an $mp \times p$ zero matrix except for its last p rows being an identity matrix.

Proof. The proof is similar to that of Theorem 3.1 in [28]. □

Let $(\tilde{\lambda}_i, \tilde{g}_i)$ be an eigenpair of $\mathscr{H}_{mp}$. Then the thick-restarted block Arnoldi method makes use of the Ritz pair $(\tilde{\lambda}_i, \tilde{x}_i)$ as an approximation to an eigenpair of A [3,4], where $\tilde{x}_i = \mathscr{U}_m \tilde{g}_i$ is called a Ritz vector. Defining the residual $\tilde{r}_i = A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i$, we obtain from (2.1) that

$$\tilde{r}_i = A\mathscr{U}_m \tilde{g}_i - \tilde{\lambda}_i \mathscr{U}_m \tilde{g}_i = V_{m+1} H_{m+1,m} E_m^H \tilde{g}_i,$$

and

$$\|\tilde{r}_i\|_2 = \|H_{m+1,m} E_m^H \tilde{g}_i\|_2,$$

which can be used as a cheap stopping criterion for convergence.

2.2. The modified Ritz vector

It has been shown that Ritz vectors may converge very slowly or even fail to do so, even if the Ritz values do [31]. In this subsection, we introduce the modified Ritz vector proposed recently [24,23,25,26,32], and show why the modified Ritz vector is better than the regular Ritz vector in general.

Recall that the m-step block Arnoldi process generates an orthonormal basis for the block Krylov subspace $\mathscr{K}_{m+1}(A, V_1) = span\{V_1, V_2, \ldots, V_m, V_{m+1}\}$. Given a Ritz pair $(\tilde{\lambda}_i, \tilde{x}_i)$, the modified Ritz vector $x_i^M \in span\{\tilde{x}_i, V_{m+1}\}$ satisfies the following optimal property [25,26]

$$\|(A - \tilde{\lambda}_i I) x_i^M\|_2 = \min_{\substack{\alpha_i \in \mathbb{C}, \eta_i \in \mathbb{C}^p \\ |\alpha_i|^2 + \|\eta_i\|_2^2 = 1}} \|(A - \tilde{\lambda}_i I)(\alpha_i \tilde{x}_i + V_{m+1}\eta_i)\|_2. \tag{2.2}$$

Let $\begin{bmatrix} \alpha_i \\ \eta_i \end{bmatrix}$ be the right singular vector corresponding to smallest singular value of $\left[\tilde{r}_i,\ (A - \tilde{\lambda}_i I) V_{m+1}\right]$, and denote by

$$z_i^M = \begin{bmatrix} \alpha_i \tilde{g}_i \\ \eta_i^M \end{bmatrix}, \tag{2.3}$$

then the modified Ritz vector corresponding to $\tilde{\lambda}_i$ is [25,26]

$$x_i^M = \mathscr{U}_{m+1} z_i^M. \tag{2.4}$$

The residual vector corresponding to the modified Ritz pair $(\tilde{\lambda}_i, x_i^M)$ is

$$\begin{aligned} r_i^M &= (A - \tilde{\lambda}_i I)\mathscr{U}_{m+1} z_i^M, \\ &= (A - \tilde{\lambda}_i I)[\mathscr{U}_m, V_{m+1}] \begin{bmatrix} \alpha_i \tilde{g}_i \\ \eta_i \end{bmatrix}, \\ &= \alpha_i V_{m+1} H_{m+1,m}(E_m^H \tilde{g}_i) + (A - \tilde{\lambda}_i I) V_{m+1} \eta_i. \end{aligned}$$

It is seen from (2.2) that

$$\|r_i^M\|_2 \le \|\tilde{r}_i\|_2, \tag{2.5}$$

so the modified Ritz vector x_i^M is at least as good as the Ritz vector $\tilde{x}_i$.

Next we aim to give a quantitive description of why the modified Ritz vector is better than the Ritz vector. For simplicity, we neglect the subscripts here.

Theorem 2.2. *Denote by $\mathscr{B}$ the subspace spanned by the columns of $B = (A - \tilde{\lambda}_i I)V_{m+1}$, and by w the left singular vector corresponding to smallest singular value of $C = [\tilde{r},\ B]$. Suppose that B is of full rank, let $\angle(\tilde{r}, \mathscr{B}) = \arcsin \|(I - BB^\dagger)\tilde{r}\|_2 / \|\tilde{r}\|_2$ be the acute angle between $\tilde{r}$ and $\mathscr{B}$, and let $\angle(w, \mathscr{B}) = \arcsin \|(I - BB^\dagger)w\|_2$ be the acute angle between w and $\mathscr{B}$, then*

$$\frac{\|r^M\|_2}{\|\tilde{r}\|_2} = \frac{|\alpha| \sin \angle(\tilde{r}, \mathscr{B})}{\sin \angle(w, \mathscr{B})}, \tag{2.6}$$

where $(\cdot)^\dagger$ denotes the pseudo-inverse of a given matrix.

Proof. Since w is the left singular vector associated with $\sigma_{\min}(C)$, we have

$$\begin{bmatrix} \tilde{r},\ B \end{bmatrix} \begin{bmatrix} \alpha \\ \eta^M \end{bmatrix} = \sigma_{\min}(C) \cdot w.$$

That is,

$$B\eta^M = -\alpha\tilde{r} + \sigma_{\min}(C) \cdot w, \tag{2.7}$$

and

$$\eta^M = -\alpha B^\dagger \tilde{r} + \sigma_{\min}(C) \cdot B^\dagger w. \tag{2.8}$$

Therefore,

$$B\eta^M = -\alpha BB^\dagger \tilde{r} + \sigma_{\min}(C) \cdot BB^\dagger w. \tag{2.9}$$

Equating (2.7) and (2.9) gives

$$\alpha(I - BB^\dagger)\tilde{r} = \sigma_{\min}(C) \cdot (I - BB^\dagger)w.$$

Thus,

$$|\alpha| \cdot \|\tilde{r}\|_2 \cdot \frac{\|(I - BB^\dagger)\tilde{r}\|_2}{\|\tilde{r}\|_2} = \sigma_{\min}(C) \cdot \|(I - BB^\dagger)w\|_2,$$

which yields

$$|\alpha| \cdot \|\tilde{r}\|_2 \cdot \sin \angle(\tilde{r}, \mathscr{B}) = \|r^M\|_2 \cdot \sin \angle(w, \mathscr{B}),$$

so we complete the proof. □

However, it seems that (2.6) is unsatisfactory in quantifying how much the quality of the Ritz vector can be improved. We next show that $\sin \angle(w, \mathscr{B}) \to 1$ as $\|r^M\|_2 \to 0$. To this end, we first need a useful lemma.

Lemma 2.3 (*[26]*)**.** *Denote by $\mathcal{W} = span\{\tilde{x},\ V_{m+1}\}$, then the modified Ritz vector x^M and the residual norm $\|r^M\|_2$ satisfy the classical orthogonal projection*

$$\begin{cases} x^M \in \mathcal{W}, \\ (A - \tilde{\lambda}I)^H (A - \tilde{\lambda}I)x^M - \|r^M\|_2^2 \cdot x^M \perp \mathcal{W}. \end{cases}$$

The following result shows that the speed of $\sin \angle(w, \mathscr{B}) \to 1$ will be much faster than that of $\|r^M\|_2 \to 0$ during iterations.

Theorem 2.4. *Under the above notation, there holds*

$$\cos\angle(w,\mathscr{B}) \le \|r^M\|_2^2 \cdot \left[\|B^\dagger\|_2^2 \cdot \left(1+\frac{1}{\sin\angle(\tilde{r},\mathscr{B})}\right)\right]. \tag{2.10}$$

Proof. Note that

$$\begin{aligned}\|B^\dagger\|_2 &= \|(B^HB)^{-1}B^H\|_2\\ &= \left\|\left[(B^HB)^{-1}B^H\right]^H\right\|_2 = \|B(B^HB)^{-1}\|_2,\end{aligned}$$

so we have

$$\begin{aligned}\cos\angle(w,\mathscr{B}) &= \|BB^\dagger w\|_2\\ &= \|B(B^HB)^{-1}B^Hw\|_2\\ &\le \|B(B^HB)^{-1}\|_2\cdot\|B^Hw\|_2 = \|B^\dagger\|_2\cdot\|B^Hw\|_2.\end{aligned} \tag{2.11}$$

Since w is the left singular vector corresponding to the smallest singular value of $C=[\tilde{r},\ B]$, it follows from (2.7) that

$$\begin{aligned}B^Hw &= B^H\cdot\frac{\alpha\tilde{r}+B\eta^M}{\sigma_{\min}(C)}\\ &= B^H\frac{(A-\tilde{\lambda}I)x^M}{\sigma_{\min}(C)}\\ &= \frac{V_{m+1}^H(A-\tilde{\lambda}I)^H(A-\tilde{\lambda}I)x^M}{\sigma_{\min}(C)}.\end{aligned} \tag{2.12}$$

Moreover, we have from Lemma 2.3 that

$$(A-\tilde{\lambda}I)^H(A-\tilde{\lambda}I)x^M - \|r^M\|_2^2\cdot x^M \perp \mathcal{W},$$

i.e.,

$$\begin{bmatrix}\tilde{x}^H\\ V_{m+1}^H\end{bmatrix}\left[(A-\tilde{\lambda}I)^H(A-\tilde{\lambda}I)x^M - \|r^M\|_2^2\cdot x^M\right] = O.$$

So we obtain

$$\begin{aligned}V_{m+1}^H(A-\tilde{\lambda}I)^H(A-\tilde{\lambda}I)x^M &= \|r^M\|_2^2\cdot V_{m+1}^Hx^M\\ &= \|r^M\|_2^2\cdot\eta^M,\end{aligned} \tag{2.13}$$

where we used $V_{m+1}^H\tilde{x}=0$ and $V_{m+1}^HV_{m+1}=I$. From (2.12) and (2.13), we get $B^Hw=\|r^M\|_2\cdot\eta^M$, and

$$\|B^Hw\|_2 = \|r^M\|_2\cdot\|\eta^M\|_2, \tag{2.14}$$

where we used $\sigma_{\min}(C)=\|r^M\|_2$. Furthermore, it is seen from (2.8) that

$$\begin{aligned}\eta^M &= \|r^M\|_2\cdot\left[B^\dagger w - \frac{\alpha B^\dagger\tilde{r}}{\|r^M\|_2}\right]\\ &= \|r^M\|_2\cdot B^\dagger\left[w-\frac{\alpha\tilde{r}}{\|r^M\|_2}\right].\end{aligned}$$

As a result,

$$\begin{aligned}\|\eta^M\|_2 &\le \|r^M\|_2\cdot\|B^\dagger\|_2\cdot\left[1+|\alpha|\frac{\|\tilde{r}\|_2}{\|r^M\|_2}\right]\\ &\le \|r^M\|_2\cdot\|B^\dagger\|_2\cdot\left[1+\frac{1}{\sin\angle(\tilde{r},\mathscr{B})}\right].\end{aligned} \tag{2.15}$$

where we used the relation from (2.6) that $\frac{\|\tilde{r}\|_2}{\|r^M\|_2} \le \frac{1}{|\alpha|\sin\angle(\tilde{r},\mathscr{B})}$. So we obtain (2.10) from combining (2.11), (2.14) and (2.15). □

We point out that if $\tilde{\lambda}$ is not an eigenvalue of A and $\mathscr{U}_{m+1}$ is of full rank, then $\sin\angle\left(\tilde{r},\mathscr{B}\right)\neq 0$. Indeed, suppose that $\sin\angle\left(\tilde{r},\mathscr{B}\right)=0$, then $\tilde{r}\in\mathscr{B}$, and there exists a vector $u\in\mathbb{C}^p$ such that $\tilde{r}=(A-\tilde{\lambda}I)V_{m+1}u$. Thus $(A-\tilde{\lambda}I)(\mathscr{U}_m\tilde{g}-V_{m+1}u)=O$,

which implies that

$$(A-\tilde{\lambda}I)\mathscr{U}_{m+1}\begin{bmatrix}\tilde{g}\\ -u\end{bmatrix}=O.$$

Since $\tilde{\lambda}$ is not an eigenvalue of A, $A-\tilde{\lambda}I$ is nonsingular; moreover, if $\mathscr{U}_{m+1}$ is of full rank, then $\tilde{g}=O$, which is a contradiction.

Theorem 2.4 indicates that the speed with which $\cos\angle(w,\mathscr{B})$ tends to zero will be much faster than that with which $\|r^M\|_2$ does so, provided B is not too ill-conditioned and $\sin\angle(\tilde{r},\mathscr{B})$ is not too small. Therefore, as the convergence proceeds, we have $\|r^M\|_2/\|\tilde{r}\|_2\to|\alpha|\cdot\sin\angle(\tilde{r},\mathscr{B})\le|\alpha|\le 1$. That is, $\|r^M\|_2$ can be (much) smaller than $\|\tilde{r}\|_2$, which explains why the modified Ritz vector x^M is better than the Ritz vector $\tilde{x}$ in some degree.

2.3. A new algorithm for large eigenproblems

As was mentioned before, Ritz vectors may not be good approximations since they may converge very slowly or even fail to do so [31]. In order to partially circumvent this difficulty, we make use of the modified Ritz vector to take the place of the Ritz vector in the thick-restarted block Arnoldi algorithm, and propose a modified thick-restarted block Arnoldi algorithm for large nonsymmetric eigenproblems. The main algorithm of this paper is described as follows.

Algorithm 3 (*A Modified Thick-restarted Block Arnoldi Algorithm for Large Eigenproblems*)**.**

1. Start: Choose m, the step of block Arnoldi procedure; p, the block size; k_0, the number of approximate eigenvectors which are retained from one cycle to the next; also l, the desired number of eigenpairs (with $k_0\ge l$). Given an initial $n\times p$ column orthonormal block vector V_1, as well as a prescribed tolerance tol;
2. The first iteration: Run Algorithm 1 for the computation of $\mathscr{U}_{m+1}=[V_1,\ldots,V_m,V_{m+1}]$ and $\mathscr{H}_{mp}$. Compute all the eigenpairs $(\tilde{\lambda}_i,\tilde{g}_i)$, $i=1,\ldots,mp$, of $\mathscr{H}_{mp}$. Then take $x_i^M=W_iz_i$ as approximations to x_i, $i=1,\ldots,l$, where z_i are computed as follows [28]:

$$\begin{aligned}&Z=AV_{m+1};\\ &Z_1=Z^HZ;\qquad Z_2=Z^HV_{m+1};\\ &\textbf{for } i=1,2,\ldots,l\\ &\quad Z_3=H_{m+1,m}\tilde{g}_i(mp-p+1:mp);\\ &\quad S=(V_{m+1}Z_3)^HZ-\tilde{\lambda}_iZ_3^H;\\ &\quad T=Z_1-2\,real(\tilde{\lambda}_i)Z_2+|\tilde{\lambda}_i|^2I;\\ &\quad \tilde{C}=\begin{bmatrix}\|Z_3\|_2^2 & S\\ S^H & T\end{bmatrix},\\ &\textbf{end}\end{aligned}$$

   and z_i, $i=1,\ldots,l$, are the eigenvectors corresponding to the smallest eigenvalues of the Hermitian matrix $\tilde{C}$ [25,26]. Select $(\tilde{\lambda}_i,x_i^M)$, $i=1,\ldots,l$ as the desired eigenpairs; goto **Step** 5;
3. Apply the block Arnoldi procedure from the current point, to complete the computation of new block vectors of $\mathscr{U}_{m+1}$ as well as the new entries of the $(mp+p)\times mp$ matrix $\overline{\mathscr{H}}_{mp}$, where the current point is V_{k_0+2p};
4. Compute all the eigenpairs $(\tilde{\lambda}_i,\tilde{g}_i)$, $i=1,\ldots,mp$, of $\mathscr{H}_{mp}$, and take $x_i^M=W_iz_i$ $i=1,\ldots,l$, as approximations to x_i, $i=1,\ldots,l$ (see **Step** 2);
5. Check convergence: Compute $\|r_i^M\|_2=\sqrt{\lambda_{\min}(\tilde{C})}$, $i=1,\ldots,l$, where $\lambda_{\min}(\tilde{C})$ is the smallest eigenvalue of $\tilde{C}$. If they are all satisfied with the given accuracy tol, then stop, else continue;
6. Orthonormalization of the k_0 short vectors: Orthonormalize the $\tilde{g}_i's$, for $i=1,\ldots,k_0$. Separate $\tilde{g}_i$ into real part and imaginary part if it is complex, in order to form a real $mp\times k_0$ matrix $P_{k_0}=[p_1,p_2,\ldots,p_{k_0}]$. Both parts of complex vectors need to be included, so temporarily reduce k_0 by 1 if necessary (or k_0 can be increased by 1);
7. Orthonormalization of the k_0+p short vectors: Extend $p_1,p_2,\ldots,p_{k_0}$ to a length $(mp+p)\times k_0$ matrix $\widehat{P}_{k_0}$ by appending a $p\times k_0$ zeros matrix, and set $P_{k_0+1}=[\widehat{P}_{k_0},E_{m+1}]$, where E_{m+1} is an $(mp+p)\times p$ zero matrix except for its last p rows being an identity matrix. Note that P_{k_0+1} is an $(mp+p)\times(k_0+p)$ matrix;
8. Form portions of new $\overline{\mathscr{H}}_{mp}$ and $\mathscr{U}_{m+1}$ using old $\overline{\mathscr{H}}_{mp}$ and $\mathscr{U}_{m+1}$: Let $\overline{\mathscr{H}}_{k_0}^{new}=P_{k_0+1}^H\overline{\mathscr{H}}_{mp}P_{k_0}$, and $V_{k_0+p}^{new}=\mathscr{U}_{m+1}P_{k_0+1}$, then set $\overline{\mathscr{H}}_{k_0}=\overline{\mathscr{H}}_{k_0}^{new}$ and $V_{k_0+p}=V_{k_0+p}^{new}$;
9. Orthonormalizing AV_{m+1} with respect to $V_{k_0+p}^{new}$, which yields an $n\times p$ block vector v_{k_0+2p}, and set $V_{k_0+2p}=V_{k_0+2p}^{new}=[V_{k_0+p}^{new},\ v_{k_0+2p}]$, then compute $\overline{\mathscr{H}}_{k_0+p}^{new}=(V_{k_0+2p}^{new})^HAV_{k_0+p}^{new}$; go to **Step** 3.

Now we consider some properties of the new algorithm. The following theorem establishes a block Arnoldi-like relation for Algorithm 3.

Theorem 2.5. *At each iteration, it holds that*

$$A\mathscr{U}_m=\mathscr{U}_m\mathscr{H}_{mp}+V_{m+1}H_{m+1,m}E_m^H. \tag{2.16}$$

Proof. At each iteration after the first, it is sufficient to prove

$$AV_{k_0+p}^{new} = V_{k_0+2p}^{new}\overline{\mathscr{H}}_{k_0+p}^{new}, \tag{2.17}$$

where $V_{k_0+p}^{new}$ is an n by k_0+p matrix, $V_{k_0+2p}^{new}$ is an n by k_0+2p matrix, and $\overline{\mathscr{H}}_{k_0+p}^{new}$ is a k_0+2p by k_0+p matrix. Note that $x_i^M \in span\{\tilde{x}_i, V_{m+1}\}$, and $Ax_i^M \in span\{\tilde{x}_i, V_{m+1}, AV_{m+1}\}$, thus

$$A[x_1^M, x_2^M, \ldots, x_{k_0}^M] \subseteq span\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{k_0}, V_{m+1}, AV_{m+1}\}.$$

It follows from Algorithm 3 that

$$span\{V_{k_0+p}^{new}\} = span\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{k_0}, V_{m+1}\},$$

and

$$AV_{k_0+p}^{new} \subseteq span\{V_{k_0+p}^{new}, AV_{m+1}\} = span\{V_{k_0+2p}^{new}\}.$$

So there exists a $(k_0+2p)\times(k_0+p)$ matrix $\overline{\mathscr{H}}_{k_0+p}^{new}$, such that

$$AV_{k_0+p}^{new} = V_{k_0+2p}^{new}\overline{\mathscr{H}}_{k_0+p}^{new},$$

which completes the proof. □

Now let's consider how to compute $\overline{\mathscr{H}}_{k_0+p}^{new}$ efficiently in practice. At the first glance, it seems that one has to pay $(k_0+2p)\times(k_0+p)$ inner products for computing $\overline{\mathscr{H}}_{k_0+p}^{new}$, c.f. Step 9 of Algorithm 3. Indeed, as we will show, the computation of the projection matrix only requires $(k_0+2p)\times p$ inner products. By (2.17),

$$\begin{aligned}\overline{\mathscr{H}}_{k_0+p}^{new} &= \begin{bmatrix}(V_{k_0+p}^{new})^H \\ v_{k_0+2p}^H\end{bmatrix} A \begin{bmatrix}V_{k_0}^{new} & V_{m+1}\end{bmatrix}, \\ &= \begin{bmatrix}(V_{k_0+p}^{new})^H AV_{k_0}^{new} & (V_{k_0+p}^{new})^H AV_{m+1} \\ v_{k_0+2p}^H AV_{k_0}^{new} & v_{k_0+2p}^H AV_{m+1}\end{bmatrix}.\end{aligned}$$

It follows from Theorem 2.1 that

$$(V_{k_0+p}^{new})^H AV_{k_0}^{new} = \overline{\mathscr{H}}_{k_0}^{new},$$

and

$$v_{k_0+2p}^H AV_{k_0}^{new} = v_{k_0+2p}^H V_{k_0+p}^{new}\overline{\mathscr{H}}_{k_0}^{new} = O.$$

Therefore,

$$\overline{\mathscr{H}}_{k_0+p}^{new} = \begin{bmatrix}\overline{\mathscr{H}}_{k_0}^{new} & (V_{k_0+p}^{new})^H AV_{m+1} \\ O & v_{k_0+2p}^H AV_{m+1}\end{bmatrix}.$$

Recall that both $\overline{\mathscr{H}}_{k_0}^{new}$ and AV_{m+1} are already available. Therefore, one only needs to form $(V_{k_0+p}^{new})^H AV_{m+1}$ and $v_{k_0+2p}^H AV_{m+1}$, which requires $(k_0+2p)\times p$ inner products.

The following theorem indicates that the search subspace generated by Algorithm 3 after restarting contains smaller Krylov subspaces with each of the desired Ritz vectors as the starting vector. Note that the dimension of the search subspace is mp.

Theorem 2.6. *Let k_0 be a multiple of p, and let $q = \frac{mp-k_0}{p}$. Define*

$$\begin{aligned}&\mathscr{S} = span\{x_1^M, x_2^M, \ldots, x_{k_0}^M, V_{m+1}, AV_{m+1}, \ldots, A^{q-1}V_{m+1}\},\\ &\mathscr{S}_i = span\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{k_0}, A\tilde{x}_i, \ldots, A^q\tilde{x}_i\}, \quad 1 \le i \le k_0.\end{aligned}$$

Then

$$\mathscr{S}_i \subseteq \mathscr{S} = span\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{k_0}, V_{m+1}, AV_{m+1}, \ldots, A^{q-1}V_{m+1}\}, \quad 1 \le i \le k_0.$$

Proof. Notice that

$$\begin{aligned}\mathscr{S} &= span\{x_1^M, x_2^M, \ldots, x_{k_0}^M, V_{m+1}, AV_{m+1}, \ldots, A^{q-1}V_{m+1}\},\\ &\subseteq span\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{k_0}, V_{m+1}, AV_{m+1}, \ldots, A^{q-1}V_{m+1}\}.\end{aligned}$$

Moreover, it is obvious to see that the two subspaces are the same, since the dimension of the two subspaces is equal.

Let $\mathcal{S}_i = span\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{k_0}, A\tilde{x}_i, \ldots, A^{q+1}\tilde{x}_i\}$, we next show that $\mathcal{S}_i \subseteq \mathcal{S}$, $1 \le i \le k_0$. Indeed,

$$A\tilde{x}_i = \tilde{\lambda}_i\tilde{x}_i + V_{m+1}H_{m+1,m}E_m^H\tilde{g}_i \in span\{\tilde{x}_i, V_{m+1}\}, \quad 1 \le i \le k_0.$$

That is, $A\tilde{x}_i$ is a linear combination of $\tilde{x}_i$ and V_{m+1}. Similarly,

$$\begin{aligned} A^2\tilde{x}_i &= \tilde{\lambda}_i A\tilde{x}_i + AV_{m+1}H_{m+1,m}E_m^H\tilde{g}_i, \\ &= \tilde{\lambda}_i^2\tilde{x}_i + \tilde{\lambda}_i V_{m+1}H_{m+1,m}E_m^H\tilde{g}_i + AV_{m+1}H_{m+1,m}E_m^H\tilde{g}_i, \\ &\in span\{\tilde{x}_i, V_{m+1}, AV_{m+1}\}. \end{aligned}$$

Therefore, $A^2\tilde{x}_i$, $1 \le i \le k_0$, is a linear combination of $\tilde{x}_i$, V_{m+1} and AV_{m+1}. Using the same trick, for any $1 \le i \le k_0$, we can show that $A^j\tilde{x}_i$, $1 \le j \le q$, is a linear combination of $\tilde{x}_i$, V_{m+1}, AV_{m+1}, $\ldots$, $A^{j-1}V_{m+1}$, which completes the proof. □

3. A Subspace-Block Arnoldi algorithm for computing a few dominant eigenpairs of large matrices

Subspace or simultaneous iterations are very popular for computing a few dominant eigenvalues and the associated eigenvectors of a large sparse matrix. However, it has been shown that even if the distance between the wanted eigenvector and the projection subspace approaches zero, the Ritz vector may converge erratically or even may not converge [31]. In order to circumvent this difficulty, Jia [27] proposed a refined subspace iterative algorithm. In this section, we knit the refined subspace iterative algorithm together with the modified thick-restarted Arnoldi algorithm, and propose a Subspace-Block Arnoldi algorithm for computing a few dominant eigenpairs of large matrices. The hybrid algorithm can be viewed as a generalization of the Power–Arnoldi algorithm proposed recently [28].

3.1. The subspace iteration and the refined subspace iterative algorithm

Subspace iteration can be viewed as a block generalization of the classical power method [33]. Although the method is not competitive with some projection methods, it is still one of the most important methods used in structural engineering [3].

Given an $n \times p$ orthonormal block vector $X_0 = [x_1, x_2, \ldots, x_p]$, the subspace iterative algorithm amounts to computing the matrix $X_k = A^kX_0$ for a certain power k. Label the eigenvalues of A in the decreasing order of their module

$$|\lambda_1| \ge |\lambda_2| \ge \cdots \ge |\lambda_p| > |\lambda_{p+1}| \ge \cdots \ge |\lambda_n|. \tag{3.1}$$

Then under some mild assumption on X_0, as k increase, the subspace $span\{A^kX_0\}$ tends to the subspace spanned by the associated eigenvectors $x_1, \ldots, x_p$ of A; see, e.g., [3]. In [27], Jia further investigated the subspace iterative algorithm, and derived a more efficient refined subspace iteration algorithm for computing a few dominant eigenpairs of large matrices. The improvements are twofold: First, the refined strategy was used to extract refined Ritz vectors from a given subspace. Second, a novel strategy was exploited to construct a better initial block vector when restarting.

The refined subspace iterative algorithm is outlined as follows, for more details on how to compute the refined Ritz vectors in Step 3, as well as to update the initial block vector in Step 4, refer to [27].

Algorithm 4 (*The Refined Subspace Iterative Algorithm*)**.**

1. Start: Given the number l of the wanted dominant eigenpairs; pick the subspace dimension p with $p \ge l$. Choose an initial $n \times p$ column orthonormal block X, and a prescribed tolerance tol;
2. Compute $X := AX$;
3. Rayleigh–Ritz projection and the refined strategy: Make a QR decomposition to get $X := XR$ and form $B = X^HAX$. Then calculate the refined Ritz vectors u_i, $i = 1, 2, \ldots, l$, corresponding to the l dominant eigenvalues of B;
4. Convergence test: If the l dominant refined approximate eigenpairs satisfy a prescribed accuracy, then stop; otherwise, update X and go to Step 2.

3.2. A Subspace-Block Arnlodi algorithm

Motivated by the Power–Arnoldi algorithm proposed in [28], we combine the refined subspace iterative algorithm with the modified thick-restarted block Arnoldi algorithm, and propose a hybrid algorithm for computing a few dominant eigenvalues as well as the corresponding invariant subspace of a large, nonsymmetric matrix. The resulting algorithm is called the Subspace-Block Arnoldi algorithm.

The principle of the hybrid algorithm is described as follows. Given an n by p column orthonormal block vector X_0, we first run the refined subspace iteration several times to get a rough convergence. If the accuracy is not satisfied with the prescribed tolerance *tol*, then we use the resulting block vector as the initial guess, and run the modified thick-restarted block Arnoldi algorithm several times, to get another approximation. If the new approximation is still unsatisfactory, return to the refined subspace algorithm again, using the approximation obtained from the block Arnoldi as the initial guess. Proceed the above procedure analogously until convergence.

The key of the hybrid algorithm is when and how to terminate the refined subspace iteration and trigger the modified thick-restarted block Arnoldi algorithm. We exploit the following strategy; for more details, refer to [28]. In the new algorithm, three parameters β, *maxit1* and *maxit2* are used to flip-flop between the refined subspace iteration and the modified thick-restarted block Arnoldi algorithm. Denote by $r^{(curr)}$ the residual norm of the current iteration, and by $r^{(prev)}$ that of the previous iteration. We check whether ratio $= r^{(curr)}/r^{(prev)}$ is greater than β, say, 0.5. If so, set *restart* := *restart* +1 and examine whether *restart* is larger than the pre-determined number *maxit1*, say, 5. If so, terminate the refined subspace iteration and run the modified thick-restarted block Arnoldi algorithm. In summary, we can present the Subspace-Block Arnoldi algorithm.

Algorithm 5 (*A Subspace-Block Arnoldi Algorithm for Large Eigenproblems*)**.**

1. Start: Choose m, the steps of block Arnoldi procedure; p, the block size, which is also the refined subspace dimension; and k_0, the number of approximate eigenvectors which are retained from one cycle to the next; l, the number of desired eigenpairs, with $l \leq p$. Choose an initial $n \times p$ column orthonormal block X_0, and a prescribed tolerance tol, three parameters β, maxit1 and maxit2 that are used to flip flop between the refined subspace iteration and the thick-restarted modified block Arnoldi algorithm;
2. Run the refined subspace iteration:

restar $t = 0$; $r = 1$;
while restart $<$ maxit1 & $r > tol$
(2.1) $r_0 := r$; % residual norm of the "previous" iteration
(2.2) Run Algorithm 4, **if** $r \leq tol$, **then** stop;
(2.3) **if** $r/r_0 > \beta$ % the convergence rate
(2.4) restart := restart+1;
(2.5) **end if**
end while

3. Run Algorithm 3 with X_0 as the initial guess, where X_0 is the approximation obtained from the refined subspace iteration:

restart := 0,
while restart $<$ maxit2 & $r > tol$
(3.1) $r_0 := r$; % residual norm of the "previous" iteration
(3.2) Run Algorithm 3, **if** $r \leq tol$, **then** stop;
(3.3) **if** $r/r_0 > \beta$ % the convergence rate
(3.4) restart := restart+1;
(3.5) **end if**
end while

Go to **Step** 2, and run the refined subspace iteration with $X_0 = V_{k+p}^{new}(:, 1:p)$ as the initial guess.

Heuristically, we can interpret the mechanism of Algorithm 5 as follows. As the iteration proceeds, the refined subspace iteration can purify the initial block vector for the block Arnoldi algorithm. On the other hand, the block Arnoldi algorithm can provide a better initial guess for the refined subspace algorithm. Consequently, a periodic combination of Algorithms 3 and 4 can improve the performance of the two original algorithms; see the numerical experiments made in Section 4. One is also recommended to see [28] for theoretical analysis of the hybrid strategy.

Finally, we point out that it is difficult to determine the *optimal* parameters such as β, *maxit1* and *maxit2* for a general matrix, and the answer is problem-dependent [28]. Fortunately, we find experimentally that the performance of the Subspace-Block Arnoldi algorithm seems insensitive to the chosen parameters.

4. Numerical experiments

In this section, we report some numerical experiments and illustrate the numerical behavior of the new algorithms. All the algorithms are run using Matlab 7.0 on a 1.6 GHz dual core Intel(R) Pentium(R) processor with 1 GB main memory. To make a fair and reasonable comparison, in each example, the same block vector is generated randomly in a uniform distribution, orthogonalized and utilized as the initial guess. The algorithms will be stopped as soon as [27]

$$\frac{\max\limits_{1\leq i\leq l} \|A\widehat{x}_i - \widehat{\lambda}_i\widehat{x}_i\|_2}{\|A\|_F} \leq tol, \tag{4.1}$$

where $(\widehat{\lambda}_i, \widehat{x}_i)$ are approximate eigenpairs, *tol* is a user described tolerance, and $\|A\|_F$ is the Frobenius norm of A, which is needed to be computed only once and stored for later use. In order to show numerical behavior of Algorithm 3 and Algorithm 5, we compare them with Algorithm 2, Baglama's augmented block Householder Arnoldi algorithm [10] (ABHA, whose MATLAB file is available from http://www.math.uri.edu/~jbaglama/), the block Krylov–Schur algorithm due to Zhou and Saad [17], as well as Jia's refined subspace algorithm [24].

Table 1
Numerical results of Example 1.

p	m	k_0	ABHA		Algorithm 2		Algorithm 3		**Eff** (%)
			time	mv	*time*	mv	*time*	mv	
	10	12	34.9	4 754	29.7	3 836	10.1	3 102	19.1
2	8	6	41.8	11 026	17.1	4 896	9.95	3 878	20.8
	4	4	46.8	19 648	27.6	12 436	23.1	9 478	23.8
	6	8	33.8	15 204	21.8	8 098	14.1	5 451	32.7
3	5	6	74.9	36 930	53.0	21 248	45.8	16 142	24.0
	4	4	67.2	46 800	47.4	18 372	33.9	13 513	26.5
	4	12	45.0	18 810	30.3	9 620	23.3	6 937	27.9
5	3	8	81.1	36 370	23.3	11 383	18.9	8 532	22.7
	2	4	85.5	48 160	61.0	27 004	50.4	22 449	16.9

Example 1. Numerical results of Algorithms 2 and 3, and Baglama's algorithm for computing the two rightmost eigenpairs of the *BWM2000* matrix, $tol = 1e{-}6$.

Table 2
Numerical results of Example 1.

p	m	k_0	ABHA		Algorithm 2		Algorithm 3		**Eff** (%)
			time	mv	*time*	mv	*time*	mv	
	10	8	1.03	448	0.57	320	0.54	298	6.88
2	8	6	1.63	994	0.67	556	0.59	528	5.04
	5	6	2.98	1726	2.06	1262	1.35	1188	5.86
	6	8	0.45	444	0.41	438	0.38	401	8.45
3	5	7	2.92	1689	0.81	655	0.76	634	3.21
	4	5	0.93	1131	0.87	943	0.80	876	7.10
	4	8	0.83	790	0.73	560	0.67	517	7.68
5	3	6	0.94	930	0.86	861	0.78	794	7.78

Example 1. Numerical results of Algorithms 2 and 3, and Baglama's algorithm for computing the four leftmost eigenpairs of the *Sherman4* matrix, $tol = 1e{-}6$.

In all the tables, we denote by *time* the CPU time used in seconds, and by mv the number of matrix-vector products required for convergence. We denote by *ABHA* the augmented block householder Arnoldi algorithm due to Baglama [10], by Algorithm 2 the thick-restarted block Arnoldi algorithm, and by Algorithm 3 the thick-restarted block Arnoldi algorithm with modified Ritz vectors. Denote by Algorithm 4 the refined subspace algorithm of Jia [24], and by Algorithm 5 the Subspace-Block Arnoldi algorithm, and by *Kry–Sch* the block Krylov–Schur algorithm proposed by Zhou and Saad [17].

Example 1. This example tries to illustrate superiority of the modified thick-restarted block Arnoldi algorithm over the thick-restarted block Arnoldi algorithm and Baglama's augmented block householder Arnoldi algorithm. There are three test matrices in this example, which are all from the Matrix Market [2]. We run Algorithms 2 and 3, as well as Baglama's algorithm on the three problems. All the algorithms will be stopped as soon as the residual norms are below $tol = 10^{-6}$. So as to show the efficiency of Algorithm 3 with respect to Algorithm 2, we define

$$\mathbf{Eff} = \frac{mv_{Alg.2} - mv_{Alg.3}}{mv_{Alg.2}}, \tag{4.2}$$

where $mv_{Alg.2}$ and $mv_{Alg.3}$ stand for the number of matrix-vector products required by Algorithms 2 and 3, respectively. Note that the higher **Eff** is, the better the modified algorithm will be.

The first test matrix is the *BWM2000* matrix. It is a 2000×2000 nonsymmetric matrix which arises from modeling the concentration waves for reaction and transport interaction of chemical solutions in a tubular reactor. We compute the two eigenvalues with the largest real part. Table 1 lists the CPU time and the number of matrix-vector products used for convergence.

It is seen from Table 1 that both Algorithms 2 and 3 perform better than Baglama's algorithm, while Algorithm 3 converges the fastest, especially when the dimension of the search subspace is small (recall that the dimension of the search subspaces is $m \times p$). For instance, when $m = 4,\ p = 2,\ k_0 = 4$, Algorithm 3 and Baglama's algorithm use 23.1 s and 46.8 s, respectively, to reach the same accuracy. In other words, Algorithm 3 is about two times faster than Baglama's algorithm. Fig. 1 plots convergence curves of the three algorithms when $m = 10,\ p = 2,\ k_0 = 12$. It is seen that the new algorithm converges smoothly.

The second test matrix is the *Sherman4* matrix. It is a 1104×1104 nonsymmetric matrix, and stems from an oil reservoir modeling. We are interested in the four eigenvalues with smallest real parts. Table 2 gives the numerical results. For this problem, all the algorithms work quite well, and Algorithm 2 is a little better than Baglama's algorithm, while Algorithm 3 outperforms the other two in many cases. Fig. 2 depicts convergence curves of the three algorithms when $m = 5,\ p = 2,\ k_0 = 6$, we see that the new algorithm converges faster than the other two.

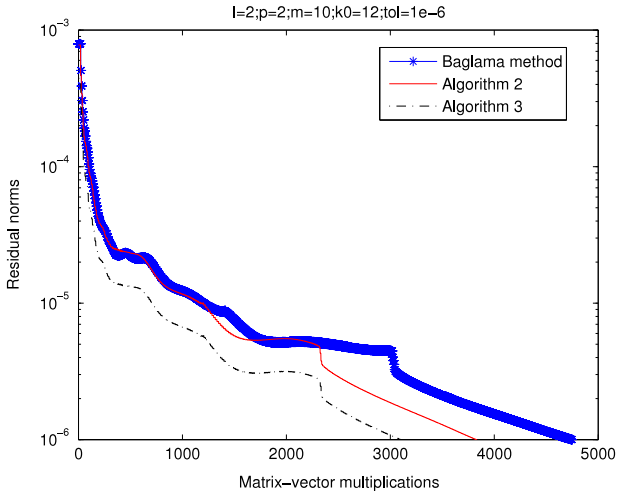

Fig. 1. Example 1: convergence curves of the three algorithms on the *BWM2000* matrix.

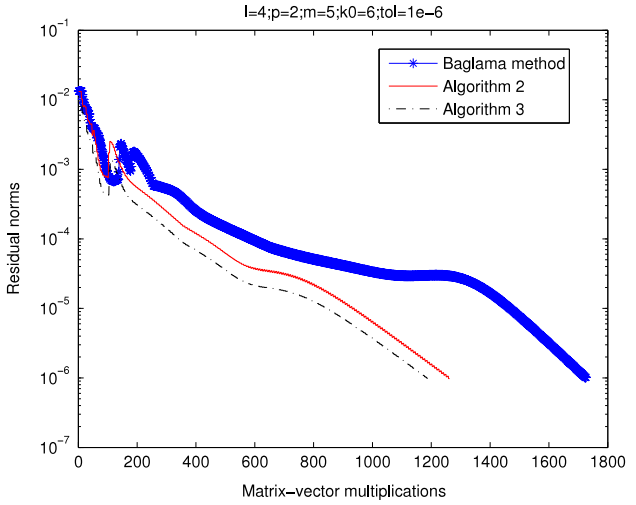

Fig. 2. Example 1: convergence curves of the three algorithms on the *Sherman4* matrix.

Table 3
Numerical results of Example 1.

p	m	k_0	ABHA		Algorithm 2		Algorithm 3		**Eff** (%)
			time	*mv*	*time*	*mv*	*time*	*mv*	
	10	12	9.42	5 748	12.1	8 145	9.23	5 707	29.9
2	8	10	13.2	9 550	12.6	9 573	10.1	7 553	21.1
	5	6	31.4	27 440	23.7	20 561	21.5	18 539	9.83
	6	10	12.8	9 816	18.8	14 232	17.3	12 003	15.6
3	5	6	15.2	15 543	14.3	14 424	13.8	12 373	14.2
	4	10	19.8	16 210	42.3	31 270	38.6	25 515	18.4
5	3	6	27.9	24 190	45.7	34 899	41.4	30 854	11.6
	2	4	48.3	42 260	34.1	28 636	28.5	25 821	9.83

Example 1. Numerical results of Algorithm 2, Algorithm 3, and Baglama's algorithm for computing the three eigenpairs with smallest magnitude, the *1138bus* matrix, $tol = 1e{-}6$.

The third test problem is the *1138bus* matrix. The numerical results are listed in Table 3, where three eigenvalues with the smallest magnitude are required. For this example, we cannot tell which algorithm performs the best. For instance, when $m \times p$ is large, Baglama's algorithm outperforms the other two in many cases. We see that when $m = 4,\ p = 5,\ k_0 = 10$, Baglama's algorithm uses 19.8 s, while Algorithms 2 and 3 make use of 42.3 s and 38.6 s, respectively. On the other hand, when $m \times p$ is relatively small, Algorithms 2 and 3 often work better than Baglama's algorithm, and Algorithm 3 is better

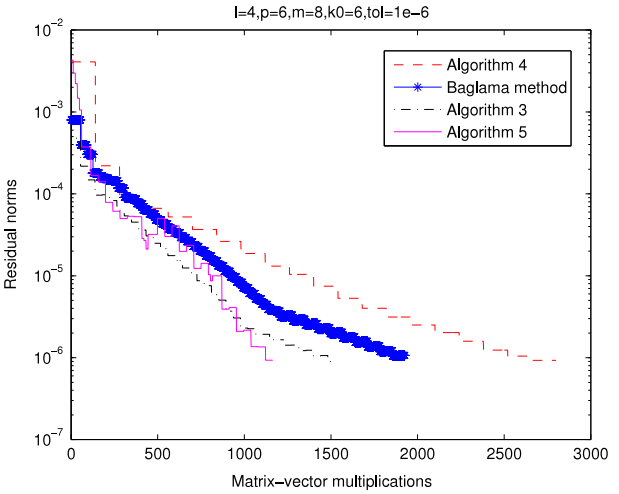

Fig. 3. Example 2: convergence history of the four algorithms.

than Algorithm 2. For example, when $m = 2, p = 5, k_0 = 4$, Baglama's algorithm uses 48.3 s, while Algorithms 2 and 3 need 34.1 and 28.5 s to reach the desired accuracy. So our new algorithm is promising for very large eigenproblems, where the dimension of the search subspaces is often required to be small.

Example 2. In this example, we show the efficiency of the Subspace-Block Arnoldi algorithm for computing a few dominant eigenpairs of large matrices. The test matrix arises from the following constant-coefficient convection–diffusion equation

$$-\triangle u(x, y) + p_1 u_x(x, y) + p_2 u_y(x, y) - p_3 u(x, y) = f(x, y)$$

defined on the unit square region $[0, 1] \times [0, 1]$ with the boundary condition $u(x, y) = 0$, and p_1, p_2, p_3 being nonnegative constants. Discretizing the above equation by five point difference on a uniform N by N grid and numbering the grid points using the row wise natural ordering yields a block tridiagonal matrix of the form

$$A = \begin{pmatrix} T & (\beta+1)I & & & \\ (-\beta+1)I & T & (\beta+1)I & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & (\beta+1)I \\ & & & (-\beta+1)I & T \end{pmatrix},$$

with

$$T = \begin{pmatrix} 4-\tau & \gamma-1 & & & \\ -\gamma-1 & 4-\tau & \gamma-1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \gamma-1 \\ & & & -\gamma-1 & 4-\tau \end{pmatrix},$$

where $\beta = p_1 h/2, \gamma = p_2 h/2, \tau = p_3 h^2$ and $h = 1/(N+1)$. The order of A is $n = N^2$.

The matrix A with $p_1 = 1, p_2 = p_3 = 0$ and $n = 6400$ is tested. We aim to compute the four largest eigenpairs in magnitude. We run Algorithm 3, the refined subspace algorithm (Algorithm 4), the Subspace-Block Arnoldi algorithm (Algorithm 5), and Baglama's algorithm on this problem. In Algorithm 5, we choose $\beta = 0.5$, $maxit1 = 2$, and $maxit2 = 5$. All the algorithms will be stopped as soon as the residual norms are below $tol = 1e{-}6$. Table 4 gives the numerical results, and Fig. 3 depicts convergence history of the four algorithms when $m = 8, p = 6, k_0 = 6$.

For the sake of justice, the dimension of the search subspaces are the same for all the algorithms. Let $V_1 = orth(rand(n, p))$ be the initial block vector for Algorithms 3 and 5 and Baglama's algorithm. Then the refined subspace algorithm makes use of $\tilde{V}_1 = orth([V_1, rand(n, m \times p - p)])$ as the initial guess. Note that there is no Gram–Schmidt orthogonalization in the refined subspace algorithm, so it is cheaper than the other three algorithms.

We see from Table 4 that Algorithm 3 outperforms Baglama's algorithm in terms of matrix-vector products and CPU time, while Algorithm 5 is superior to Algorithm 3 in many cases. The three block Arnoldi-based algorithms may use fewer matrix-vector products than Algorithm 4, however, the latter may need less CPU time than the other three algorithms. The

Table 4
Numerical results of Example 2.

l	p	m	k_0	Algorithm 4		ABHA		Algorithm 3		Algorithm 5	
				time	*mv*	*time*	*mv*	*time*	*mv*	*time*	*mv*
4	5	5	6	8.95	5041	27.3	3405	20.3	2405	18.1	2025
4	5	6	6	13.1	3870	17.4	2130	20.3	2219	14.9	1886
4	5	8	6	15.7	5684	14.7	2310	13.3	2119	8.92	1264
4	6	10	6	16.3	7040	20.5	1788	18.1	1578	15.3	1310
4	6	8	6	5.95	2800	23.3	1956	20.2	1524	15.4	1184
4	6	6	6	11.1	5511	24.1	3324	18.5	2472	14.5	1200
4	8	6	6	15.3	5740	27.2	3856	20.7	1904	15.3	1430
4	8	8	8	10.9	3384	12.9	1792	8.54	1584	8.02	1516
4	10	6	8	11.6	3520	17.4	1880	14.3	1734	8.42	1400

Example 2. Numerical results of the four algorithms, $tol = 1e{-}6$.

Table 5
Numerical results of Example 3.

l	p	m	k_0	Algorithm 4		ABHA		Algorithm 3		Algorithm 5	
				time	*mv*	*time*	*mv*	*time*	*mv*	*time*	*mv*
3	4	8	6	0.64	4140	1.53	1896	1.38	1254	0.70	690
3	6	4	6	0.63	4002	2.14	2064	1.52	1344	0.76	681
4	6	5	6	1.06	6192	1.74	1956	1.53	1836	0.82	704
4	6	6	6	0.95	5200	1.53	1260	1.23	1092	0.96	690
4	8	6	6	1.46	6860	2.23	1232	2.18	1106	1.20	776
5	8	5	6	1.41	7015	1.98	1984	1.56	1000	1.03	671
5	7	5	6	1.37	7100	1.86	1925	1.41	1463	1.02	881
5	10	4	7	1.07	5290	2.59	2640	2.03	1502	1.34	1036

Example 3. Numerical results of the four algorithms, $tol = 1e{-}6$.

reason is that, as we have pointed out, the refined subspace algorithm is (much) cheaper than the other three algorithms; see also the numerical experiments made in [27].

When $m \times p$ is relatively small, Algorithm 4 may converge faster than Algorithm 5, although the number of matrix-vector products needed by the former can be much larger that of the latter, cf. $m = 5, p = 5, k_0 = 6$. However, when $m \times p$ is large, Algorithm 5 may work better than the refined subspace iteration both in terms of matrix-vector products and CPU time, say, when $m = 6, p = 10, k_0 = 8$. Therefore, Algorithm 5 is a competitive candidate for computing a few dominant eigenvalues and the corresponding eigenvectors of large, nonsymmetric matrices.

Example 3. This example is due to Morgan [30]. The test matrix is a tridiagonal matrix with 1, 2, 2.05, 2.1, 3, 4, 5, . . . , 998 on the main diagonal, -0.1 in each superdiagonal position and 0.1 in each subdiagonal position. We run Baglama's algorithm, Algorithms 3–5 for computing a few dominant eigenpairs of the matrix. In Algorithm 5, we choose $\beta = 0.7$, $maxit1 = 2$ and $maxit2 = 6$. The stopping criterion is $tol = 10^{-6}$. Table 5 lists the numerical results obtained.

One observes that all the algorithms run quite well, while Algorithm 5 is superior to other three algorithms in terms of matrix-vector products. Again, we see that the refined subspace algorithm requires more matrix-vector products than the others, but the CPU time required by the algorithm may be the least. As was pointed out in Example 2, this is due to the fact that the cost of the refined subspace algorithm is much less than that of the others. Fig. 4 depicts the convergence history of the four algorithms when $l = 3, m = 4, p = 6, k_0 = 6$. We see that Algorithm 5 runs quite well.

Example 4. In this example, we compare our new algorithms with the block Krylov–Schur algorithm (Kry–Sch) due to Zhou and Saad [17] for large symmetric eigenvalue problems. The test matrix is the *tuma2* matrix which is available from http://www.cise.ufl.edu/research/sparse/matrices/. It is a real symmetric matrix of order $n = 12{,}992$, with 49,365 nonzeros. We are interested in the six dominant eigenpairs of the matrix, and all the algorithms will be stopped as soon as the residual norms are below $tol = 1e - 10$.

For this problem, we directly apply Algorithms 2, 3 and 5 to this large symmetric eigenproblem. Notice that the cost of the three algorithms needed per iteration is higher than that of the block Krylov–Schur algorithm in which the block Lanczos process is used instead of the block Arnoldi procedure. In Algorithm 5, we set $\beta = 0.7$, $maxit1 = 2$, and $maxit2 = 6$. Table 6 lists the numerical results obtained.

Three remarks are given. First, it is seen that the block Krylov–Schur algorithm may use a lot more matrix-vector products than the other three algorithms for convergence, however, it may need the least CPU time among the four algorithms. Indeed, as was pointed out before, the reason is due to the fact that the block Krylov–Schur algorithm is based on the block Lanczos process [14,17].

Second, when the dimension of the search subspaces is low, Algorithms 2 and 3 can be much faster than the block Krylov–Schur algorithm. For instance, when $l = 6, m = 3, p = 7, k_0 = 12$, Algorithms 2 and 3 are about twice as fast

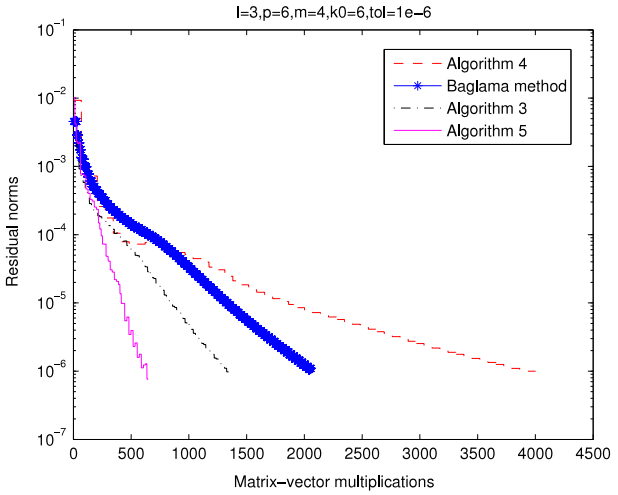

Fig. 4. Example 3: convergence curves of the four algorithms.

Table 6
Numerical results of Example 4.

l	p	m	k_0	Algorithm 2		Algorithm 3		Kry–Sch		Algorithm 5	
				time	*mv*	*time*	*mv*	*time*	*mv*	*time*	*mv*
6	6	3	10	30.5	1682	32.2	1520	46.3	6660	*n.c.*	*n.c.*
6	6	4	10	39.2	1858	32.6	1724	20.2	2508	20.2	1532
6	6	5	10	33.2	1470	24.4	1356	14.2	1812	17.1	1288
6	7	3	12	20.8	1029	24.0	946	48.8	6720	*n.c.*	*n.c.*
6	7	4	12	16.6	812	16.6	755	14.6	1694	25.8	1266
6	7	5	12	16.3	748	16.5	778	10.9	1295	16.8	864
6	8	3	15	25.7	978	35.4	977	62.3	7784	*n.c.*	*n.c.*
6	8	4	15	21.8	797	23.9	754	16.0	1664	36.5	1473
6	8	5	15	21.6	740	20.9	698	10.7	1216	20.8	770

Example 4. Numerical results of the four algorithms on the *tuma2* matrix, $tol = 1e-10$. Here *n.c.* stands for no convergence occurs even using 1e+4 matrix-vector products.

as the block Krylov–Schur algorithm. This is favorable when the matrix in question is very large. We see that Algorithm 3 works better than Algorithm 2 in terms of matrix-vector products. Moreover, we also observe that Algorithm 5, the hybrid algorithm, may not converge when m is very small. The reason is that the refined subspace method does not perform well on this problem. Fortunately, when m becomes large, Algorithm 5 is often superior to the three algorithms in terms of matrix-vector products. This implies that the hybrid strategy is still valid for this problem.

Third, we remind the reader that the block Krylov–Schur algorithm only works for large symmetric eigenproblems [17], while our new algorithms can be used to calculate a few selected eigenpairs of both symmetric and non-symmetric matrices.

Example 5. The aim of this paper is twofold. First, we show sharpness of Theorem 2.4. Second, we try to show that when m is sufficiently large, the Subspace-Block Arnoldi algorithm can work quite well even when the refined subspace iteration does not work. The problem is from [2]. Dielectric channel wave-guide problems arise from many integrated circuit application. Consider the governing Helmholtz equation for the metric field H

$$\begin{cases} \nabla^2 H_x + k^2 n^2(x,y) H_x = \tau^2 H_x, \\ \nabla^2 H_y + k^2 n^2(x,y) H_y = \tau^2 H_y. \end{cases}$$

By finite difference, we obtain a nonsymmetric matrix eigenvalue problem of the form

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} H_x \\ H_y \end{pmatrix} = \tau^2 \begin{pmatrix} B_{11} & \\ & B_{22} \end{pmatrix} \begin{pmatrix} H_x \\ H_y \end{pmatrix},$$

where C_{11} and C_{22} are five or tri-diagonal matrices, C_{12} and C_{21} are (tri-)diagonal matrices, and B_{11} and B_{22} are nonsingular diagonal matrices. There are eigenvalues with negative real part several orders of magnitude larger than the desired eigenvalues with positive real part, and also the desired eigenvalues are clustered for large n. So this problem presents a challenge to the existing numerical methods [1].

We test the problem with $n = 8192$ and calculate some rightmost eigenpairs (which are also the ones in largest magnitude), and the algorithms will be stopped as soon as the residual norms are below $tol = 10^{-6}$. We run Algorithms 3–5,

Table 7
Numerical results of Example 5.

l	p	m	k_0	Algorithm 4		ABHA		Algorithm 3		Algorithm 5	
				time	*mv*	*time*	*mv*	*time*	*mv*	*time*	*mv*
3	10	6	8	*n.c*	*n.c*	90.1	5240	75.4	3398	38.1	2319
3	6	6	6	*n.c*	*n.c*	77.6	5820	50.4	2292	18.3	1575
4	10	6	8	*n.c*	*n.c*	123	5900	77.6	3242	66.1	2354
4	8	8	8	*n.c*	*n.c*	71.2	4768	65.0	3376	40.8	1800
4	6	8	6	*n.c*	*n.c*	102	6120	59.6	3456	21.5	1462
4	6	6	6	*n.c*	*n.c*	73.8	6684	46.3	3132	23.3	1624

Example 5. Numerical results of the four algorithms, $tol = 1e{-}6$. Here *n.c.* stands for no convergence occurs even using 1e+4 matrix-vector products.

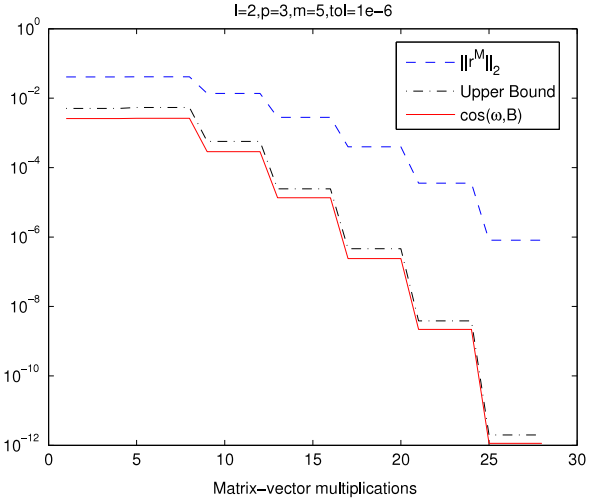

Fig. 5. Example 5: sharpness of Theorem 2.4 for the *non-restarted* modified block Arnoldi algorithm.

as well as Baglama's algorithm on this problem. For Algorithm 5, we pick $\beta = 0.7$, $maxit1 = 2$ and $maxit2 = 6$. Table 7 lists the numerical results obtained.

It is obvious to see from Table 7 that Algorithm 5 is more efficient than the other three algorithms, even if the refined subspace algorithm does not work at all. This can be interpreted as follows. As iteration proceeds, the refined subspace iteration can purify the initial block vector for the block Arnoldi algorithm. On the other hand, the block Arnoldi algorithm can provide a better initial guess for the refined subspace iteration. As a result, a periodic combination of the two algorithms may improve the performance of both the original algorithms considerably.

In order to show sharpness of Theorem 2.4, in Figs. 5 and 6 we plot curves of $\|r^M\|_2$, $\cos\angle(w, \mathscr{B})$, and the right hand side of (2.10), respectively. Two comments are in order. First, we observe that the upper bound of (2.10) is sharp, both for the restarted and non-restarted modified block Arnoldi versions. Second, as was pointed out in Remark 1, we see that the speed that $\cos\angle(w, \mathscr{B})$ tends to zero is much faster than that with which $\|r^M\|_2$ does so, and $\cos\angle(w, \mathscr{B}) = O(\|r^M\|_2^2)$. As a result, $\|r^M\|_2$ may be (much) smaller than $\|\tilde{r}\|_2$ in practice, cf. Theorem 2.2. This explains in some degree why the modified block Arnoldi algorithm outperforms the standard block Arnoldi algorithm in many cases [25,26,32].

5. Conclusion and future work

In this paper, we first introduce a thick-restarted block Arnoldi algorithm for large unsymmetric eigenproblems, and then make use of the modified Ritz vectors [24,23,25,26,32] to take the place of Ritz vectors, and propose a thick-restarted block Arnoldi algorithm with modified Ritz vectors. Some properties of this new algorithm are given. Moreover, we combine the refined subspace algorithm with the modified thick-restarted block Arnoldi algorithm, and propose a Subspace-Block Arnoldi algorithm for computing a few dominant eigenpairs of large matrices. Numerical results show the effectiveness of our new algorithms.

However, there is still much work which needs to done. For instance, how to understand the superiority of the Subspace-Block Arnoldi algorithm over its original counterpart from a theoretical point of view. Can we determine the optimal parameters so that the Subspace-Block Arnoldi algorithm can work more efficiently? They are under investigation and are certainly a part of our future work.

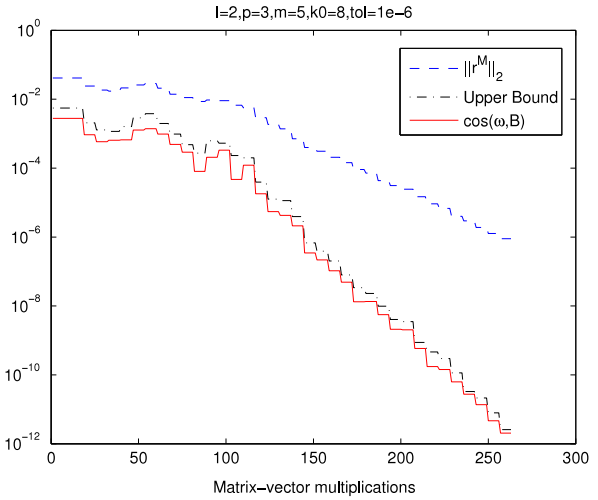

Fig. 6. Example 5: sharpness of Theorem 2.4 for the *thick-restarted* modified block Arnoldi algorithm.

Acknowledgements

Special appreciation goes to two anonymous referees for their stimulating suggestions and invaluable comments that enable us to greatly improve the presentation of this paper. Meanwhile, we would like to express our sincere thanks to Prof. James Baglama and Prof. Yunkai Zhou for providing us MATLAB files of the augmented block Householder Arnoldi algorithm [10] and the block Krylov–Schur algorithm [17]. We also thank Prof. Tim Davis for providing us data files of some test matrices used in this paper.

The first author is supported by the Postgraduate Science Topical Foundation of Xuzhou Normal University under grant 08YLB023.

The second author is supported by the National Science Foundation of China under grant 10901132, the Natural Science Foundation for Colleges and Universities in Jiangsu Province under grant 08KJB110012, the Qing-Lan Project of Jiangsu Province under grant QL200612, and the Natural Science Foundation of Xuzhou Normal University under grant 08XLY01.

References

[1] Z. Bai, R. Barrett, D. Day, J. Demmel, J. Dongarra, A test matrix collection for non-Hermitian eigenvalue problems, Technical Report CS-97-355, University of Tennessee, Knoxville, TN, 1997.
[2] http://math.nist.gov/MatrixMarket/.
[3] Y. Saad, Numerical Methods for Large Eigenvalue Problems, in: Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, 1992.
[4] G. Stewart, Matrix Algorithms II: Eigensystems, SIAM, Philadelphia, 2001.
[5] W. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Quart. Appl. Math. 9 (1951) 17–29.
[6] Y. Saad, Projection methods for solving large unsymmetric matrices, Linear Algebra Appl. 34 (1980) 269–295.
[7] J. Baglama, Augmented block householder Arnoldi method, Linear Algebra Appl. 429 (2008) 2315–2334.
[8] M. Sadkane, A block-Arnoldi Chebyshev method for computing leading eigenpairs of large sparse unsymmetric matrices, Numer. Math. 64 (1993) 181–194.
[9] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, Philadelphia, 2000.
[10] J. Baglama, D. Calvetti, L. Reichel, IRBL: an implicitly block Lanczos method for large-scale Hermitian eigenproblems, SIAM J. Sci. Comput. 24 (2003) 1650–1677.
[11] G.H. Golub, F. Luk, M. Overton, A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix, ACM Trans. Math. Software 7 (1981) 149–169.
[12] Z. Jia, A refined iterative algorithm based on the block Arnoldi process for large unsymmetric eigenproblems, Linear Algebra Appl. 270 (1998) 171–189.
[13] R. Morgan, Restarted block GMRES with deflation of eigenvalues, Appl. Numer. Math. 54 (2005) 222–236.
[14] Y. Saad, On the rates of convergence of the Lanczos and the block-Lanczos methods, SIAM J. Numer. Anal. 17 (1980) 687–706.
[15] M. Sadkane, Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems, Numer. Math. 64 (1993) 195–211.
[16] M. Sadkane, R. Sidje, Implementation of a variable block Davidson method with deflation for solving large sparse eigenproblems, Numer. Algorithms 20 (2) (1999) 217–240.
[17] Y. Zhou, Y. Saad, Block Krylov–Schur method for large symmetric eigenvalue problems, Numer. Algorithms 47 (2008) 341–359.
[18] R. Morgan, M. Zeng, A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity, Linear Algebra Appl. 415 (2006) 96–113.
[19] D. Sorensen, Implicit application of polynomial filters in a *k*-step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1992) 357–385.
[20] R. Morgan, GMRES with deflated restarting, SIAM. J. Sci. Comput. 24 (2002) 20–37.
[21] K. Wu, H. Simon, Thick-restart Lanczos method for symmetric eigenvalue problems, SIAM J. Matrix Anal. Appl. 22 (2000) 602–616.
[22] I. Yamazaki, Z. Bai, H. Simon, L. Wang, K. Wu, Adaptive projection subspace dimension for the thick-restart Lanczos method, ACM Trans. of Math. Software 37 (3) (2010).
[23] Z. Jia, L. Elsener, Improving eigenvectors in Arnoldi's method, J. Comput. Math. 18 (2000) 265–276.
[24] Z. Jia, A variation on the Arnoldi method for large unsymmetric eigenproblems, Acta Math. Appl. Sin. 14 (1998) 425–432.

[25] G. Wu, An iterative block Arnoldi algorithm with modified approximate eigenvectors for large nonsymmetric eigenvalue problems, Appl. Math. Comput. 153 (2004) 611–643.
[26] G. Wu, A modified harmonic block Arnoldi algorithm with adaptive shifts for large interior eigenproblems, J. Comput. Appl. Math. 205 (2007) 343–363.
[27] Z. Jia, A refined subspace iterative algorithm for large sparse eigenproblems, Appl. Numer. Math. 32 (1) (2000) 35–52.
[28] G. Wu, Y. Wei, A Power–Arnoldi algorithm for computing PageRank, Numer. Linear Algebra Appl. 14 (2007) 521–546.
[29] K. Wu, H. Simon, Thick-restart Lanczos method for sysmmetric eigenvalue problems, SIAM J. Matrix Anal. Appl. 22 (2000) 602–616.
[30] R. Morgan, On restarting the Arnoldi method for nonsymmetric eigenvalue problems, Math. Comput. 65 (1996) 1213–1230.
[31] Z. Jia, G. Stewart, An analysis of the Rayleigh–Ritz method for approximating eigenspaces, Math. Comput. 70 (2001) 637–647.
[32] G. Wu, Y. Zhang, Y. Wei, Krylov subspace algorithms for computing GeneRank for the analysis of microarray data mining, J. Comput. Biol. 17 (2010) 631–646.
[33] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd edition, The Johns Hopkins University Press, Baltimore, London, 1996.