Computers and Structures 186 (2017) 11–21

Contents lists available at ScienceDirect

Computers and Structures

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

The Bathe subspace iteration method enriched by turning vectors

Ki-Tae Kim, Klaus-Jürgen Bathe *

Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA

ARTICLE INFO

Article history:
Received 29 December 2016
Accepted 19 February 2017
Available online 29 March 2017

Keywords:
Structural dynamics
Frequencies and mode shapes
Finite element method
Bathe subspace iteration
Large eigenvalue problems
Parallel processing

ABSTRACT

We present a novel extension of the Bathe subspace iteration method for the solution of the generalized eigenvalue problem in structural dynamics. The key idea is to enrich the subspace by using some turning vectors to replace current iteration vectors. The turning vectors are evaluated from the subspace of the current iteration. The scheme is simple and a considerable improvement in computational efficiency is achieved. A simplified convergence analysis is given and the results of some example solutions show the effectiveness of the method.

1. Introduction

An essential step in structural dynamics is to evaluate some frequencies and mode shapes of the structure considered, and in today's analyses very large finite element systems are solved. Hence extensive efforts have been made to establish effective eigensolution techniques. The Bathe subspace iteration method [1–4] is a successful scheme and has been abundantly employed in research and industry because of its robustness and effectiveness.

An attractive ingredient of the Bathe subspace iteration method is that it is particularly amenable to parallel implementations. As discussed in Ref. [5], by partitioning the iteration vectors, most computations in the method can be programmed in shared and distributed memory processing and a linear increase in solution time with the number of eigenpairs sought is achieved.

Another widely used method is the Lanczos method [4,6]. Although the method initially suffered from numerical instabilities resulting from the loss of orthogonality, techniques have been proposed to overcome this deficiency and at present variations of the Lanczos method enjoy much success, see for example [7–9]. Using this method, the computational cost increases almost linearly proportional to the number of eigenpairs sought, which makes the method also efficient when many eigenpairs are needed. However, since the algorithmic steps in the solution are sequential, the technique is not directly suitable to parallel processing.

Since the Bathe subspace iteration method is widely used and particularly suited to parallel processing, it is of great interest to speed up the iterations. This speed-up should be achieved even without parallel processing and should be present in particular when seeking many eigenpairs (typically more than 100 pairs) of very large eigenvalue problems. In fact, single processor solutions provide a good test for basic increases in efficiency, and any significant decrease in solution times is very valuable.

In this paper, we present a novel algorithm to accelerate the Bathe subspace iteration method. In the basic method, the subspace iteration vectors turn in each iteration a certain amount towards the required subspace vectors [4,5]. The fundamental idea for accelerating the iterations is to use the direction of turning of the subspace in the iterations. This is achieved by establishing new iteration vectors denoted as "forward turning vectors" to replace iteration vectors that are much less useful. This enrichment of the iterations is a simple addition to the basic Bathe subspace iteration method, and yields a considerable reduction in computational cost.

In the following sections, we first describe the algorithm, referred to as the "enriched subspace iteration method". Then we provide a simplified convergence analysis to give some insight into how the new vectors accelerate the iterations. Finally, we give the results of some example solutions to demonstrate the performance of the scheme.

2. The enriched subspace iteration method

We consider the generalized symmetric eigenvalue problem

$$\mathbf{K}\boldsymbol{\varphi} = \lambda\mathbf{M}\boldsymbol{\varphi} \tag{1}$$

* Corresponding author.
E-mail address: kjb@mit.edu (K.J. Bathe).

http://dx.doi.org/10.1016/j.compstruc.2017.02.006
0045-7949/

where $\mathbf{K}$ and $\mathbf{M}$ are the stiffness matrix and the consistent mass matrix of a finite element system with n degrees of freedom. The matrices $\mathbf{K}$ and $\mathbf{M}$ are assumed (without loss of generality [4]) to be positive definite. We seek the smallest p eigenvalues and corresponding eigenvectors $(\lambda_i, \boldsymbol{\varphi}_i)$; $i = 1, \ldots, p$, with the ordering

$$0 < \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_p, \tag{2}$$

which satisfy

$$\mathbf{K}\boldsymbol{\varphi}_i = \lambda_i \mathbf{M}\boldsymbol{\varphi}_i \tag{3}$$

and

$$\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{M}\boldsymbol{\varphi}_j = \delta_{ij}, \tag{4}$$

$$\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{K}\boldsymbol{\varphi}_j = \lambda_i \delta_{ij} \tag{5}$$

where δ_{ij} is the Kronecker delta.

In the following we describe the algorithm used to solve for these eigenvalues and eigenvectors.

2.1. The algorithm

We start the enriched subspace iteration method by using first a simple scheme given in Refs. [4,5] to construct q linearly independent vectors in $\mathbf{MX}_0$, with $q > p$, in which the diagonal entries of $\mathbf{M}$, unit vectors that excite the degrees of freedom of the maximum values m_{ii}/k_{ii} and a random vector are used. Usually $q = \max\{2p, p+8\}$. Using the linear independent vectors in $\mathbf{MX}_0$, we perform a single iteration using the basic subspace iteration method to obtain the $\mathbf{M}$-orthonormal starting iteration vectors in $\mathbf{X}_1$, which span the q-dimensional subspace E_1, and makes it possible to directly use the enrichment algorithm.

– The 10 steps in each iteration

For each iteration $k = 1, 2, \ldots$, the following steps 1–10 are performed until convergence is established to the p required eigenvalues. Steps 1–5 include the enrichment by establishing and using the 'forward turning vectors' and steps 6–10 are as in the basic subspace iteration method.

1. For $k = 1, 2, \ldots$, partition the iteration vectors which span E_k

$$\mathbf{X}_k = [\boldsymbol{\Phi}_k, \mathbf{X}_k^a, \mathbf{X}_k^b] \tag{6}$$

where $\boldsymbol{\Phi}_k$ stores the p_k vectors which have converged to the required tolerance in the previous iterations, with $p_1 = 0$, and the rest of the iteration vectors are equally partitioned into $\mathbf{X}_k^a$ and $\mathbf{X}_k^b$, which are both of order $n \times r_k$. Hence we use $r_k = (q - p_k)/2$.

2. Evaluate $\overline{\mathbf{X}}_{k+1}^a$

$$\mathbf{K}\overline{\mathbf{X}}_{k+1}^a = \mathbf{M}\mathbf{X}_k^a. \tag{7}$$

3. Construct $\mathbf{Y}_k$ of order $n \times r_k$ using the following steps in part (a) and thereafter part (b):

(a) Calculate in reverse order the amount of turning of the iteration vectors in $\overline{\mathbf{X}}_{k+1}^a$ and choose the vectors for which the measure used is larger than the tolerance *tolt*, i.e., for $i = r_k, r_k - 1, \ldots, 2, 1$, with $t_k = 0$, evaluate in this part (a):

$$\widehat{\mathbf{x}}_i = \overline{\mathbf{x}}_i^{a(k+1)} - \mathbf{X}_k\left(\mathbf{X}_k^{\mathrm{T}}\mathbf{M}\overline{\mathbf{x}}_i^{a(k+1)}\right) - \sum_{j=1}^{t_k}\mathbf{u}_j\left(\mathbf{u}_j^{\mathrm{T}}\mathbf{M}\overline{\mathbf{x}}_i^{a(k+1)}\right) \tag{8}$$

where $\overline{\mathbf{x}}_i^{a(k+1)}$ is the ith column vector in $\overline{\mathbf{X}}_{k+1}^a$ and the last term is only included if $t_k \geqslant 1$. Then evaluate

$$\alpha_i = \frac{\widehat{\mathbf{x}}_i^{\mathrm{T}}\mathbf{M}\widehat{\mathbf{x}}_i}{(\overline{\mathbf{x}}_i^{a(k+1)})^{\mathrm{T}}\mathbf{M}\overline{\mathbf{x}}_i^{a(k+1)}}. \tag{9}$$

If $\alpha_i \leqslant tolt$ go to the next value of i. If $\alpha_i > tolt$ we proceed as follows

$$t_k = t_k + 1, \tag{10}$$

$$\mathbf{u}_{t_k} = \frac{\widehat{\mathbf{x}}_i}{\sqrt{\widehat{\mathbf{x}}_i^{\mathrm{T}}\mathbf{M}\widehat{\mathbf{x}}_i}}, \tag{11}$$

$$\mathbf{v}_{t_k} = \overline{\mathbf{x}}_i^{a(k+1)} \tag{12}$$

and now go to the next value of i.

(b) Next, let t_k be the last value reached in the above loop, part (a). We now construct $\mathbf{Y}_k = \left[\mathbf{x}_1^{b(k)}, \ldots \mathbf{x}_{r_k - t_k}^{b(k)}, \mathbf{v}_1, \ldots, \mathbf{v}_{t_k}\right]$ where $\mathbf{x}_i^{b(k)}$, $i = 1, \ldots, r_k - t_k$, are the first $r_k - t_k$ column vectors in $\mathbf{X}_k^b$. We denote the ith column vector in $\mathbf{Y}_k$ by $\mathbf{y}_i^{(k)}$. Calculate for $i = 1, \ldots, t_k$

$$\widetilde{\mathbf{x}}_i = \mathbf{v}_i - \mathbf{X}_k^a\left((\mathbf{X}_k^a)^{\mathrm{T}}\mathbf{M}\mathbf{v}_i\right) - \boldsymbol{\Phi}_k(\boldsymbol{\Phi}_k^{\mathrm{T}}\mathbf{M}\mathbf{v}_i) - \sum_{j=1}^{r_k - t_k + i - 1}\mathbf{y}_j^{(k)}\left((\mathbf{y}_j^{(k)})^{\mathrm{T}}\mathbf{M}\mathbf{v}_i\right), \tag{13}$$

$$\mathbf{y}_{r_k - t_k + i}^{(k)} = \frac{\widetilde{\mathbf{x}}_i}{\sqrt{\widetilde{\mathbf{x}}_i^{\mathrm{T}}\mathbf{M}\widetilde{\mathbf{x}}_i}}. \tag{14}$$

Here, we call $\widetilde{\mathbf{x}}_i$, $i = 1, \ldots, t_k$, and their normalizations by Eq. (14) the 'turning vectors'.

4. Evaluate $\overline{\mathbf{Y}}_{k+1}$ from

$$\mathbf{K}\overline{\mathbf{Y}}_{k+1} = \mathbf{M}\mathbf{Y}_k \tag{15}$$

where we now have in $\overline{\mathbf{Y}}_{k+1}$ the 'forward turning vectors' which are the key ingredient to obtain a faster convergence.

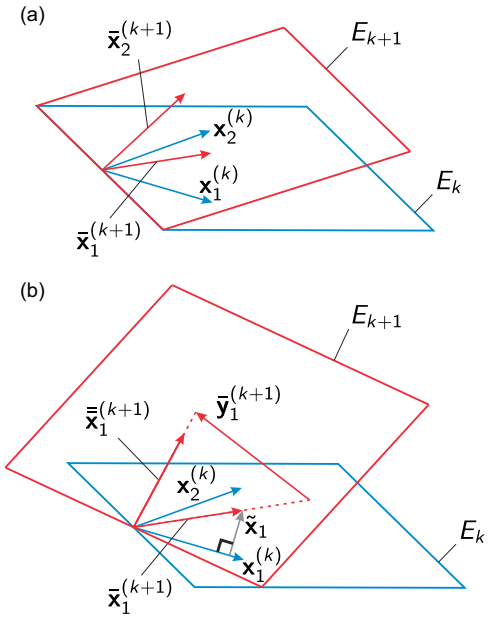

Fig. 1. Geometrical illustration of the subspaces E_k and E_{k+1} when $\mathbf{M} = \mathbf{I}$ and only two iteration vectors are considered: (a) in the basic method and (b) in the enriched method; the red (blue) vectors span the red (blue) plane. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Comparison of operation counts (number of multiplications); the results of the basic method are taken from Refs. [3,4]; for the enriched method, we also consider the algorithm summarized in Appendix A; here m denotes the mean half-bandwidth of $\mathbf{K}$ and $\mathbf{M}$.

Basic subspace iteration method			Enriched subspace iteration method		
Operation	Calculation	Number of operations	Operation	Calculation	Number of operations
Factorization of $\mathbf{K}$	$\mathbf{K}=\mathbf{LDL}^{\mathrm{T}}$	$\frac{1}{2}nm^2+\frac{3}{2}nm$	Factorization of $\mathbf{K}$	$\mathbf{K}=\mathbf{LDL}^{\mathrm{T}}$	$\frac{1}{2}nm^2+\frac{3}{2}nm$
Iteration	$\mathbf{K}\overline{\mathbf{X}}_{k+1}=\mathbf{R}_k$	$nq(2m+1)$	Iteration	$\mathbf{R}_k=\left[\underset{n\times p_k}{\boldsymbol{\Psi}_k},\underset{n\times r_k}{\mathbf{R}_k^a},\underset{n\times r_k}{\mathbf{R}_k^b}\right]$	
	$\mathbf{K}_{k+1}=\overline{\mathbf{X}}_{k+1}^{\mathrm{T}}\mathbf{R}_k$	$\frac{1}{2}nq(q+1)$		$\mathbf{K}\overline{\mathbf{X}}_{k+1}^a=\mathbf{R}_k^a$	$nr_k(2m+1)$
	$\overline{\mathbf{R}}_{k+1}=\mathbf{M}\overline{\mathbf{X}}_{k+1}$	$nq(2m+1)$		$[\mathbf{D}_{k+1}^{\mathrm{T}},\mathbf{A}_{k+1}^{\mathrm{T}},\mathbf{B}_{k+1}^{\mathrm{T}}]=(\overline{\mathbf{X}}_{k+1}^a)^{\mathrm{T}}\mathbf{R}_k$	$nr_k(p_k+\frac{r_k+1}{2}+r_k)$
	$\mathbf{M}_{k+1}=\overline{\mathbf{X}}_{k+1}^{\mathrm{T}}\overline{\mathbf{R}}_{k+1}$	$\frac{1}{2}nq(q+1)$		$\overline{\mathbf{R}}_{k+1}^a=\mathbf{M}\overline{\mathbf{X}}_{k+1}^a$	$nr_k(2m+1)$
	$\mathbf{K}_{k+1}\mathbf{Q}_{k+1}=\mathbf{M}_{k+1}\mathbf{Q}_{k+1}\boldsymbol{\Lambda}_{k+1}$	$o(q^3)$		$\mathbf{C}_{k+1}=(\overline{\mathbf{X}}_{k+1}^a)^{\mathrm{T}}\overline{\mathbf{R}}_{k+1}^a$	$\frac{1}{2}nr_k(r_k+1)$
	$\mathbf{R}_{k+1}=\overline{\mathbf{R}}_{k+1}\mathbf{Q}_{k+1}$	nq^2		Check the amount of turning (see step (a) in Appendix A)	$o(qr_k^2)$
Total in single iteration		$nq(4m+2q+3)+o(q^3)$			
Sturm sequence check	$\overline{\mathbf{K}}=\mathbf{K}-\mu\mathbf{M}$	$n(m+1)$		Calculate $\mathbf{S}_k(=\mathbf{MY}_k)$ (see step (b) in Appendix A)	$nt_k(q+1)$
	$\overline{\mathbf{K}}=\mathbf{LDL}^{\mathrm{T}}$	$\frac{1}{2}nm^2+\frac{3}{2}nm$		$\mathbf{K}\overline{\mathbf{Y}}_{k+1}=\mathbf{S}_k$	$nr_k(2m+1)$
				$\mathbf{K}_{k+1}=\begin{bmatrix}\boldsymbol{\Lambda}_k & & \text{sym.}\\ \mathbf{0} & \mathbf{A}_{k+1} & \\ \mathbf{0} & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}}\mathbf{R}_k^a & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}}\mathbf{S}_k\end{bmatrix}$	$nr_k(t_k+\frac{r_k+1}{2})$
				$\overline{\mathbf{R}}_{k+1}^b=\mathbf{M}\overline{\mathbf{Y}}_{k+1}$	$nr_k(2m+1)$
				$\mathbf{M}_{k+1}=\begin{bmatrix}\mathbf{I} & & \text{sym.}\\ \mathbf{D}_{k+1}^{\mathrm{T}} & \mathbf{C}_{k+1} & \\ \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}}\boldsymbol{\Psi}_k & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}}\overline{\mathbf{R}}_{k+1}^a & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}}\overline{\mathbf{R}}_{k+1}^b\end{bmatrix}$	$nr_k(p_k+r_k+\frac{r_k+1}{2})$
				$\mathbf{K}_{k+1}\mathbf{Q}_{k+1}=\mathbf{M}_{k+1}\mathbf{Q}_{k+1}\boldsymbol{\Lambda}_{k+1}$	$o(q^3)$
				$\mathbf{R}_{k+1}=[\boldsymbol{\Psi}_k,\overline{\mathbf{R}}_{k+1}^a,\overline{\mathbf{R}}_{k+1}^b]\mathbf{Q}_{k+1}$	nq^2
			Total in single iteration		$2nr_k(4m+q+3)+nq^2+nt_k(q+r_k+1)+o(q^3)$
			Sturm sequence check	$\overline{\mathbf{K}}=\mathbf{K}-\mu\mathbf{M}$	$n(m+1)$
				$\overline{\mathbf{K}}=\mathbf{LDL}^{\mathrm{T}}$	$\frac{1}{2}nm^2+\frac{3}{2}nm$

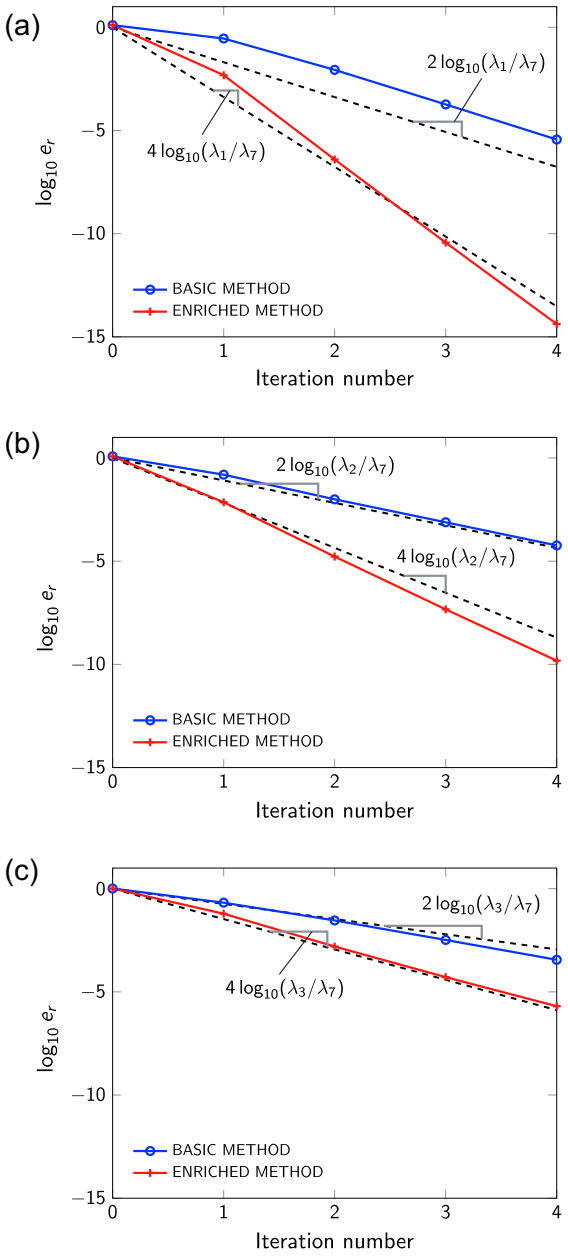

Fig. 2. Relative errors measured in the solution of the simple example problem when using the basic method and the enriched method: (a) first, (b) second and (c) third eigenvalues; the dashed lines give the theoretical convergence rates.

5. Construct $\overline{\mathbf{X}}_{k+1}$ where the column vectors span E_{k+1}

$$\overline{\mathbf{X}}_{k+1} = [\boldsymbol{\Phi}_k, \overline{\mathbf{X}}^a_{k+1}, \overline{\mathbf{Y}}_{k+1}]. \tag{16}$$

6. Project the matrices $\mathbf{K}$ and $\mathbf{M}$ onto the subspace E_{k+1}

$$\mathbf{K}_{k+1} = \overline{\mathbf{X}}^T_{k+1}\mathbf{K}\overline{\mathbf{X}}_{k+1}, \tag{17}$$

$$\mathbf{M}_{k+1} = \overline{\mathbf{X}}^T_{k+1}\mathbf{M}\overline{\mathbf{X}}_{k+1}. \tag{18}$$

7. Solve for the eigensystem of the projected matrices

$$\mathbf{K}_{k+1}\mathbf{Q}_{k+1} = \mathbf{M}_{k+1}\mathbf{Q}_{k+1}\boldsymbol{\Lambda}_{k+1}. \tag{19}$$

8. Calculate an improved approximation to the eigenvectors

$$\mathbf{X}_{k+1} = \overline{\mathbf{X}}_{k+1}\mathbf{Q}_{k+1}. \tag{20}$$

9. Measure which of the eigenvalue approximations $\lambda_i^{(k+1)}$ have converged [4], that is, satisfy

$$\left[1 - \frac{(\lambda_i^{(k+1)})^2}{(\mathbf{q}_i^{rr(k+1)})^T\mathbf{q}_i^{rr(k+1)}}\right]^{1/2} \leqslant tolc; \quad i = p_k + 1, \ldots, p \tag{21}$$

where $\mathbf{q}_i^{rr(k+1)}$ is the $(i-p_k)$th column vector in $\mathbf{Q}^{rr}_{k+1}$ with $\mathbf{Q}^{rr}_{k+1}$ being a $2r_k \times 2r_k$ submatrix obtained by partitioning $\mathbf{Q}_{k+1}$ into

$$\mathbf{Q}_{k+1} = \begin{bmatrix} \mathbf{Q}^{cc}_{k+1} & \mathbf{Q}^{cr}_{k+1} \\ \mathbf{Q}^{rc}_{k+1} & \mathbf{Q}^{rr}_{k+1} \end{bmatrix} \tag{22}$$

where the submatrices $\mathbf{Q}^{cc}_{k+1}, \mathbf{Q}^{cr}_{k+1}$ and $\mathbf{Q}^{rc}_{k+1}$ are of order $p_k \times p_k$, $p_k \times 2r_k$ and $2r_k \times p_k$, respectively.

10. Update the number of converged iteration vectors to p_{k+1} and increase k if $p_{k+1} < p$.

– End of 10 steps in each iteration

In the above iteration, it is effective to order the iteration vectors such that the corresponding approximate eigenvalues increase in magnitude. Then, provided that the starting iteration vectors in $\mathbf{X}_1$ are not $\mathbf{M}$-orthogonal to one of the eigenvectors sought we have, for $i = 1, \ldots, p$,

$$\lambda_i^{(k+1)} \to \lambda_i \quad \text{and} \quad \mathbf{x}_i^{(k+1)} \to \boldsymbol{\varphi}_i \quad \text{as} \quad k \to \infty \tag{23}$$

where $\mathbf{x}_i^{(k+1)}$ is the ith column vector in $\mathbf{X}_{k+1}$, see Section 3.

Note that in step 3(a) we consider the vectors $\widehat{\mathbf{x}}_i$, $i = r_k, r_k - 1, \ldots, 2, 1$, which are $\mathbf{M}$-orthogonal to all previous iteration vectors in $\mathbf{X}_k$ and choose only the corresponding vectors $\overline{\mathbf{x}}_i^{a(k+1)}$ that are providing a stable solution. With the ordering of the iteration vectors given above, the amount of turning is negligible for the first few iteration vectors, and near convergence very small. Hence we consider the vectors $\overline{\mathbf{x}}_i^{a(k+1)}$ in reverse order in Eq. (8). In step 3(b) then, the vectors $\widetilde{\mathbf{x}}_i$, $i = 1, \ldots, t_k$, are $\mathbf{M}$-orthogonalized to the vectors in $\boldsymbol{\Phi}_k$ and $\mathbf{X}^a_k$ and the first $r_k - t_k$ column vectors in $\mathbf{X}^b_k$.

In the actual implementation, we do not evaluate Eq. (8), and instead proceed to calculate the α_i values using the efficient algorithm given Appendix A. Also, as shown in the appendix some results obtained in step 3 can be reused in Eqs. (17) and (18).

The turning vectors in $\mathbf{Y}_k$ are $\mathbf{M}$-orthonormal and improved by solving Eq. (15). The entries in $\overline{\mathbf{Y}}_{k+1}$ corresponding to the turning vectors are the 'forward turning vectors'. For example, consider a subspace E_k spanned by $\{\mathbf{x}_1^{(k)}, \mathbf{x}_2^{(k)}\}$. We also assume that the turning vector of $\mathbf{x}_1^{(k)}$ is used, so that $\mathbf{y}_1^{(k)} = \widetilde{\mathbf{x}}_1 \Big/ \sqrt{\widetilde{\mathbf{x}}_1^T\mathbf{M}\widetilde{\mathbf{x}}_1}$. Since $\mathbf{y}_1^{(k)}$ can be expressed as a linear combination of $\mathbf{x}_1^{(k)}$ and $\overline{\mathbf{x}}_1^{(k+1)}$, $\overline{\mathbf{y}}_1^{(k+1)}$ is equal to a linear combination of $\overline{\mathbf{x}}_1^{(k+1)}$ and $\overline{\overline{\mathbf{x}}}_1^{(k+1)}$ where $\overline{\overline{\mathbf{x}}}_1^{(k+1)} = \mathbf{K}^{-1}(\mathbf{M}\overline{\mathbf{x}}_1^{(k+1)})$, and thus $\overline{\mathbf{y}}_1^{(k+1)}$ can be interpreted as a forward turning vector. Fig. 1 shows geometrically the subspace E_{k+1} used in the basic subspace iteration method and that used in the enriched subspace iteration method, when $\mathbf{M} = \mathbf{I}$.

Table 1 summarizes the complete procedure of the enriched subspace iteration method and gives the number of operations required with a comparison to the basic subspace iteration method. Here, one operation means one multiplication which almost always is followed by one addition.

An important aspect of enriching the subspace by forward turning vectors is that for the vector for which the turning is used, a single iteration in the enriched subspace iteration method has

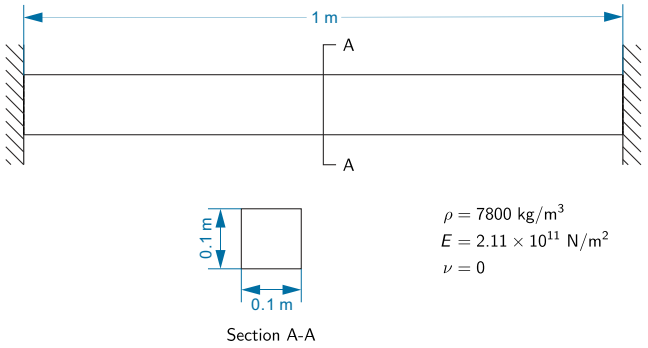

Fig. 3. Clamped-clamped beam problem.

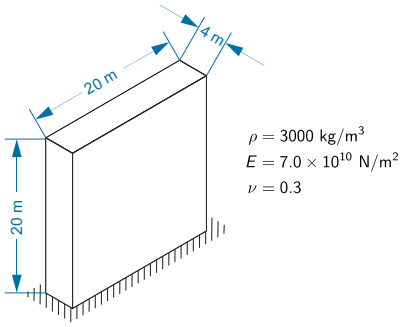

Fig. 4. Wall problem.

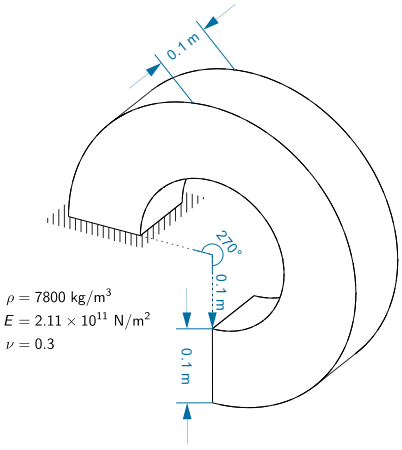

Fig. 5. Ring problem.

the effect of two iterations in the basic subspace iteration method. Furthermore, the computational effort related to the enrichment is not expensive. In a single iteration the number of operations (neglecting the $o(q^3)$ operations) of the enriched method is $2nr_k(4m+q+3)+nq^2+nt_k(q+r_k+1)$ and the basic method is $nq(4m+2q+3)$ where m is the half-bandwidth of $\mathbf{K}$ and $\mathbf{M}$, see Table 1. Of course, when $r_k = q/2$ and $t_k = 0$, the computational effort in both methods is the same. As an example when the enrichment is used, consider in three-dimensional analysis, $m = q$ and $r_k = t_k = q/2$ (although in practice r_k actually decreases as the iterations are performed). The ratio of the number of operations per iteration is then about 1.125, giving a 12.5% increase in the computational effort per iteration using the enrichment scheme. In many practical large scale analyses the increase in the cost is probably smaller because the half-bandwidth of the system is large, and indeed the increase may be practically negligible. Hence, considering not only the acceleration of the iteration by the use of turning vectors but also the inexpensive computational cost, the enrichment leads to a significant speed-up in solving the eigenvalue problem.

It is also important to note that the use of the turning vectors preserves the important characteristic of the basic subspace iteration method that the method is amenable to parallel processing, as discussed in Ref. [5]. Indeed, the computations in steps 2, 4, 6 and 8, in which most of the computational effort in the iterations is expended, can be parallelized as presented in Ref. [5]. The operations in step 3 can be also partly parallelized. Only some operations in the last summation of Eq. (13) are not amenable to parallel processing because in the loop the vector $\mathbf{v}_i$ is used with the vectors $\mathbf{y}_j^{(k)}, j = r_k - t_k + 1, \ldots, r_k - t_k + i - 1$, which were previously calculated in the same loop. However, the computational effort for this sequential algorithm is relatively small compared to the other computations which can be performed in parallel processing.

Several different ways could be developed to exploit the idea of using turning vectors, like dividing the iteration vectors in step 1 such that the number of vectors in $\mathbf{X}_k^a$ is equal to $p - p_k$. However, we focus in this paper on the simple dividing scheme described above.

2.2. Additional aspects

In the enriched subspace iteration method, the reduction in solution time is achieved by, firstly, the enrichment using forward turning vectors, which is most significant, and, secondly, by not performing the iteration for the converged iteration vectors. The second aspect is not important when only few eigenpairs are required, as it was the case in the early use of the method.

Considering that in the method the multiplication of the consistent mass matrix $\mathbf{M}$ by the iteration vectors might be computationally expensive and that the mode shapes are in many cases not very different from those of the same discretization using a lumped mass matrix, a further reduction in computations may be achieved by performing first a few iterations using a diagonal mass matrix. We construct this diagonal mass matrix using the diagonal elements of the consistent mass matrix $\mathbf{M}$ and scaling these elements so that all mass of the finite elements is applied. We call the eigenproblem using this lumped mass matrix the "associated eigenvalue problem" to the generalized eigenvalue problem in Eq. (1).

The solution procedure is then as follows:

(i) Establish the q starting iteration vectors, $q > p$, as discussed in Section 2.1.
(ii) Perform the enriched subspace iterations to solve the "associated eigenvalue problem" until a stopping criterion is satisfied (this step might not be included, see example solutions).
(iii) Using the solutions obtained from step (ii) continue the enriched subspace iterations to solve the generalized eigenvalue problem (Eq. (1)).
(iv) Carry out the Sturm sequence check to verify that the required eigenvalues and eigenvectors have indeed been calculated [1,4].

Simple options are possible for the stopping criterion in step (ii): for example, we may stop when the error bounds for all the eigenvalues sought are below the tolerance *tols* or we may stop

Table 2
Finite element meshes used to solve the beam problem.

Name	Number of elements (cross section × length)	Degrees of freedom	Half-bandwidth
B-MESH1	8 × 8 × 2200	534,357	243
B-MESH2	10 × 10 × 2800	1,016,037	363
B-MESH3	12 × 12 × 3000	1,520,493	507

Table 3
Finite element meshes used to solve the wall problem.

Name	Number of elements (width × length × height)	Degrees of freedom	Half-bandwidth
W-MESH1	12 × 52 × 52	107,484	2067
W-MESH2	12 × 72 × 72	204,984	2847
W-MESH3	12 × 86 × 86	291,798	3393

Table 4
Finite element meshes used to solve the ring problem.

Name	Number of elements (radius × axis × circumference)	Degrees of freedom	Half-bandwidth
R-MESH1	14 × 14 × 320	216,000	675
R-MESH2	18 × 18 × 360	389,880	1083
R-MESH3	22 × 22 × 380	603,060	1587

Table 5
CPU time for calculating the smallest $p = 100$ eigenvalues of the beam problem when using the associated eigenvalue problem; mesh used is B-MESH1 ($n = 534,357$ and $m = 243$); if the associated eigenvalue problem is not used, 15 iterations are performed and 1540 s are used.

tols	Number of iterations performed or number of iterations allowed in the associated eigenvalue problem	Total number of iterations performed	Total CPU time (s)
10^{-1}	9	17	1583
10^{-2}	10	18	1639
10^{-3}	11	18	1630
10^{-4}	12	19	1687
10^{-5}	13	20	1735
10^{-6}	15	22	1826
	3	15	1542
	4	15	1480
	5	15	1471
	6	16	1539

when the number of iterations reaches a prescribed number of iterations. We experimented with both options, see Section 4.

3. A simplified convergence analysis

In the previous section we discussed the new scheme and merely stated that if the subspace iteration converges, the required eigenpairs have been obtained. Here we examine in a simple way the iteration properties to obtain insight into how the forward turning vectors accelerate the convergence of the iterations.

The first (conceptual) step is to consider the iterations in the basis of eigenvectors [4,10,11]

$$\mathbf{X}_k = \mathbf{\Phi}\mathbf{Z}_k \tag{24}$$

where $\mathbf{\Phi} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_n]$ and since $\mathbf{X}_k^{\mathrm{T}}\mathbf{M}\mathbf{X}_k = \mathbf{I}$, it follows that $\mathbf{Z}_k^{\mathrm{T}}\mathbf{Z}_k = \mathbf{I}$. Introducing this change of basis into Eq. (7) leads to

$$\mathbf{\Lambda}\overline{\mathbf{Z}}_{k+1}^a = \mathbf{Z}_k^a \tag{25}$$

where $\mathbf{Z}_k^a$ of order $n \times r_k$ corresponds to the first r_k vectors in $\mathbf{Z}_k$. We assume here, for simplicity, that $q = 2p$, $p_k = 0$, hence $r_k = p$, and that all the turning vectors are employed for use of corresponding

Table 6
CPU time for calculating the smallest $p = 100$ eigenvalues of the wall problem when using the associated eigenvalue problem; mesh used is W-MESH1 ($n = 107,484$ and $m = 2067$); if the associated eigenvalue problem is not used, 12 iterations are performed and 1282 s are used.

tols	Number of iterations performed or number of iterations allowed in the associated eigenvalue problem	Total number of iterations performed	Total CPU time (s)
10^{-1}	4	12	1140
10^{-2}	5	13	1219
10^{-3}	6	14	1272
10^{-4}	7	15	1318
10^{-5}	10	18	1452
10^{-6}	13	21	1531
	3	12	1156
	4	12	1140
	5	13	1219
	6	14	1272

Table 7
CPU time for calculating the smallest $p = 100$ eigenvalues of the ring problem when using the associated eigenvalue problem; mesh used is R-MESH1 ($n = 216,000$ and $m = 675$); if the associated eigenvalue problem is not used, 11 iterations are performed and 977 s are used.

tols	Number of iterations performed or number of iterations allowed in the associated eigenvalue problem	Total number of iterations performed	Total CPU time (s)
10^{-1}	4	12	944
10^{-2}	6	13	992
10^{-3}	7	14	1038
10^{-4}	8	15	1080
10^{-5}	9	16	1056
10^{-6}	12	19	1213
	3	12	956
	4	12	944
	5	12	949
	6	13	1012

Table 8
Speed-up of the enriched method when compared to the basic method for the beam problem.

Numerical model	p	Speed-up
B-MESH1 ($n = 534,357$ and $m = 243$)	50	4.95
	100	3.62
	150	2.93
B-MESH2 ($n = 1,016,037$ and $m = 363$)	50	5.41
	100	3.23
	150	2.75
B-MESH3 ($n = 1,520,493$ and $m = 507$)	50	4.54
	100	3.18
	150	2.73

forward turning vectors. The subspace E_{k+1} is then the subspace spanned by $\{\overline{\mathbf{Z}}_{k+1}^a, \overline{\overline{\mathbf{Z}}}_{k+1}^a\}$ with $\overline{\overline{\mathbf{Z}}}_{k+1}^a = \mathbf{\Lambda}^{-1}\overline{\mathbf{Z}}_{k+1}^a$.

For the analysis, let us consider a matrix $\mathbf{\Xi}_k$ defined by [4,11]

$$\mathbf{\Xi}_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \\ \xi_{q+1,1}^{(k)} & \xi_{q+1,2}^{(k)} & \cdots & \xi_{q+1,q}^{(k)} \\ \xi_{q+2,1}^{(k)} & \xi_{q+2,2}^{(k)} & \cdots & \xi_{q+2,q}^{(k)} \\ \vdots & \vdots & & \vdots \\ \xi_{n,1}^{(k)} & \xi_{n,2}^{(k)} & \cdots & \xi_{n,q}^{(k)} \end{bmatrix}. \tag{26}$$

Table 9
CPU time used in each iteration for calculating the smallest $p = 100$ eigenvalues of the beam problem when using the basic method and the enriched method; mesh used is B-MESH3 ($n = 1,520,493$ and $m = 507$); 0 denotes establishing the starting iteration vectors and performing a single iteration in the basic method, as described in Section 2.1, for the associated eigenvalue problem; 00 denotes performing the **M**-orthonormalization of the iteration vectors.

Basic subspace iteration method		Enriched subspace iteration method			
Total number of iterations performed	Average CPU time (s) in each iteration/ Total CPU time (s)	Iteration number	Number of turning vectors used in iteration	Cumulative number of converged vectors after iteration	CPU time (s) used in iteration/ Rounded cumulative CPU time (s)
44	583/ 25,664	0			367/367
		1	100		439/805
		2	84		429/1234
		3	68		412/1646
		4	49		388/2034
		5	36		367/2401
		00			308/2709
		6	81	6	640/3349
		7	47	26	598/3947
		8	31	35	530/4477
		9	28	42	502/4979
		10	21	50	477/5456
		11	16	57	451/5907
		12	12	66	430/6338
		13	7	77	400/6738
		14	3	87	366/7104
		15	1	91	337/7442
		16	0	99	326/7768
		17	1	109	305/8072

Assuming that the iteration vectors in $\mathbf{Z}_k$ are not deficient in the vectors $\widehat{\mathbf{e}}_i$, $i = 1, \ldots, q$, which are the eigenvectors corresponding to the lowest q eigenvalues of $\mathbf{\Lambda}$, the vectors $\mathbf{z}_i^{a(k)}$, $i = 1, \ldots, p$, in $\mathbf{Z}_k^a$ can be expressed as

$$\mathbf{z}_i^{a(k)} = \gamma_i^{(k)} \boldsymbol{\xi}_i^{(k)} + \delta_i^{(k)} \mathbf{r}_i^{(k)} \tag{27}$$

with the ith column vector $\boldsymbol{\xi}_i^{(k)}$ in $\mathbf{\Xi}_k$, some coefficients $\gamma_i^{(k)}$ and $\delta_i^{(k)}$, and a residual vector $\mathbf{r}_i^{(k)}$. We also note that since the vectors in $\mathbf{Z}_k^a$ in the basic subspace iteration with p iteration vectors converge to the smallest p eigenvectors sought [11], we have that, for $i = 1, \ldots, p$,

$$\boldsymbol{\xi}_i^{(k)} \to \widehat{\mathbf{e}}_i, \quad \gamma_i^{(k)} \to 1 \quad \text{and} \quad \delta_i^{(k)} \to 0 \quad \text{as} \quad k \to \infty. \tag{28}$$

Let us now assume that $\delta_i^{(k)}$ is small, equal to $\varepsilon_i^{(k)} \ll 1$. The corresponding vectors $\overline{\overline{\mathbf{z}}}_i^{a(k+1)}$, $i = 1, \ldots, p$, in $\overline{\overline{\mathbf{Z}}}_{k+1}^a$ are then

$$\overline{\overline{\mathbf{z}}}_i^{a(k+1)} = \gamma_i^{(k)} \mathbf{\Lambda}^{-2} \boldsymbol{\xi}_i^{(k)} + \varepsilon_i^{(k)} \mathbf{\Lambda}^{-2} \mathbf{r}_i^{(k)} \tag{29}$$

and we have

$$\begin{aligned}
\frac{\|\lambda_i^2 \overline{\overline{\mathbf{z}}}_i^{a(k+1)} - \widehat{\mathbf{e}}_i\|_2}{\|\mathbf{z}_i^{a(k)} - \widehat{\mathbf{e}}_i\|_2} &= \frac{\|\gamma_i^{(k)} \lambda_i^2 \mathbf{\Lambda}^{-2} \boldsymbol{\xi}_i^{(k)} + o(\varepsilon_i^{(k)}) \mathbf{r}_i^{(k)} - \widehat{\mathbf{e}}_i\|_2}{\|\gamma_i^{(k)} \boldsymbol{\xi}_i^{(k)} + o(\varepsilon_i^{(k)}) \mathbf{r}_i^{(k)} - \widehat{\mathbf{e}}_i\|_2} \\
&= \frac{\|\lambda_i^2 \mathbf{\Lambda}^{-2} \boldsymbol{\xi}_i^{(k)} - \widehat{\mathbf{e}}_i\|_2}{\|\boldsymbol{\xi}_i^{(k)} - \widehat{\mathbf{e}}_i\|_2} + o(\varepsilon_i^{(k)}) \\
&= \sqrt{\frac{\sum_{j=q+1}^{n} \left(\frac{\lambda_i}{\lambda_j}\right)^4 \left(\xi_{j,i}^{(k)}\right)^2}{\sum_{j=q+1}^{n} \left(\xi_{j,i}^{(k)}\right)^2}} + o(\varepsilon_i^{(k)}) \\
&\leqslant \left(\frac{\lambda_i}{\lambda_{q+1}}\right)^2 + o(\varepsilon_i^{(k)})
\end{aligned} \tag{30}$$

where we used that $\gamma_i^{(k)} = 1 + o(\varepsilon_i^{(k)})$ with $o(\varepsilon_i^{(k)})$ denoting "of order $\varepsilon_i^{(k)}$".

Since $\mathbf{z}_i^{(k+1)}$ is the best approximation to the eigenvector $\widehat{\mathbf{e}}_i$ in the subspace E_{k+1}, we finally obtain

$$\frac{\|\mathbf{z}_i^{(k+1)} - \widehat{\mathbf{e}}_i\|_2}{\|\mathbf{z}_i^{(k)} - \widehat{\mathbf{e}}_i\|_2} \leqslant \left(\frac{\lambda_i}{\lambda_{q+1}}\right)^2 + o(\varepsilon_i^{(k)}); \quad i = 1, \ldots, p. \tag{31}$$

We can therefore conclude that using $q = 2p$ and $p_k = 0$, provided the iteration vectors in $\mathbf{X}_k$ are ordered appropriately, are not deficient in the eigenvectors, and in each iteration all turning vectors are used, the iteration vectors $\mathbf{x}_i^{(k)}$, $i = 1, \ldots, p$, converge with the rates $(\lambda_i/\lambda_{q+1})^2$. The rates of convergence of the corresponding eigenvalues are $(\lambda_i/\lambda_{q+1})^4$, $i = 1, \ldots, p$, because these are calculated from the Rayleigh quotient.

We illustrate the convergence of the new scheme in a simple example using the diagonal matrix $\mathbf{K} = \text{diag}(1, 2, \ldots, 12)$ and $\mathbf{M} = \mathbf{I}$, and seeking the smallest $p = 3$ eigenvalues. For the solution, we set $q = 2p$ and $p_k = 0$, and all turning vectors are used in the iterations. We also compare the results with those from the basic subspace iteration method with the number of iteration vectors $q = 2p$. In both cases, six random vectors are used as the starting vectors.

Fig. 2 shows the relative errors of the calculated eigenvalue approximations, denoted as $e_r = (\lambda_i^{(k)} - \lambda_i)/\lambda_i$ when using the subspace iteration method with and without the enrichment. In the figure, the dashed lines are the theoretical rates of convergence. We see that the computed rates (slopes in Fig. 2) follow quite closely the theoretically predicted rates and the rate of convergence from the solution using the enriched subspace iteration method is about two times the rate of convergence of the basic subspace iteration method.

4. Illustrative example solutions

The objective in this section is to demonstrate the performance of the enrichment scheme through some example solutions. The observed computational cost of the enriched method is compared to the observed cost of the basic method, using each time $q = 2p$ iteration vectors. In both cases, the computational expense compared is the CPU time required to calculate all p eigenvalues with $tolc = 10^{-6}$ for convergence, with the time used for the factorization of the stiffness matrix **K** and the Sturm sequence check not included. In the enriched subspace iterations we use $tolt = 10^{-8}$, see Section 2.1.

The solutions are obtained using a laptop with a single core Intel 2.40 GHz CPU and 24 GB RAM. In each case, we use the simple scheme to establish the starting iteration vectors described in Section 2.1.

We consider three problems: a clamped-clamped beam structure with mass density $\rho = 7800\,\text{kg/m}^3$, Young's modulus $E = 2.11 \times 10^{11}\,\text{N/m}^2$ and Poisson's ratio $\nu = 0$, a wall structure with $\rho = 3000\,\text{kg/m}^3$, $E = 7.0 \times 10^{10}\,\text{N/m}^2$ and $\nu = 0.3$ and a ring structure with $\rho = 7800\,\text{kg/m}^3$, $E = 2.11 \times 10^{11}\,\text{N/m}^2$ and $\nu = 0.3$, as shown in Figs. 3–5, respectively. The beam problem is taken from Ref. [5]. The structures are modeled using three-dimensional 8-node brick elements, and in all cases using the consistent mass matrix. We calculate the solutions with three different mesh sizes for each problem, see Tables 2–4, to investigate whether the solution time increases linearly with the number of the required eigenpairs. Here, the half-bandwidth means the mean half-band width of the stiffness matrix **K** after factorization [4].

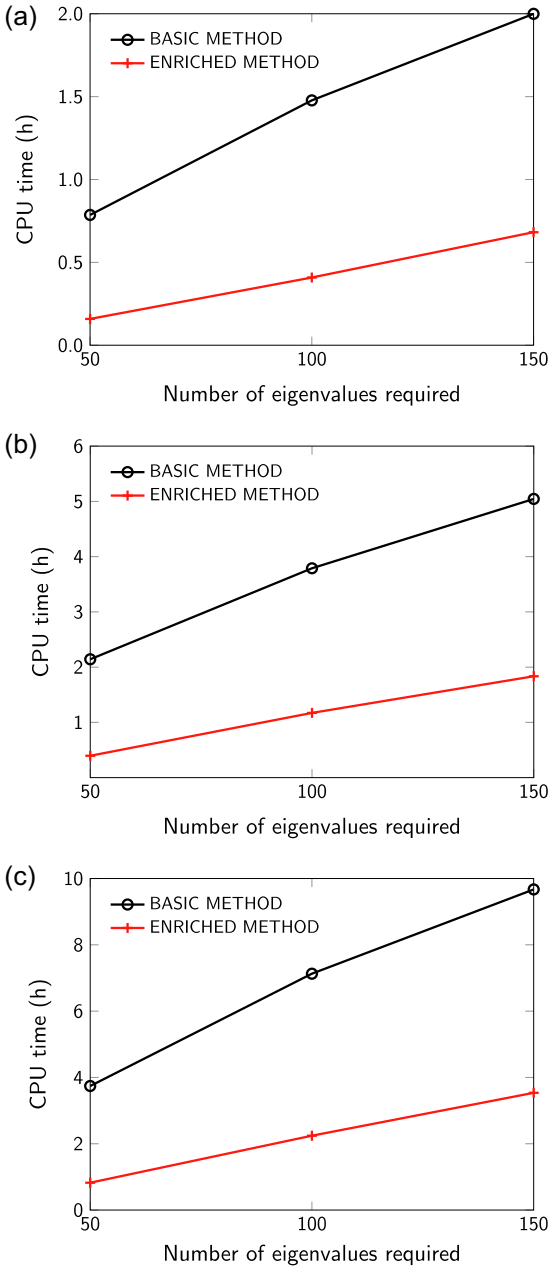

Fig. 6. CPU time for calculating the smallest p eigenvalues of the beam problem when using the basic method and the enriched method; meshes used are (a) B-MESH1 ($n = 534,357$ and $m = 243$), (b) B-MESH2 ($n = 1,016,037$ and $m = 363$) and (c) B-MESH3 ($n = 1,520,493$ and $m = 507$).

Table 10
Speed-up of the enriched method when compared to the basic method for the wall problem.

Numerical model	p	Speed-up
W-MESH1 ($n = 107,484$ and $m = 2067$)	50	3.25
	100	3.82
	150	3.69
W-MESH2 ($n = 204,984$ and $m = 2847$)	50	3.15
	100	3.45
	150	3.53
W-MESH3 ($n = 291,798$ and $m = 3393$)	50	3.23
	100	3.59
	150	3.50

Table 11
CPU time used in each iteration for calculating the smallest $p = 100$ eigenvalues of the wall problem when using the basic method and the enriched method; mesh used is W-MESH3 ($n = 291,798$ and $m = 3393$); 0 denotes establishing the starting iteration vectors and performing a single iteration in the basic method, as described in Section 2.1, for the associated eigenvalue problem; 00 denotes performing the **M**-orthonormalization of the iteration vectors.

Basic subspace iteration method		Enriched subspace iteration method			
Total number of iterations performed	Average CPU time (s) in each iteration/ Total CPU time (s)	Iteration number	Number of turning vectors used in iteration	Cumulative number of converged vectors after iteration	CPU time (s) used in iteration/ Rounded cumulative CPU time (s)
35	593/ 20,757	0			294/294
		1	100		309/603
		2	81		308/912
		3	65		298/1210
		4	56		288/1497
		5	44		262/1760
		00			304/2065
		6	78	3	601/2666
		7	22	18	590/3256
		8	14	40	542/3798
		9	16	55	481/4279
		10	16	66	432/4712
		11	13	83	404/5116
		12	19	97	350/5466
		13	22	102	313/5779

To establish how many iterations might best be performed using the associated eigenvalue problem (using the associated lumped mass matrix) before switching to the solution using the consistent mass matrix, we first solve the beam, the wall and the ring problems using the finite element meshes B-MESH1, W-MESH1 and R-MESH1, respectively. Tables 5–7 show the results obtained when $p = 100$. When solving the associated eigenvalue problem, we applied the two different stopping criteria mentioned in Section 2.2.

As shown in Table 5, for the beam problem, the most efficient solution is reached when the maximum number of iterations allowed in the associated eigenvalue problem is equal to 5. For the wall and ring problems, however, it is most efficient to use the tolerance stopping criterion with $tols = 10^{-1}$, see Tables 6 and 7. From these numerical experiments, we can conclude that it is probably reasonable to stop solving the associated eigenvalue problem when the error bounds for all required eigenvalue approximations are below $tols = 10^{-1}$ or a maximum of 5 iterations has been reached. This is also reasonable because then about one-third of the total number of iterations might be performed using the associated eigenvalue problem. We use this stopping criterion in the following example solutions. However, we also note that in each case the solution time saved by considering in the first iterations the associated eigenvalue problem is not large. Hence in practice, we may actually not first consider this problem but directly solve the generalized eigenvalue problem in all iterations.

4.1. Solution of beam problem

Table 8 shows the speed-up obtained using the enriched subspace iteration method, defined as the CPU time used in the basic method divided by the CPU time used in the enriched method. The enriched method is about 3–5 times faster than the basic method, depending on the number of degrees of freedom used and the number of eigenpairs sought. For the case B-MESH3

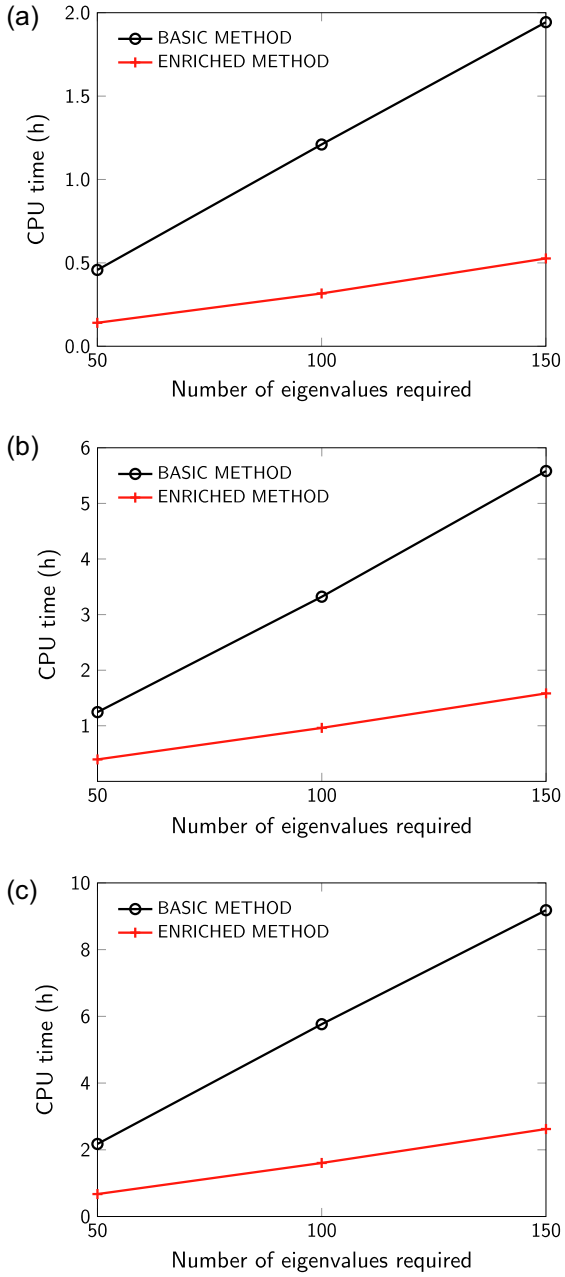

Fig. 7. CPU time for calculating the smallest p eigenvalues of the wall problem when using the basic method and the enriched method; meshes used are (a) W-MESH1 ($n = 107,484$ and $m = 2067$), (b) W-MESH2 ($n = 204,984$ and $m = 2847$) and (c) W-MESH3 ($n = 291,798$ and $m = 3393$).

Table 12
Speed-up of the enriched method when compared to the basic method for the ring problem.

Numerical model	p	Speed-up
R-MESH1 ($n = 216,000$ and $m = 675$)	50	3.37
	100	3.26
	150	3.44
R-MESH2 ($n = 389,880$ and $m = 1083$)	50	3.41
	100	3.20
	150	3.44
R-MESH3 ($n = 603,060$ and $m = 1587$)	50	3.17
	100	3.01
	150	3.07

Table 13
CPU time used in each iteration for calculating the smallest $p = 100$ eigenvalues of the ring problem when using the basic method and the enriched method; mesh used is R-MESH3 ($n = 603,060$ and $m = 1587$); 0 denotes establishing the starting iteration vectors and performing a single iteration in the basic method, as described in Section 2.1, for the associated eigenvalue problem; 00 denotes performing the **M**-orthonormalization of the iteration vectors.

Basic subspace iteration method		Enriched subspace iteration method			
Total number of iterations performed	Average CPU time (s) in each iteration/ Total CPU time (s)	Iteration number	Number of turning vectors used in iteration	Cumulative number of converged vectors after iteration	CPU time (s) used in iteration/ Rounded cumulative CPU time (s)
29	596/ 17,278	0			323/323
		1	100		342/665
		2	80		338/1002
		3	74		326/1329
		4	62		310/1639
		00			305/1945
		5	79	5	618/2563
		6	33	19	595/3158
		7	16	36	547/3705
		8	14	48	500/4205
		9	13	67	461/4666
		10	13	86	407/5074
		11	21	99	348/5422
		12	20	103	309/5731

with $p = 100$, Table 9 gives the computational cost per iteration with the number of forward turning vectors actually used and the convergence history. We note that the average computational time used per iteration is higher in the basic method because the converged vectors are still included in the iterations.

Fig. 6 gives the CPU times used in the basic method and the enriched method. We see that the CPU time increases almost linearly with the number of frequencies and mode shapes required and that the slopes of the curves are smaller for the enriched method.

4.2. Solution of wall problem

In this problem solution the finite element matrices have a large half-bandwidth. The speed-up is also significant, an improvement of a factor larger than 3 is obtained, see Table 10. For the case W-MESH3 with $p = 100$, Table 11 gives the computational cost per iteration with the number of forward turning vectors actually used and the convergence history. We again note that the average computational time used per iteration is higher in the basic method.

Fig. 7 shows the CPU times used in the basic method and the enriched method. As in the previous example solutions, we observe that regardless of n, the computational cost increases almost in linear proportion to the required eigenpairs, and that the slopes of the curves are smaller for the enriched method.

4.3. Solution of ring problem

Table 12 shows the speed-up achieved by the enriched subspace iteration method, and we see that the speed-up factor is larger than 3. For the case R-MESH3 with $p = 100$, Table 13 gives the CPU time required per iteration with the number of forward turning vectors actually used and the convergence history. We again observe that in the enriched method, the computational cost per iteration gradually decreases and the total number of iterations performed is smaller.

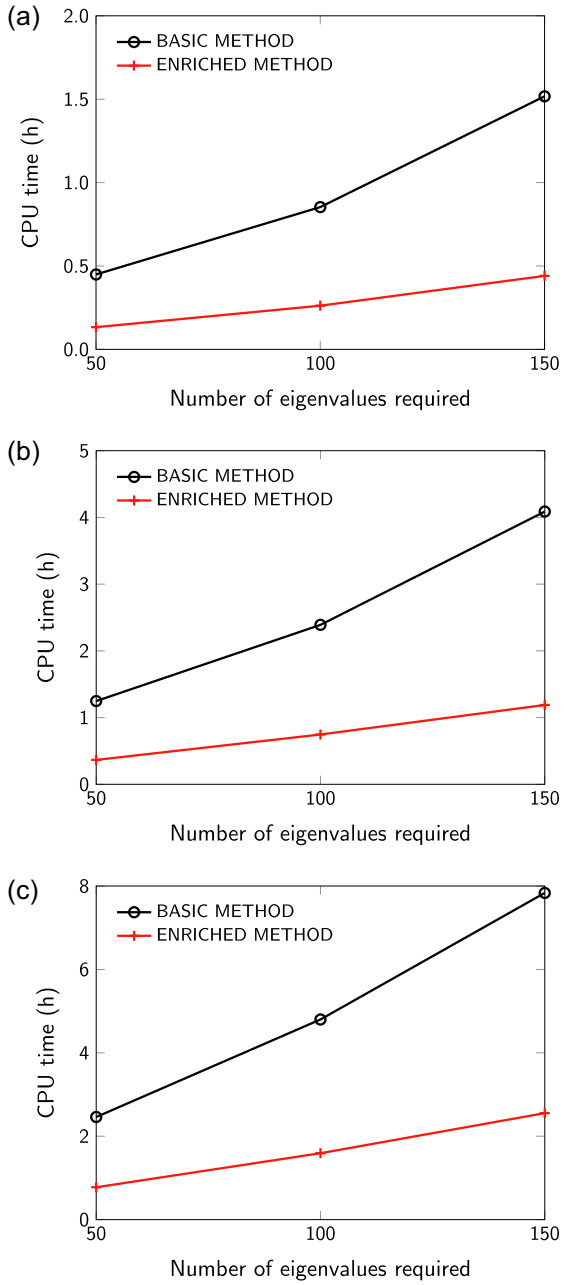

Fig. 8. CPU time for calculating the smallest p eigenvalues of the ring problem when using the basic method and the enriched method; meshes used are (a) R-MESH1 ($n = 216,000$ and $m = 675$), (b) R-MESH2 ($n = 389,880$ and $m = 1083$) and (c) R-MESH3 ($n = 603,060$ and $m = 1587$).

Here, we also see approximately a linear increase in the CPU time and the slopes of the curves are smaller in the enriched method, see Fig. 8.

5. Concluding remarks

The objective in this paper was to present a novel effective scheme to accelerate the Bathe subspace iteration method, especially for obtaining many eigenpairs of very large eigenvalue problems. The basic subspace iteration method was developed when only a relatively small number of frequencies and mode shapes were used for dynamic analyses, however, the demands in analyses have increased very much. In today's finite element practice, one hundred or more frequencies and mode shapes may be used.

We showed that the use of forward turning vectors in the subspace iterations is inexpensive and significantly speeds up the convergence of the iteration vectors. We measured speed-ups of 3–5 in some example solutions using a single core laptop machine. This speed-up is clearly significant, but the idea of using the "turning of iteration vectors" in the Bathe subspace iteration method might be further explored to possibly reach an even greater efficiency.

While we focused in the paper on the case of using the consistent mass matrix (because of its wide use), the improvements in the subspace iteration method can also directly be employed in the case of a lumped mass matrix, and we expect then too a significant speed-up.

We did not discuss and show solutions using parallel processing and instead focused on the basic improvements reached without parallel processing. However, the equations used show that the computations in the enriched subspace iteration method can be directly parallelized as in the original basic subspace iteration method [5]. With the parallel processing the enriched subspace iteration method presented in the paper will likely provide a very effective solution scheme.

Appendix A. Calculation of the amount of vector turning and the turning vectors

In this Appendix A we provide an efficient algorithm to estimate the amount of turning and to calculate the turning vectors.

In iteration k, after evaluating $\overline{\mathbf{X}}^a_{k+1}$ first calculate matrices $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$, $\mathbf{C}_{k+1}$, which are all of order $r_k \times r_k$, and $\mathbf{D}_{k+1}$ of order $p_k \times r_k$

$$\mathbf{A}_{k+1} = (\mathbf{X}^a_k)^{\mathrm{T}}\mathbf{M}\overline{\mathbf{X}}^a_{k+1}, \tag{32}$$

$$\mathbf{B}_{k+1} = (\mathbf{X}^b_k)^{\mathrm{T}}\mathbf{M}\overline{\mathbf{X}}^a_{k+1}, \tag{33}$$

$$\mathbf{C}_{k+1} = (\overline{\mathbf{X}}^a_{k+1})^{\mathrm{T}}\mathbf{M}\overline{\mathbf{X}}^a_{k+1}, \tag{34}$$

$$\mathbf{D}_{k+1} = \mathbf{\Phi}^{\mathrm{T}}_k\mathbf{M}\overline{\mathbf{X}}^a_{k+1} \tag{35}$$

and then construct $\mathbf{Y}_k$ of order $n \times r_k$ by following the steps in part (a) and thereafter in part (b):

(a) Calculate in reverse order the amount of turning of the iteration vectors in $\overline{\mathbf{X}}^a_{k+1}$, and choose the vectors for which the measure used is larger than the tolerance *tolt*, i.e., for $i = r_k, r_k - 1, \ldots, 2, 1$, with initially $t_k = 0$, first let

$$l = t_k + 1, \tag{36}$$

$$h_l = i \tag{37}$$

and then calculate for $m = 1, \ldots, l$,

$$j = h_m, \tag{38}$$

$$r_{lm} = C^{(k+1)}_{ij} - (\mathbf{a}^{(k+1)}_i)^{\mathrm{T}}\mathbf{a}^{(k+1)}_j - (\mathbf{b}^{(k+1)}_i)^{\mathrm{T}}\mathbf{b}^{(k+1)}_j - (\mathbf{d}^{(k+1)}_i)^{\mathrm{T}}\mathbf{d}^{(k+1)}_j - \sum_{k=1}^{m-1}\frac{r_{lk}r_{mk}}{r_{kk}} \tag{39}$$

where $\mathbf{a}^{(k+1)}_i$, $\mathbf{b}^{(k+1)}_i$ and $\mathbf{d}^{(k+1)}_i$ are ith column vectors in $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ and $\mathbf{D}_{k+1}$, respectively. If $r_{ll}/C^{(k+1)}_{ii} \leqslant tolt$ go to the next value of i. If $r_{ll}/C^{(k+1)}_{ii} > tolt$ we proceed as follows

$$t_k = l, \tag{40}$$

$$\mathbf{v}_{t_k} = \overline{\mathbf{x}}^{a(k+1)}_i \tag{41}$$

and now go to the next value of i.

(b) Next, let t_k be the last value reached in the above loop in part (a). We now construct $\mathbf{Y}_k = \left[\mathbf{x}_1^{b(k)}, \ldots \mathbf{x}_{r_k - t_k}^{b(k)}, \mathbf{v}_1, \ldots, \mathbf{v}_{t_k}\right]$ and calculate for $i = 1, \ldots, t_k$,

$$l = h_i, \tag{42}$$

$$\widetilde{\mathbf{x}}_i = \mathbf{v}_i - \mathbf{X}_k^a \mathbf{a}_l^{(k+1)} - \mathbf{\Phi}_k \mathbf{d}_l^{(k+1)} - \sum_{j=1}^{r_k - t_k} \mathbf{y}_j^{(k)} b_{j,l}^{(k+1)} - \sum_{j=r_k - t_k + 1}^{r_k - t_k + i - 1} \mathbf{y}_j^{(k)} ((\mathbf{y}_j^{(k)})^{\mathrm{T}} \mathbf{M} \mathbf{v}_i), \tag{43}$$

$$\mathbf{y}_{r_k - t_k + i}^{(k)} = \frac{\widetilde{\mathbf{x}}_i}{\sqrt{\widetilde{\mathbf{x}}_i^{\mathrm{T}} \mathbf{M} \widetilde{\mathbf{x}}_i}}. \tag{44}$$

Note that the matrix $\mathbf{A}_{k+1}$ and some parts of the matrix $\mathbf{B}_{k+1}$ can be used in the calculation of $\mathbf{K}_{k+1}$, and the matrices $\mathbf{C}_{k+1}$ and $\mathbf{D}_{k+1}$ can be used in the construction of $\mathbf{M}_{k+1}$, i.e.,

$$\begin{aligned} \mathbf{K}_{k+1} &= \begin{bmatrix} \mathbf{\Phi}_k^{\mathrm{T}} \\ (\overline{\mathbf{X}}_{k+1}^a)^{\mathrm{T}} \\ \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}} \end{bmatrix} \mathbf{K} \begin{bmatrix} \mathbf{\Phi}_k & \overline{\mathbf{X}}_{k+1}^a & \overline{\mathbf{Y}}_{k+1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{\Lambda}_k & & \text{sym.} \\ \mathbf{0} & \mathbf{A}_{k+1} & \\ \mathbf{0} & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}} \mathbf{M} \mathbf{X}_k^a & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}} \mathbf{M} \mathbf{Y}_k \end{bmatrix} \end{aligned} \tag{45}$$

where the entries in the first $r_k - t_k$ rows of the two matrices $\mathbf{B}_{k+1}$ and $\mathbf{Y}_k^{\mathrm{T}} \mathbf{M} \overline{\mathbf{X}}_{k+1}^a$ are the same, and

$$\begin{aligned} \mathbf{M}_{k+1} &= \begin{bmatrix} \mathbf{\Phi}_k^{\mathrm{T}} \\ (\overline{\mathbf{X}}_{k+1}^a)^{\mathrm{T}} \\ \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}} \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{\Phi}_k & \overline{\mathbf{X}}_{k+1}^a & \overline{\mathbf{Y}}_{k+1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & & \text{sym.} \\ \mathbf{D}_{k+1}^{\mathrm{T}} & \mathbf{C}_{k+1} & \\ \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}} \mathbf{M} \mathbf{\Phi}_k & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}} \mathbf{M} \overline{\mathbf{X}}_{k+1}^a & \overline{\mathbf{Y}}_{k+1}^{\mathrm{T}} \mathbf{M} \overline{\mathbf{Y}}_{k+1} \end{bmatrix}. \end{aligned} \tag{46}$$

References

[1] Bathe KJ. Solution methods for large generalized eigenvalue problems in structural engineering, Report UCSESM 71–20. Berkeley: University of California; 1971.

[2] Bathe KJ, Wilson EL. Large eigenvalue problems in dynamic analysis. ASCE J Eng Mech Div 1972;98:1471–85.

[3] Bathe KJ, Wilson EL. Solution methods for eigenvalue problems in structural mechanics. Int J Numer Meth Eng 1973;6:213–26.

[4] Bathe KJ. Finite element procedures. 2nd ed. Watertown, MA: Klaus-Jürgen Bathe; 2014.

[5] Bathe KJ. The subspace iteration method – revisited. Comput Struct 2013;126:177–83.

[6] Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J Res Natl Bureau Stand 1950;45:255–82.

[7] Paige CC. Computational variants of the Lanczos method for the eigenproblem. IMA J Appl Math 1972;10:373–81.

[8] Ericsson T, Ruhe A. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. Math Comput 1980;35:1251–68.

[9] Calvetti D, Reichel L, Sorensen DC. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. Electr Trans Numer Anal 1994;2:1–21.

[10] Rutishauser H. Computational aspects of F. L. Bauer's simultaneous iteration method. Numer Math 1969;13:4–13.

[11] Bathe KJ. Convergence of subspace iteration. In: Bathe KJ, Oden JT, Wunderlich W, editors. Formulations and numerical algorithms in finite element analysis. Cambridge, MA: M.I.T. Press; 1977. p. 575–98.