IMPROVING EIGENPAIRS OF AUTOMATED MULTILEVEL SUBSTRUCTURING WITH SUBSPACE ITERATIONS *

JIACONG YIN[†‡], HEINRICH VOSS[‡], AND PU CHEN[†]

Key words. Eigenvalue, eigenvector, AMLS, subspace iteration

AMS subject classification. 65F15

Abstract. This paper improves eigenvalues and eigenvectors obtained from the Automated Multilevel Substructuring (AMLS) method by subspace iterations. Two subspace iteration strategies are proposed: one is independent from AMLS and the other takes advantage of the transformed block diagonal stiffness matrix from AMLS for subspace iterations. Numerical experiments via four problems with different scales show that: (a) the errors of AMLS eigenpairs could be significantly reduced at the lower end of the spectrum with quite a few subspace iteration steps; (b) the latter subspace iteration method that cooperates with AMLS is more efficient than the former one in terms of computational costs.

1. Introduction. The Automated Multilevel Substructuring (AMLS) method, proposed by Bennighof and his colleagues [7, 20, 8], is an efficient condensation method to determine hundreds to thousands of eigenmodes and frequency responses for quite large and complex structures. Compared with the classical block Lanczos algorithm [17], AMLS reduces computational resources in terms of time and hardware requirements. Hence, it has been successfully applied to many engineering areas in recent years, including the automobile industry [23], ship vibrations [24] and electromagnetic problems [31, 27]. The standard AMLS only addresses linear eigenvalue problems. Elssel and Voss firstly developed it for gyroscopic [14] and nonlinear problems [12, 15]. Then, alternative AMLS schemes for strong and week fluid-solid interaction problems were researched by Stammberger and Voss as well [28].

Nevertheless, the accuracy of AMLS is not as good as the classical Lanczos method. Bekas and Saad pointed that AMLS is actually the first order approximation of a corresponding rational eigenvalue problem [6]. The error of AMLS depends on the selected cutoff frequency that truncates eigenmodes of each substructure. Generally speaking, the higher cutoff frequency is set, the better results come out. Yang et al. [30] derived a rough error bound of AMLS. Later a better a priori bound was given by Elssel and Voss [13]. To improve AMLS, the shift acceleration technique was researched by Elssel [12] and Ko et al. [22]. Liao, Bai and Gao also suggested a moment-matching mode selection strategy [25] to catch important eigenmodes of substructures.

This study chooses a different way to improve AMLS, i.e. employing the subspace iteration method to refine the eigenpairs computed by AMLS. The subspace iteration algorithm is a generalized inverse power method to compute several lowest eigenvalues for large sparse systems. It was firstly studied by Clint and Jennings in 1970 [9] and named by Bathe and Wilson [5]. In the past decades, subspace iteration method has

*This work was supported by the National Natural Science Foundation of China under the grant No.10972005 and the China Scholarship Council with the contract No.2010601199.

[†]State Key Laboratory for Turbulence and Complex Systems & Department of Mechanics and Aerospace Engineering, College of Engineering, Peking University. 100871 Beijing, China. ({jcyin, chenpu}@pku.edu.cn)

[‡]Institute of Numerical Simulation, Hamburg University of Technology, D-21071 Hamburg, Germany. (voss@tu-harburg.de)

integrated many useful techniques, including shifts [4, 29], Sturm's sequence check and convergence criteria [3], etc. A recent review of the development of subspace iterations was presented in [32].

In this work, we propose two subspace iteration methods for the standard AMLS, considering only linear eigenvalue problems. The first one, Method A, is independent from AMLS. It just takes AMLS as a pre-conditioner to construct the initial subspace, excluding any AMLS data for iterations. But soon we noticed that the transformed stiffness matrix in AMLS is block diagonal. Utilizing this property could reduce the computational costs for both linear equation solutions and matrix condensations in subspace iterations. Thus, the second one, Method B, will use the transformed stiffness matrix and the original untransformed mass matrix for subspace iterations.

The paper is organized as follows. Section 2 gives a brief review of the theory and implementation of AMLS, which will contain graphical partition, block Gauss elimination, matrices condensation, solution of final reduced system and computation of eigenvectors. Section 3 discusses the proposed two AMLS subspace iteration methods in detail. Their performance and efficiency will be given via four problems with different scales in Section 4. Finally, this paper ends with a short conclusion in Section 5.

2. The standard AMLS. The purpose of AMLS for linear eigenvalue problems is to find n_p eigensolutions of

$$Kx = \lambda Mx, \tag{2.1}$$

in a frequency range of interest. Usually (2.1) is a finite element model of some problem and the stiffness matrix $K \in \mathbb{R}^{n\times n}$ and the mass matrix $M \in \mathbb{R}^{n\times n}$ are symmetric positive definite.

2.1. Partition and AMLS tree. Similarly as in the component mode synthesis (CMS) [19, 10] the structure is partitioned into a small number of substructures based on the sparsity pattern of the system matrices, but more generally than in CMS these substructures in turn are substructured on a number of levels yielding a tree topology for the substructures.

We stress the fact that substructuring does not necessarily mean that it is obtained by a domain decomposition of a real structure, but it is understood in a purely algebraic sense. Although substructuring by hand may yield much smaller interfaces than the ones obtained by automatic partitioners [13], the dissection of the matrices can be derived by applying a graph partitioner like CHACO [18] or METIS [21] to the undirected graph corresponding to the nonzero pattern of the matrices under consideration. These programs have been designed to construct fill–reducing orderings of sparse matrices for efficient factorization but have shown to be a beneficial basis of substructuring methods as well.

For convenience to review AMLS, this paper takes a binary partition with only two levels of substructures as an example. But we declare that all derived formulae and notations were generalized to non-binary partitions with more levels of substructures for implementation.

With that specific two-level binary partition, the original stiffness matrix K is

reordered as

$$P^TKP = \begin{pmatrix} K_{11} & & K_{13} & & & & K_{17} \\ & K_{22} & K_{23} & & & & K_{27} \\ K_{31} & K_{32} & K_{33} & & & & K_{37} \\ & & & K_{44} & & K_{46} & K_{47} \\ & & & & K_{55} & K_{56} & K_{57} \\ & & & K_{64} & K_{65} & K_{66} & K_{67} \\ K_{71} & K_{72} & K_{73} & K_{74} & K_{75} & K_{76} & K_{77} \end{pmatrix}, \tag{2.2}$$

where P is the permutation matrix that represents the reordering operation of partition, $K_{ij} \in \mathbb{R}^{n_i \times n_j}$ and $K_{ji} = K_{ij}^T, (i, j = 1, ..., 7)$. The reordered mass matrix P^TMP has the same block structure with (2.2). However, the reordering of (K, M) is usually implemented implicitly. So we do not distinguish (P^TKP, P^TMP), (K, M) in the following parts of this paper.

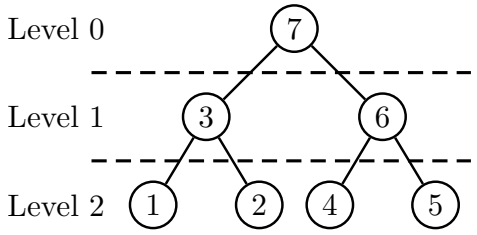

FIG. 2.1. *The two-level AMLS partition tree*

AMLS *partition tree* is an important concept for the organization of substructures. The corresponding tree of (2.2) is given in Fig.2.1. Here we explain some technical terms and notations related to the tree. Firstly, each circle in Fig.2.1 labeled by a number s ($s = 1, \ldots, 7$) is a *treenode* that corresponds to a substructure of (K_{ss}, M_{ss}). For convenience, the set of all treenodes on level l will be denoted as $\mathcal{S}_l$, e.g. in Fig.2.1, $\mathcal{S}_{l=2} = \{1, 2, 4, 5\}$, $\mathcal{S}_{l=1} = \{3, 6\}$ and $\mathcal{S}_{l=0} = \{7\}$. A treenode s that has at least one sub-treenode j is called the *parent* of j, denoted as $s = \mathcal{P}(j)$; while the sub-treenode j is the *child* of s, denoted as $j \in \mathcal{C}(s)$. Here $\mathcal{C}(s)$ refers to the set of children of the treenode s. For example, $3 = \mathcal{P}(1)$ and $3 \in \mathcal{C}(7)$. Treenode 7 is the parent of treenode 1's parent, so 7 is called an *ancestor* of 1, denoted as $7 \in \mathcal{A}(1)$. We define that $\mathcal{P}(s)$ is also an ancestor of treenode s, i.e. $\mathcal{P}(s) \subset \mathcal{A}(s)$. Finally, we call the treenode that does not have any ancestor as the *root* of the tree; while those treenodes which do not have any children are called the *leaves* of the tree. For Fig.2.1, the root is treenode 7 and the set of leaves equals $\mathcal{S}_{l=2}$.

The creation of an AMLS partition tree follows the sequence of *preorder traversal* of the tree, i.e. visiting a treenode s always before its children, $\mathcal{C}(s)$. For Fig.2.1, this sequence is {7,3,1,2,6,4,5}, e.g.

2.2. The block Gaussian elimination. After the partition, AMLS starts a block Gaussian elimination approach to transform K into a block diagonal matrix $\tilde{K}$, i.e.

$$\tilde{K} = U^TKU = \mathrm{diag}(K_{11}, K_{22}, \tilde{K}_{33}, K_{44}, K_{55}, \tilde{K}_{66}, \tilde{K}_{77}), \tag{2.3}$$

in which U is the transformation matrix. The corresponding transformed mass matrix $\tilde{M}$ is

$$\tilde{M} = U^T M U = \begin{pmatrix} M_{11} & & \tilde{M}_{13} & & & & \tilde{M}_{17} \\ & M_{22} & \tilde{M}_{23} & & & & \tilde{M}_{27} \\ \tilde{M}_{31} & \tilde{M}_{32} & \tilde{M}_{33} & & & & \tilde{M}_{37} \\ & & & M_{44} & & \tilde{M}_{46} & \tilde{M}_{47} \\ & & & & M_{55} & \tilde{M}_{56} & \tilde{M}_{57} \\ & & & \tilde{M}_{64} & \tilde{M}_{65} & \tilde{M}_{66} & \tilde{M}_{67} \\ \tilde{M}_{71} & \tilde{M}_{72} & \tilde{M}_{73} & \tilde{M}_{74} & \tilde{M}_{75} & \tilde{M}_{76} & \tilde{M}_{77} \end{pmatrix}. \tag{2.4}$$

In implementation, the block Gaussian elimination approach performs on substructures with a sequence of *postorder traversal* of the corresponding AMLS partition tree. Postorder traversal means to visit a substructure s always later than its children $\mathcal{C}(s)$. For Fig.(2.1), this sequence is {1,2,3,4,5,6,7}. The block Gaussian elimination on each substructure (treenode) s produces a factor, U_s, with the form of

$$U_s = \begin{pmatrix} I & & & & \\ & \ddots & \Psi_{si} & & \Psi_{s7} \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{pmatrix}, \quad \forall s \in \mathcal{S}_{l=2,1},\ i = \mathcal{A}(s). \tag{2.5}$$

Thus, we have

$$U = \prod_{s=1}^{6} U_s. \tag{2.6}$$

Those Ψ_{si}, non-diagonal blocks of U_s, are known as *constraint modes*. If we use a superscript $[s]$ to denote that the elimination on substructures $\{1, \ldots, s\}$ have been finished and let

$$(\tilde{K}_{ij}^{[0]}, \tilde{M}_{ij}^{[0]}) = (K_{ij}, M_{ij}),$$

then the sub-matrices of $(U_1 \ldots U_s)^T K (U_1 \ldots U_s)$, $(U_1 \ldots U_s)^T M (U_1 \ldots U_s)$ and the expression of Ψ_{si} could be presented as follows:

(a) for the case that s is a leaf, i.e. $s \in \mathcal{S}_{l=2}$, $\forall i, j \in \mathcal{A}(s)$, we have

$$\Psi_{si} = -K_{ss}^{-1} K_{si}, \tag{2.7a}$$

$$\tilde{K}_{si}^{[s]} = 0, \tag{2.7b}$$

$$\tilde{M}_{si}^{[s]} = M_{si} + M_{ss}\Psi_{si}, \tag{2.7c}$$

$$\tilde{K}_{ij}^{[s]} = \tilde{K}_{ij}^{[s-1]} + \Psi_{si}^T K_{sj}, \tag{2.7d}$$

$$\tilde{M}_{ij}^{[s]} = \tilde{M}_{ij}^{[s-1]} + \Psi_{si}^T M_{sj} + M_{si}^T \Psi_{sj} + \Psi_{si}^T M_{ss} \Psi_{sj}. \tag{2.7e}$$

(b) for the case that s is not a leaf, i.e. $s \in \mathcal{S}_{l=1}$, $\forall i, j \in \mathcal{A}(s)$, we have

$$\Psi_{si} = -(\tilde{K}_{ss}^{[s-1]})^{-1}\tilde{K}_{si}^{[s-1]}, \tag{2.8a}$$
$$\tilde{K}_{si}^{[s]} = 0, \tag{2.8b}$$
$$\tilde{M}_{si}^{[s]} = \tilde{M}_{si}^{[s-1]} + \tilde{M}_{ss}^{[s-1]}\Psi_{si}, \tag{2.8c}$$
$$\tilde{K}_{ij}^{[s]} = \tilde{K}_{ij}^{[s-1]} + \Psi_{si}^{T}\tilde{K}_{sj}^{[s-1]}, \tag{2.8d}$$
$$\tilde{M}_{ij}^{[s]} = \tilde{M}_{ij}^{[s-1]} + \Psi_{si}^{T}\tilde{M}_{sj}^{[s-1]} + (\tilde{M}_{si}^{[s-1]})^{T}\Psi_{sj} + \Psi_{si}^{T}\tilde{M}_{ss}^{[s-1]}\Psi_{sj}. \tag{2.8e}$$

Obviously, the transformed $(\tilde{K}_{ss}, \tilde{M}_{ss})$, $(s \in \mathcal{S}_{l=1,0})$, in (2.3) and (2.4) is explicitly determined when the eliminations on substructures $\{1, \ldots, (s-1)\}$ have been finished,

$$(\tilde{K}_{ss}, \tilde{M}_{ss}) = (\tilde{K}_{ss}^{[s-1]}, \tilde{M}_{ss}^{[s-1]}), \quad \forall s \in \mathcal{S}_{l=1,0}. \tag{2.9}$$

However, the non-diagonal blocks, $\tilde{M}_{si}$, $(s \in \mathcal{S}_{l=1,0}, i \in \mathcal{A}(s))$, are determined by

$$\tilde{M}_{si} = \tilde{M}_{si}^{[i]}, \quad \forall i \in \mathcal{A}(i) \tag{2.10}$$

in an implicit way that will be further described in the next subsection.

2.3. Matrices condensation. Remember that AMLS is essentially a condensation method. It contains two step:

(a) determining n_{m_s} eigensolutions, $n_{m_s} \ll n_s$, of each substructure s

$$K_{ss}Z_s = M_{ss}Z_{ss}\Lambda_s, \quad \forall s \in \mathcal{S}_{l=2} \tag{2.11a}$$
$$\tilde{K}_{ss}Z_s = \tilde{M}_{ss}Z_{ss}\Lambda_s, \quad \forall s \in \mathcal{S}_{l=1,0} \tag{2.11b}$$

not exceeding a given *cutoff frequency* ω_{cutoff}:

$$\lambda_{s,i} < \omega_{\text{cutoff}}^2 \equiv \lambda_{\text{cutoff}}, (i = 1, \ldots, n_{m_s}). \tag{2.12}$$

Here $Z_s = [z_{s,1}, \ldots, z_{s,n_{m_s}}] \in \mathbb{R}^{n_s \times n_{m_s}}$ and $\Lambda_s = \text{diag}[\lambda_{s,1}, \ldots, \lambda_{s,n_{m_s}}]$.

(b) using the global projection matrix $Z \in \mathbb{R}^{n \times n_m}$ $(n_m = \sum n_{m_s},\ n_m \ll n)$,

$$Z = \text{diag}(Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7), \tag{2.13}$$

to project $(\tilde{K}, \tilde{M})$ onto a reduced subspace. The final reduced matrices $K_c \in \mathbb{R}^{n_m \times n_m}$ and $M_c \in \mathbb{R}^{n_m \times n_m}$ are

$$K_c = Z^T\tilde{K}Z = \text{diag}(\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \Lambda_5, \Lambda_6, \Lambda_7) \tag{2.14}$$

and

$$M_c = Z^T\tilde{M}Z = \begin{pmatrix} I & & m_{13} & & & & m_{17} \\ & I & m_{23} & & & & m_{27} \\ m_{31} & m_{32} & I & & & & m_{37} \\ & & & I & & m_{46} & m_{47} \\ & & & & I & m_{56} & m_{57} \\ & & & m_{64} & m_{65} & I & m_{67} \\ m_{71} & m_{72} & m_{73} & m_{74} & m_{75} & m_{76} & I \end{pmatrix}, \tag{2.15}$$

respectively, in which $m_{js} = m_{sj}^T$, $m_{js} = Z_j^T\tilde{M}_{js}Z_s$.

In implementation, the block Gaussian elimination approach and these condensation operations are performed in an interleaving way: as soon as a sub-matrix pencil of (K_{ss}, M_{ss}) or $(\tilde{K}_{ss}, \tilde{M}_{ss})$ has been formed, the eigensolution of (Λ_s, Z_s) and the calculation of m_{js} will be started immediately.

The principle to obtain m_{js} is: (a) if s is a leaf, then compute

$$\tilde{\mu}_{si}^{[s]} = Z_s^T(M_{si} + M_{ss}\Psi_{si}), \quad \forall i \in \mathcal{A}(s). \tag{2.16}$$

(b) if s is not a leaf, then compute

$$\tilde{\mu}_{si}^{[s]} = Z_s^T(\tilde{M}_{si}^{[s-1]} + \tilde{M}_{ss}^{[s-1]}\Psi_{si}), \qquad \forall i \in \mathcal{A}(s) \tag{2.17a}$$

$$m_{js} = \tilde{\mu}_{js}^{[s-1]} Z_s, \qquad \forall j \in \mathcal{C}(s) \tag{2.17b}$$

$$\tilde{\mu}_{ji}^{[s]} = \tilde{\mu}_{js}^{[s-1]}\Psi_{si} + \tilde{\mu}_{ji}^{[s-1]}. \qquad \forall j \in \mathcal{C}(s), \quad \forall i \in \mathcal{A}(s) \tag{2.17c}$$

Comparing (2.17a) and (2.17c) with (2.8c), we can see that the updating of $\tilde{M}_{si}$ is in fact implicitly performed on the form of $\tilde{\mu}_{si} = Z_s^T\tilde{M}_{si}$, instead of $\tilde{M}_{si}$. That is because $\tilde{M}_{si}$ is only used to compute $m_{si} = Z_s^T\tilde{M}_{si}Z_i$ in the standard AMLS approach and apparently, the computational costs and the required storage of $\tilde{\mu}_{si} \in \mathbb{R}^{n_{m_s} \times n_i}$ is much less than $\tilde{M}_{si} \in \mathbb{R}^{n_s \times n_i}$ since $n_{m_s} \ll n_s$.

2.4. Final system. When all partitions, block Gaussian eliminations and the condensation operations have been finished, we obtain the final condensed eigenproblem by AMLS, i.e.

$$K_c X_c = M_c X_c \Lambda_c. \tag{2.18}$$

Its solution will provide an approximation, (Λ_c, V), of the eigenpairs of the original problem (2.1), (Λ, X). Here

$$V = UZX_c = (\prod_{s=1}^{6} U_s)ZX_c \tag{2.19}$$

contains the approximated global eigenvectors.

We summarize the standard AMLS without forming eigenvectors in Algorithm 1, in which the substructures are partitioned on the preorder traversal of the AMLS tree, while the block Gaussian eliminations and condensations are performed on the postorder traversal of the tree. Hence, during a standard AMLS procedure, each treenode will be visited twice except the leaves. The leaves are visited only once since no partition is needed at all. So we can see that the standard AMLS algorithm without forming eigenvectors needs only one pre-postorder combined traversal, i.e. {7,3,1,2,3,6,4,5,6,7} for Fig.2.1.

2.5. Eigenvectors of AMLS. To finish this section, we discuss the two computational procedures of eigenvectors of AMLS with different calculation sequences of (2.19).

The first one, a slow procedure, is to compute the projection vectors

$$T = UZ = (\prod_{s=1}^{6} U_s)Z \tag{2.20}$$

Algorithm 1 The standard AMLS algorithm

initialize current addressing substructure s to be the original structure (K, M).
while not all substructures have been addressed, **do**
  if This is the first time to visit s, **then**
    if s can not be partitioned any more (s is a leaf), **then**
      solve constraint modes Ψ_{si} by (2.7a), $\forall i \in \mathcal{A}$.
      solve the sub-eigenproblem (2.11a) to obtain (Λ_s, Z_s).
      do block Gaussian eliminations by (2.7d) and (2.7e).
      do pre-condensation by (2.16).
      mark s has been addressed and reset s to be its parent $\mathcal{P}(s)$.
    else
      partition s into several substructures $\mathcal{C}(s)$.
      reset s to be an unaddressed substructure of $\mathcal{C}(s)$.
    end if
  else
    if all substructures in $\mathcal{C}(s)$ have been addressed, **then**
      solve constraint modes Ψ_{si} by (2.8a), $\forall i \in \mathcal{A}(s)$.
      solve the sub-eigenproblem (2.11b) to obtain (Λ_s, Z_s).
      do block Gaussian eliminations by (2.8d) and (2.8e).
      do pre-condensation by (2.17a).
      compute m_{js} by (2.17b), $\forall j \in \mathcal{C}(s)$.
      update $\tilde{\mu}_{ji}^{[s]}$ by (2.17c), $\forall j \in \mathcal{C}(s)$, $\forall i \in \mathcal{A}(s)$.
      assemble the wanted (K_c, M_c) by Λ_j, Λ_s and m_{js}, $\forall j \in \mathcal{C}(s)$.
      mark s has been addressed and reset s to be its parent $\mathcal{P}(s)$.
    else
      reset s to be an unaddressed substructure of $\mathcal{C}(s)$.
    end if
  end if
end while
Solve the final system (2.18) to obtain eigenvalues Λ_c.

in advance, and then use T to calculate $V = TX_c$. Here T could also be explained as the global transformation matrix that transforms and projects (K, M) from the original space onto the AMLS reduced subspace to get (K_c, M_c). Kaplan gave the algorithm of forming T in detail [20]. If T is obtained, it is possible to directly express either eigenvectors or dynamical responses from the AMLS reduced subspace to the original space.

However, the costs of forming T is relative expensive. It is reported that the computational costs of assembling T is similar with producing M_c [20]. Therefore, we suggest a fast procedure to compute eigenvectors. Notice that $U \in \mathbb{R}^{n\times n}$, $Z \in \mathbb{R}^{n\times n_m}$, $X_c \in \mathbb{R}^{n_m\times n_p}$ and $n \gg n_m > n_p$, hence it is more efficient to compute the eigenvectors

$$\tilde{V} = ZX_c \tag{2.21}$$

in the transformed space, and then transform $\tilde{V}$ back to the original space to obtain

$$V = U\tilde{V} = U_1 \dots U_s\tilde{V}. \tag{2.22}$$

In implementation, both of these two procedures require X_c, all Z_s and Ψ_{si} that are produced during the standard AMLS, $\forall s \in \mathcal{S}_{l=0,1,2}$, $\forall i \in \mathcal{A}(s)$. The efficiency

of the fast procedure, which is also our default choice to form eigenvectors, will be displayed by numerical experiments in Section 4.1.

3. AMLS with Subspace Iterations. As mentioned in the introduction, the accuracy of the computed eigenpairs (Λ_c, V) of AMLS depends on the cutoff frequency ω_{cutoff}. However, the choice of ω_{cutoff} is heuristic. It is selected according to experiences of analysts. If an improper ω_{cutoff} was given, bad eigensolutions may come out and the spent computational resources are possibly valueless.

Hence, we try to establish a procedure that can improve the accuracy of the eigenpairs computed by AMLS, no matter if the ω_{cutoff} is good or not. In this aspect, the subspace iteration approach is a natural choice to be employed. Algorithm 2 presents a practical subspace iteration method [16] that integrates restarts, shifts, Sturm's sequence check and convergence judgment.

Algorithm 2 A practical subspace iteration method

1: determine q, the dimension of subspace.
2: initialize iteration vector $Q^{(0)} \in \mathbb{R}^{N\times q}$.
3: determine the number of maximum iteration step n_k.
4: determine the shift σ.
5: do factorization $K - \sigma M = LDL^T$.
6: do Sturm's sequence check.
7: **for** $k = 1, 2, \ldots, n_k$ **do**
8: M-orthogonalization for $Q^{(k)}$
9: compute $Q^{(k)} = (LDL^T)^{-1}MQ^{(k-1)}$.
10: compute $K_c^{(k)} = (Q^{(k)})^T K Q^{(k)}$, $M_c^{(k)} = (Q^{(k)})^T M Q^{(k)}$.
11: solve $K_c^{(k)} X_c^{(k)} = M_c^{(k)} X_c^{(k)} \Lambda_c^{(k)}$.
12: compute the eigenvectors $V^{(k)} = Q^{(k)} X_c^{(k)}$.
13: remove convergenced eigenvectors from $V^{(k)}$, take the rest of vectors and some random vectors as the new iteration vectors $Q^{(k)}$.
14: **end for**
15: go back to State 1 if not all wanted eigenpairs have been convergenced.

However, the task of this study is only searching for the possibility and efficiency of improving AMLS by use of subspace iterations. Hence, only the basic iteration frame of

$$Q^{(k+1)} = K^{-1}MQ^{(k)} \tag{3.1}$$

is kept for the current work, excluding any shifts and convergence checks at present. Based on this declaration, two subspace iteration methods will be discussed in this section.

3.1. Method A: independent from AMLS. The first computational scheme is simple: take the approximate eigenvectors V of the standard AMLS as the initial iteration vectors $Q^{(0)}$, and then start a normal subspace iteration approach from factorization of the original stiffness K. Obviously, the initial eigenvectors computed by AMLS are the only required data for this scheme, which is independent from AMLS.

Nevertheless, AMLS is designed for computing hundreds or even thousands of eigenvalues, much larger than decades of eigenpairs solved by normal subspace itera-

tion methods. This situation may result in two special characteristics that differ from normal subspace iterations methods.

(a) According to the well known convergence rate of subspace iterations for the i-th eigenpairs proved by Bathe [2],

$$\lambda_i/\lambda_{q+1}, \quad i = 1, \ldots, q, \tag{3.2}$$

the convergence of eigenpairs for AMLS subspace iterations should be faster than normal subspace iteration methods at the lower end of spectrum, because AMLS provides more initial eigenvectors for iterations. In addition, the initial subspace spanned by AMLS eigenvectors is close to the exact subspace. Therefore, it is reasonable to improve eigenpairs of AMLS by subspace iterations within quite a few iteration steps. So we directly give the maximum iteration step n_k in advance.

(b) The computational costs for matrix-vector multiplications, inner products of vectors and linear equation solutions in a single iteration step will be much higher than normal subspace iteration methods. Thus, in order to save costs, we do not form nor solve the reduced problem $(K_c^{(k)}, M_c^{(k)})$ in each iteration step until all iterations finish. This treatment is workable because the solution of $(K_c^{(k)}, M_c^{(k)})$ is useful for convergence check and have nothing to do with the convergence rate.

Considering above two special properties, we presented Method A, the subspace iterations independent from AMLS in Algorithm 3.

Algorithm 3 Method A: The subspace iterations independent from AMLS.

Require: eigenvectors V by AMLS and the maximum iteration number n_k.

1: initialize the iteration vector $Q^{(0)} = V$.
2: do triangular factorization $K = LL^T$ or $K = LDL^T$.
3: **for** $k = 1, 2, \ldots, n_k$ **do**
4: compute $R = MQ^{(k-1)}$.
5: solve $Q^{(k)}$: $KQ^{(k)} = R$ by the factorization of K.
6: **end for**
7: reduce stiffness matrix by $K_c^{(n_k)} = R^T Q^{(n_k)}$.
8: reduce mass matrix by $M_c^{(n_k)} = (Q^{(n_k)})^T M Q^{(n_k)}$.
9: solve $K_c^{(n_k)} X_c^{(n_k)} = M_c^{(n_k)} X_c^{(n_k)} \Lambda_c^{(n_k)}$ to obtain improved eigenvalues $\Lambda_c^{(n_k)}$.
10: compute improved eigenvectors $V^{(n_k)} = Q^{(n_k)} X_c^{(n_k)}$.

3.2. Method B: cooperated with AMLS. Except for initial eigenvectors, Method A does not utilize any other AMLS data for subspace iterations. The main computational costs of Method A consist of the triangular factorization, matrix-vector multiplications, inner products of vectors and linear equation solutions.

Now we propose an alternative subspace iteration scheme for AMLS. Instead of K, this new scheme takes advantage of the transformed stiffness matrix $\tilde{K}$, which is diagonal block, for subspace iterations, i.e.

$$Q^{(k+1)} = U\tilde{K}^{-1}U^T M Q^{(k)}. \tag{3.3}$$

where $\tilde{K} = U^T K U$ and U is the AMLS transformed matrix. The corresponding Method B of (3.3) is given in Algorithm 4. Our research will show that the efficiency of Method B is higher than Method A, although the forward and backward tranformations are involved in (3.3).

Algorithm 4 Method B: The subspace iterations cooperated with AMLS.

Require: transformed eigenvectors $\tilde{V}$, the transformed stiffness matrix $\tilde{K}$ and the transformation matrix U by AMLS; the maximum iteration number n_k.

1: initialize the iteration vector $\tilde{Q}^{(0)} = \tilde{V}$.
2: **for** $k = 1, 2, \ldots, n_k$ **do**
3: transform backward $Q^{(k-1)} = U\tilde{Q}^{(k-1)}$.
4: compute $R = MQ^{(k-1)}$.
5: transform forward $\tilde{R} = U^T R$.
6: solve $\tilde{Q}^{(k)}$: $\tilde{K}\tilde{Q}^{(k)} = \tilde{R}$.
7: **end for**
8: reduce stiffness matrix by $K_c^{(n_k)} = \tilde{R}^T\tilde{Q}^{(n_k)}$.
9: transform backward $Q^{(n_k)} = U\tilde{Q}^{(n_k)}$.
10: reduce mass matrix by $M_c^{(n_k)} = (Q^{(n_k)})^T M Q^{(n_k)}$.
11: solve $K_c^{(n_k)} X_c^{(n_k)} = M_c^{(n_k)} X_c^{(n_k)} \Lambda_c^{(n_k)}$ to obtain improved eigenvalues $\Lambda_c^{(n_k)}$.
12: compute improved eigenvectors $V^{(n_k)} = Q^{(n_k)} X_c^{(n_k)}$.

Here we emphasize two reasons for not using the transformed matrix $\tilde{M}$ and

$$\tilde{Q}^{(k+1)} = \tilde{K}^{-1}\tilde{M}\tilde{Q}^{(k)} \tag{3.4}$$

for the subspace iterations: (a) as mentioned in Section 2.2 and 2.3, the transformed $\tilde{M} = U^T M U$ has never been explicitly assembled in whole nor in parts, but M and U are known because U is necessary to form the final eigenvectors due to (2.19); (b) the original mass matrix M is sparse; while even if $\tilde{M}$ was formed explicitly, those dense blocks $\tilde{M}_{ij}$ also require a lot of extra storage to save.

3.3. Implementation of Method B. The subspace iteration Method B utilizes the transformed stiffness matrix $\tilde{K}$ and the transformation matrix U of AMLS, both of which are stored in distributed blocks, $\tilde{K}_{ss}$ and Ψ_{si}, corresponding to the treenode index s ($i \in \mathcal{A}(s)$) of the AMLS partition tree. Therefore, like the standard AMLS, the implementation of Algorithm 4 must be performed on several traversals of the AMLS tree. That is the topic of this subsection.

First of all, remember that the block Gauss eliminations, $\tilde{K} = U^T K U$, are performed on a postorder traversal of the AMLS tree, i.e. {1,2,3,4,5,6,7} for Fig.2.1. Thus, the inverse operation that transforms $\tilde{Q}^{(k-1)}$ back to $Q^{(k-1)}$ in State 3, Algorithm 4 could be executed by

$$(\tilde{Q}^{(k-1)})^{[s-1]} = U_s(\tilde{Q}^{(k-1)})^{[s]} \tag{3.5}$$

on an preoder traversal, i.e. {7,6,5,4,3,2,1} or {7,3,1,2,6,4,5} for Fig.2.1. For convenience, we ignore the superscript $[s]$ in (3.5) to simplify the notation of $(\tilde{Q}^{(k-1)})^{[s-1]}$ and $(\tilde{Q}^{(k-1)})^{[s]}$ as $Q^{(k-1)}$ and $\tilde{Q}^{(k-1)}$, respectively. Recall the block structure of U_s in (2.5), then (3.5) can be rewritten as

$$Q_s^{(k-1)} = \tilde{Q}_s^{(k-1)} + \sum_i \Psi_{si}\tilde{Q}_i^{(k-1)},\ Q_i^{(k-1)} = \tilde{Q}_i^{(k-1)},\ \ \forall i \in \mathcal{A}(s), \tag{3.6}$$

where $Q_i^{(k-1)}$ denotes the i-th block of $Q^{(k-1)}$, i.e.

$$Q^{(k-1)} = (\ (Q_1^{(k-1)})^T, (Q_2^{(k-1)})^T,\ \ldots\ , (Q_7^{(k-1)})^T\ )^T.$$

Similarly, the forward transformation in State 5, Algorithm 4 can be computed by

$$\tilde{R}_s = R_s,\ \tilde{R}_i = R_i + \Psi_{si}^T R_s, \quad \forall i \in \mathcal{A}(s), \tag{3.7}$$

on a postorder traversal of the AMLS tree.

In this way, the traversal implementation of the iteration body of Method B (State 3 to State 6 of Algorithm 4) are presented in Algorithm 5, as well as the computation of K_c on State 8 of Algorithm 4. Here K_c is formed on the last iteration step in order to take advantage of the block diagonal property of $\tilde{K}$. The rest of states, State 9 and 10 of Algorithm 4, could be done by an extra traversal of the AMLS partition tree.

Algorithm 5 The traversal implementation of the iteration body for Method B.

1: set current addressing substructure s to be the root of AMLS tree.
2: initialize $R = 0$ and $K_c = 0$.
3: **while** not all substructures have been addressed, **do**
4: **if** This is the first time to visit s, **then**
5: transform backward $Q^{(k-1)} = U_s \tilde{Q}^{(k-1)}$ by (3.6).
6: compute the contributions of s for $R = MQ^{(k-1)}$ by

$$R_s = R_s + M_{ss} Q_s^{(k-1)} + \sum_i M_{si} Q_i^{(k-1)}, \qquad \forall i \in \mathcal{A}(s), \tag{3.8a}$$

$$R_i = R_i + M_{si}^T Q_s^{(k-1)}, \qquad \forall i \in \mathcal{A}(s). \tag{3.8b}$$

7: **end if**
8: **if** s is a leaf **or** all substructures of $\mathcal{C}(s)$ have been addressed, **then**
9: transform forward $\tilde{R} = U_s^T R$ by (3.7).
10: solve $\tilde{Q}_s^{(k)}$: $\tilde{K}_{ss} \tilde{Q}_s^{(k)} = R_s$.
11: **if** $k = n_k$, **then**
12: compute $K_c = K_c + \tilde{R}_s \tilde{Q}_s^{(k)}$.
13: **end if**
14: mark s has been addressed and reset s to be its parent $\mathcal{P}(s)$.
15: **else**
16: reset s to be an unaddressed substructure of $\mathcal{C}(s)$.
17: **end if**
18: **end while**

4. Numerical experiments. In this section, four typical problems with different scales are selected to test the efficiency of the AMLS with subspace iteration improvements. A brief description of these problems is given in Table 4.1, in which $n_z(A)$ represents the number of non-zeros of matrix A.

All numerical experiments were tested on a Linux platform with an Intel Pentium D CPU(3.46GHz, 4 Cores) and 7.7GB Memory. The two subspace iteration methods described in Section 3 were coded based on the serial version of Elssel's AMLS implementation [12]. It employs the METIS [21] and the Intel Math Kernel Library v10.3 optimized LAPACK to give a high performance of computation. Table 4.2 lists the required disk storage for matrices that are used in AMLS and subspace iterations.

4.1. Computation of eigenvectors. First of all, we list the computational time for the standard AMLS and the computation of eigenvectors of the four problems

TABLE 4.1
Four test problems and their descriptions

Problems	Description	n	$n_z(K)$	$n_z(M)$
bcsstk24	Sports arena[11]	3,562	159,910	3,562
wtaoc01k	Blade of a 50KW wind turbine[26]	23,328	2,220,068	814,694
wtnh01k	Blade of a 1.5MW wind turbine[1]	117,990	11,243,248	5,590,256
bs6_k	Structure of connected beams[12]	517,161	20,727,171	6,909,057

TABLE 4.2
Required storage for useful matrices of four problems

Examples	K	M	$\tilde{K} = U^T K U$	U
bcsstk24	1.0MB	0.3MB	1.7MB	3.5MB
wtaoc01k	12.8MB	4.8MB	23.3MB	57.9MB
wtnh01k	64.9MB	32.5MB	126.4MB	374.2MB
bs6_k	121.1MB	42.0MB	393.3MB	1454.8MB

in Table 4.3. Here both of the two computation procedures mentioned in section 2.5 are implemented to give a comparison. These two procedures calculate exactly the same eigenvectors according to basic linear algebra knowledge. However, Table 4.3 show that the fast method presented in (2.21) and (2.22) becomes more efficient as the scale of problem increases. So we set the fast method as the default strategy in our code to compute the initial eigenvectors of AMLS for the further subspace iterations.

4.2. Modal errors of eigenpairs. Before discussions of any improvement of subspace iterations, let us review the error of eigenpairs that are produced by the standard AMLS algorithm. The relative errors between eigenvalues of AMLS and eigenvalues of some classical algorithms or codes, e.g. **eigs** of Matlab, have been reported in [8], etc. Generally speaking, if a proper cutoff frequency ω_{cutoff} is selected, the relative errors are expected to have orders of 10^{-4} to 10^{-2} for all computed eigenvalues.

Errors of eigenvectors of AMLS are rarely given in the literature. In this paper, one kind of residual errors suggested by Bathe [3], or called as *modal error*, is used to evaluate the error of each eigenpair (λ, x) of the original problem (2.1), i.e.

$$\epsilon = \frac{\|Kx - \lambda Mx\|_2}{\|\lambda Mx\|_2}. \tag{4.1}$$

Fig.4.1 provides the modal errors of computed eigenpairs of the four problems. For each problem, two cutoff eigenvalues λ_{cutoff} are chosen to give a comparison. From Fig.4.1, we can see that modal errors are slightly reduced, although the second cutoff eigenvalues (modal errors denoted by lines) are set 10 times larger than the first ones (modal errors denoted by dots). In addition, not a single eigenpair has a modal error less than 10^0. But for many classical eigensolution methods, e.g. the block Lanczos algorithm [17], modal errors can be as low as 10^{-14}. It implies that there should exist a big potential to improve AMLS, especially for eigenvectors.

4.3. Performance of AMLS with subspace iterations. For the AMLS with subspace iterations presented in this paper, the standard AMLS could be seen as a pre-conditioner that provides a good initial subspace. Then in the further iteration

TABLE 4.3
Elapsed time for computing eigenvectors of AMLS

Examples	$\lambda_{\text{cutoff}}(\text{rad}^2/\text{s}^2)$	n_p	t_{AMLS}	t_{eigvec}(Slow)	t_{eigvec}(Fast)
bcsstk24	1×10^3	201	0.78s	0.21s	0.05s
wtaoc01k	1×10^7	144	11.66s	4.63s	0.46s
wtnh01k	1×10^6	350	100.21s	196.78s	7.60s
bs6_k	3×10^{10}	300	308.39s	Out of memory	33.08s

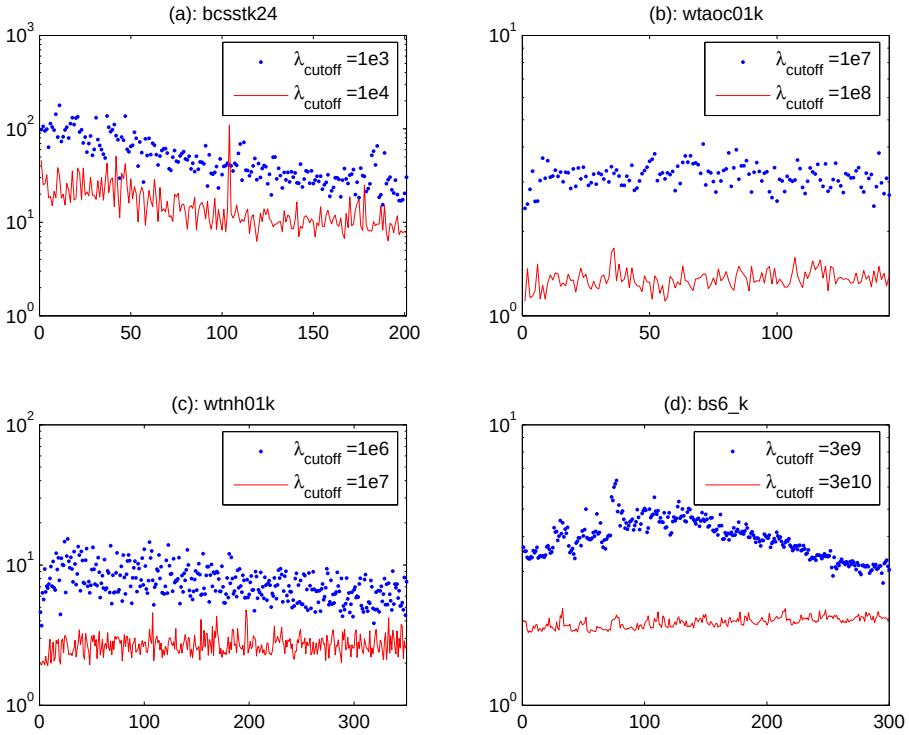

FIG. 4.1. *Modal errors of four problems (Unit of λ_{cutoff}: rad^2/s^2)*

refinements, both two subspace iteration procedures, Method A and Method B, calculate improved eigenpairs with the same precision. The reduced modal errors after subspace iterations are plotted in Fig.4.2. It demonstrates that even one or two iteration steps can significantly improve the accuracy of eigenpairs, especially at the lower end of the spectrum. This situation can be explained by (3.2), the convergence rate of the subspace iteration algorithm.

Besides eigenvectors, eigenvalues of AMLS were also enhanced considerably. Please see Fig.4.3 as an example. Here the Matlab function **eigs** was employed to compute the exact eigenvalues of this problem. Comparing Fig.4.3 (a) with (b), we find that if a higher cutoff frequency ω_{cutff} is chosen, the subspace iterations also perform much better than a lower ω_{cutff}, like the standard AMLS.

4.4. Time costs of AMLS with subspace iterations. To show the efficiency of the proposed subspace iteration Method B, we list the consumed time of Method B ($t_{\text{SI-B}}$) and Method A ($t_{\text{SI-A}}$) with only one iteration step in Table 4.4. Here the

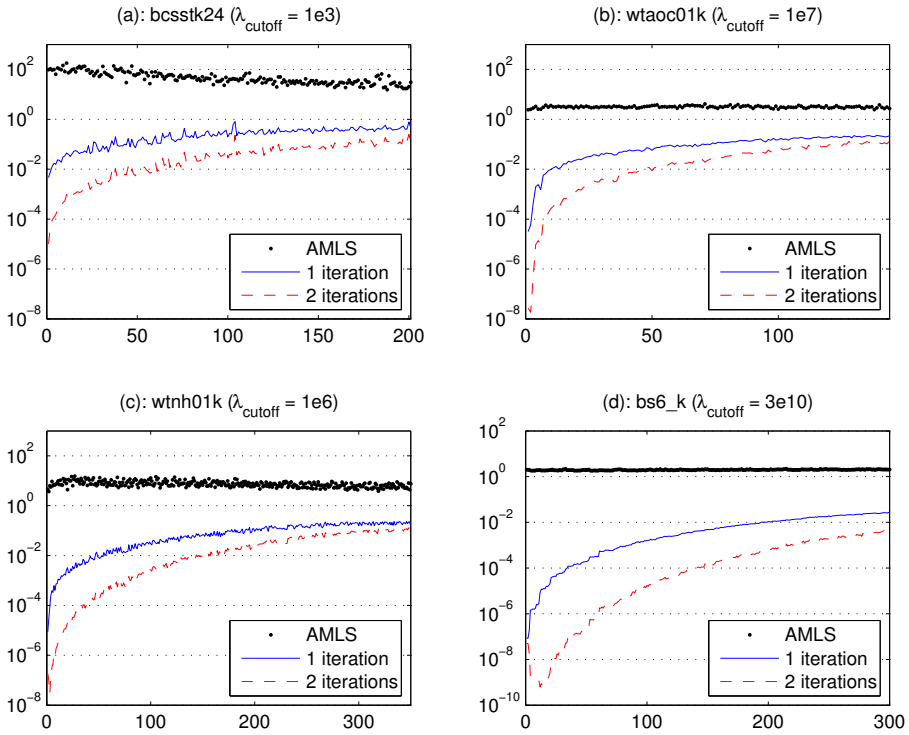

FIG. 4.2. *Modal errors of four problems with subspace iterations (Unit of* λ_{cutoff}*:* rad^2/s^2*)*

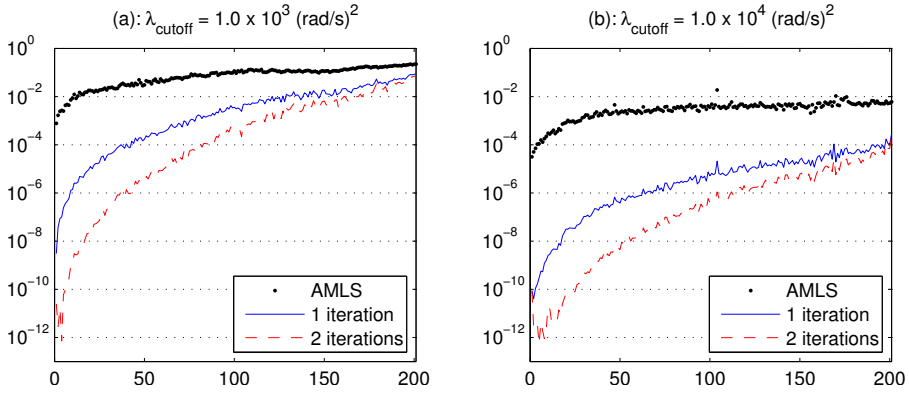

FIG. 4.3. *Relative errors of eigenvalues (Problem: bcsstk24)*

factorization (State 2, Algorithm 3) and solution (State 5, Algorithm 3) of Method A are implemented by the Intel Math Kernel Library v10.3 optimized PARDISO package to give a high performance computation. Since the factorization is performed only once, its time cost ($t_{\text{SI-A}}^{\text{Factorization}}$) is separated from $t_{\text{SI-A}}$. In order to give a clear comparison, the consumed time of the standard AMLS (t_{AMLS}) and the fast eigenvector computation method (t_{eigvec}) are included in Table 4.4 as well.

The last column of Table 4.4, $t_{\text{SI-B}}/t_{\text{SI-A}}$, displays the efficiency of the proposed subspace iteration Method B. Remember that the scale of these problems is increasing

TABLE 4.4
Elapsed time comparison of Method A and B with a single subspace iteration step.

Examples	n_p	t_{AMLS}	t_{eigvec}	$t_{\text{SI-B}}$	$t_{\text{SI-A}}^{\text{Factorization}}$	$t_{\text{SI-A}}$	$t_{\text{SI-B}}/t_{\text{SI-A}}$
bcsstk24	201	0.78s	0.05s	0.53s	0.56s	0.75s	71%
wtaoc01k	144	11.66s	0.46s	4.35s	1.28s	7.70s	56%
wtnh01k	350	100.21s	7.60s	69.55s	7.99s	167.03s	42%
bs6_k	300	308.39s	33.08s	210.07s	27.17s	385.71s	54%

from bcsstk24 to bs6_k as Table 4.1 shows. So in general, Method B is more efficient than Method A for larger problems. In table 4.4, the only exception of this conclusion appears between problems wtnh01k and bs6_k. In fact, according to Table 4.1, Problem wtnh01k has around 95.3 non-zeros per row for K and 47.4 non-zeros per row for M. But for Problem bs6_k, these two numbers are 40.0 and 13.4, respectively. In other words, wtnh01k is a more complicated structure compared with bs6_k, so that the matrix pencil (K, M) of bs6_k is more sparse than wtnh01k. This situation implies that the proposed Method B performs even better for those more complicated structures.

5. Conclusions. In this paper, the standard AMLS algorithm for linear eigenvalue problems is reviewed, including discussions of two eigenvector computational procedures. Then based on the fast eigenvector assembly, two subspace iteration methods are presented to improve eigenpairs of the standard AMLS. The proposed Method A is independent from AMLS, while Method B takes advantage of the transformed block diagonal stiffness matrix to reduce the computational costs of subspace iterations.

Our numerical experiments via four problems with different scales evaluate that even one or two subspace iterations can significantly reduce modal errors of the AMLS eigenpairs at the lower end of spectrum. The comparison of consumed time between AMLS subspace iteration Method A and B demonstrate that Method B is more efficient than Method A, especially for larger and more complicated problems.

Finally, we conclude that the proposed subspace iteration Method B successfully improve eigenpairs of AMLS with acceptable time costs. It should be possible to develop a more advanced AMLS subspace iteration method that integrates Sturm's sequence check, convergence criteria and even shifts based on the study of this paper. In addition, since Method B is implemented by traversals on the partition tree similarly with the standard AMLS algorithm, we believe that Method B could also be parallelized for much larger problems.

Acknowledgments. This work was done in Hamburg, Germany when the first author was visiting the Institute of Numerical Simulation, Hamburg University of Technology. The first author would like to thank Dr. Kolja Elssel, Dr. Markus Stammberger and Dr. Duy Nam Le for their constructive directions of coding AMLS and many other useful discussions, as well as the host of Institute of Numerical Simulation.

REFERENCES

[1] *Aerodynamics design of nh1500 wind turbine blade and its performance verifications*, tech. report, Nanjing University of Aeronautics and Astronauticse, Nanjing, Jiangsu, China, 2010.

[2] K. J. Bathe, *Formulations and Computational Algorithms in Finite Element Analysis*, M.I.T. Press, Cambridge, Massachusetts, UK, 1977, ch. Convergence of Subspace Iterations, pp. 575–598.

[3] ———, *Finite Element Procedure*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA, 1996.

[4] K. J. Bathe and S. Ramaswamy, *An accelerated subspace iteration method*, Computer Methods in Applied Mechanics and Engineering, 23 (1980), pp. 313 – 331.

[5] K. J. Bathe and E. L. Wilson, *Large Eigenvalue Problems in Dynamic Analysis*, Journal of the Engineering Mechanics Division, 98 (1972), pp. 1471–1485.

[6] C. Bekas and Y. Saad, *Computation of smallest eigenvalues using spectral schur complements*, SIAM J. Sci. Comput., 27 (2005), pp. 458–481.

[7] J. K. Bennighof and C. K. Kim, *An adaptive multi-level substructuring method for efficient modeling of complex structures*, in Proceedings of the AIAA 33rd SDM Conference, Dallas, Texas, USA, 1992, pp. 1631–1639.

[8] J. K. Bennighof and R. B. Lehoucq, *An automated multilevel substructuring method for eigenspace computation in linear elastodynamics*, SIAM J. Sci. Comput., 25 (2004), pp. 2084–2106.

[9] M. Clint and A. Jenning, *The evaluation of eigenvalues and eigenvectors of real symmetric matrices by simultaneous iteration*, The Computer Journal, 13 (1970), pp. 76–80.

[10] R. R. Craig and M. C. C. Bampton, *Coupling of substructures for dynamic analysis*, AIAA Journal, 6 (1968), pp. 1313–1319.

[11] I. S. Duff, R. Grimes, and J. Lewis, *The user's guide for the harwell-boeing sparse matrix collection (release i)*, tech. report, CERFACS, Technical Report TR/PA/92/86, Lyon, France, 1992.

[12] K. Elssel, *Automated Multilevel Substructuring for Nonlinear Eigenvalue Problems*, PhD thesis, Hamburg University of Technology, 2006.

[13] K. Elssel and H. Voss, *An a priori bound for automated multilevel substructuring*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 386–397.

[14] ———, *Reducing huge gyroscopic eigenproblems by automated multi-level substructuring*, Archive of Applied Mechanics, 76 (2006), pp. 171–179.

[15] ———, *Reducing sparse nonlinear eigenproblems by automated multi-level substructuring*, Advances in Engineering Software, 39 (2008), pp. 828–838.

[16] Y. Gong, H. Zhou, P. Chen, and M. Yuan, *Comparison of subspace iteration, iterative Ritz vector method and iterative Lanczos method*, Journal of Vibration Engineering, 18 (2005), pp. 227–232. (in Chinese).

[17] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 15 (1994), pp. 228–272.

[18] B. Hendrickson and R. Leland, *The chaco user's guide: Version 2.0*, tech. report, Sandia National Labs, Tech. Report SAND94-2692, Albuquerque, NM, June 1995.

[19] W. C. Hurty, *Vibrations of structural systems by component-mode synthesis*, Journal of the Engineering Mechanics Division, ASCE, 86 (1960), pp. 51–69.

[20] M. F. Kaplan, *Implementation of automated multilevel substructuring for frequency response analysis of structures*, PhD thesis, The University of Texas at Austin, 2001.

[21] G. Karypis and V. Kumar, *Metis - a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. version 4.0*, tech. report, University of Minnesota, Minneapolis, 1998.

[22] J. H. Ko, S. N. Jung, D. Byun, and Z. Bai, *An algebraic substructuring using multiple shifts for eigenvalue computations*, Journal of Mechanical Science and Technology, 22 (2008), pp. 440–449.

[23] A. Kropp and D. Heiserer, *Efficient broadband vibro-acoustic analysis of passenger car bodies using an fe-based component mode synthesis approach*, Journal of Computational Acoustics, 11 (2003), pp. 139–157.

[24] D. N. Le, *Extending Deterministic Vibration Analysis of Ships into the Medium Frequency Range*, PhD thesis, Hamburg University of Technology, 2009.

[25] B. S. Liao, Z. Bai, and W. Gao, *The important modes of subsystems: A moment-matching approach*, International Journal for Numerical Methods in Engineering, 70 (2007), pp. 1581–1597.

[26] L. R. McKittrick, D. S. Cairns, J. Mandell, D. C. Combs, D. Rabern, and R. D. Van Luchene, *Analysis of a composite blade design for the aoc 15/50 wind turbine using a finite element model*, tech. report, Sandia National Laboratories, SAND2001-1441, Albuquerque, NM, USA, 2001.

[27] W. Rachowicz and A. Zdunek, *Automated multi-level substructuring (amls) for electromagnetics*, Computer Methods in Applied Mechanics and Engineering, 198 (2009), pp. 1224 – 1234.

[28] M. Stammberger and H. Voss, *Automated multi-level sub-structuring for fluid-solid interaction problems*, Numerical Linear Algebra with Applications, 18 (2011), pp. 411–427.

[29] E. L. Wilson and T. Itoh, *An eigensolution strategy for large systems*, Computers & Structures, 16 (1983), pp. 259 – 265.

[30] C. Yang, W. Gao, Z. Bai, X. S. Li, L. Q. Lee, P. Husbands, and E. Ng, *An algebraic substructuring method for large-scale eigenvalue calculation*, SIAM J. Sci. Comput., 27 (2005), pp. 873–892.

[31] ———, *Algebraic sub-structuring for electromagnetic applications*, Lecture Notes in Computer Science, 3732 (2006), pp. 364 – 373.

[32] Q. C. Zhao, P. Chen, W. B. Peng, Y. C. Gong, and M. W. Yuan, *Accelerated subspace iteration with aggressive shift*, Computers & Structures, 85 (2007), pp. 1562 – 1578.