Computers and Structures xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computers and Structures

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

The subspace iteration method – Revisited

Klaus-Jürgen Bathe

Massachusetts Institute of Technology, Cambridge, MA, United States

ARTICLE INFO

Article history:
Received 22 May 2012
Accepted 5 June 2012
Available online xxxx

Keywords:
Frequencies
Mode shapes
Subspace iteration method
Parallel processing
Shared memory
Distributed memory

ABSTRACT

The objective in this paper is to present some recent developments regarding the subspace iteration method for the solution of frequencies and mode shapes. The developments pertain to speeding up the basic subspace iteration method by choosing an effective number of iteration vectors and by the use of parallel processing. The subspace iteration method lends itself particularly well to shared and distributed memory processing. We present the algorithms used and illustrative sample solutions. The present paper may be regarded as an addendum to the publications presented in the early 1970s, see Refs. [1,2], taking into account the changes in computers that have taken place.

1. Introduction

The subspace iteration method was developed by K.J. Bathe [1] for the solution of frequencies and mode shapes of structures, and in particular for the earthquake analysis of buildings and bridges [1–3]. Originally in the 1970s, relatively few eigenpairs were sought in these solutions, like the lowest 10 to 20 frequencies and mode shapes, when the model contained a total of 1000 to 10,000 degrees of freedom. However, since its original development, the subspace iteration method has been used abundantly in research and commercial finite element programs for small and very large finite element systems, and the method has naturally attracted considerable attention for improvements, see for example Refs. [4–10].

The original development of the method given in Ref. [1] is based on vector simultaneous iterations, as proposed by Bauer [11] and Rutishauser [12], but includes the important use of the Ritz method, the selection of the iteration starting vectors, the use of an effective number of starting vectors, error measures, and the Sturm sequence check. Without the use of the Ritz step, simultaneous vector iterations are not effective. While abundantly used for frequency and linearized buckling solutions in engineering and the sciences, the method is also employed in the solution of random eigenvalue problems [13]. A convergence analysis of the subspace iteration method is given in Ref. [14].

Two attractive properties of the subspace iteration method are, firstly, its robustness and efficiency and, secondly, the fact that using a starting subspace close to the subspace of interest can lead to a very fast solution. This situation is frequently encountered in engineering and the sciences, e.g. in optimization problems and in protein dynamics. We shall focus in this paper on the selection of the number of iteration vectors and illustrate a third attractive property, namely, its use in parallelized computations.

Also, as another technique, the Lanczos method can be very effective, in particular when solving for many frequencies and mode shapes [15,16]. Initially, the Lanczos method showed instabilities due to loss of orthogonality of the iteration vectors. However, these difficulties have been largely overcome and in good implementations the method can be very efficient [17,18]. A particular asset of the method is that the computational effort may increase almost linearly with the number of eigenpairs sought. This asset can render the Lanczos method attractive compared to the original subspace iteration method if many eigenpairs need be calculated. Namely, in that case, the computational effort increases larger than linear in the *original* subspace iteration method, and this increase can be significant. The Lanczos method and Bathe's subspace iteration method (or variants of these two iterative schemes) are two techniques that, at present, are very widely used for the solution of large eigenvalue problems in finite element analysis. Any noteworthy improvements to these methods are therefore of interest.

An important step in the subspace iteration method is to establish effective starting iteration vectors, which also implies to, ideally, use the optimal number of iteration vectors.

Lately much effort has been spent on using parallel processing in finite element analysis, in shared memory and distributed memory processing modes. Whereas the Lanczos method (working on individual vectors [16]) can intrinsically only be parallelized in the

E-mail address: kjb@mit.edu

0045-7949/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compstruc.2012.06.002

Please cite this article in press as: Bathe K.J. The subspace iteration method – Revisited. Comput Struct (2012), http://dx.doi.org/10.1016/j.compstruc.2012.06.002

factorization of the stiffness matrix and the forward reduction and back-substitution of the *individual* vectors, the subspace iteration method allows in addition the parallel solution of *multiple* iteration vectors, which can result in a large computational benefit.

In this paper we first present the subspace iteration method implying use on a single processor machine and discuss how to choose an effective number of iteration vectors in structural analyses. While the optimal number must clearly depend on the problem considered, a good choice can mean a significant reduction in computational time when many frequencies and mode shapes shall be computed. Based on the discussion, we arrive at a simple formula for the selection of a reasonable number of iteration vectors for any solution.

Thereafter, we consider the use of the subspace iteration method in parallel processing, on shared memory and distributed memory machines. In a brief discussion, we show how the method lends itself particularly well to parallel computations.

Finally, we give the results of some illustrative example solutions.

2. The basic subspace iteration method

The basic equations of Bathe's subspace iteration method have been published in Refs. [1,16], but we include them here for completeness of the presentation. Thereafter we focus on the evaluation of an effective number of iteration vectors.

2.1. The basic equations

Let $\mathbf{K}$ and $\mathbf{M}$ be the stiffness and mass matrices of a finite element system with n degrees of freedom, and consider the generalized symmetric eigenvalue problem

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

We seek the smallest p eigenvalues λ_i, $i = 1,\ldots,p$, and corresponding eigenvectors $\boldsymbol{\varphi}_i$, $i = 1,\ldots,p$, with the ordering

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

which satisfy

$$\mathbf{K}\boldsymbol{\varphi}_i = \lambda_i\mathbf{M}\boldsymbol{\varphi}_i; \quad i = 1,\ldots,p \tag{3}$$

and the Kronecker delta relationships

$$\begin{aligned} \boldsymbol{\varphi}_i^T\mathbf{M}\boldsymbol{\varphi}_j &= \delta_{ij} \\ \boldsymbol{\varphi}_i^T\mathbf{K}\boldsymbol{\varphi}_j &= \lambda_i\delta_{ij} \end{aligned} \tag{4}$$

If the smallest eigenvalue is actually equal to zero, a shift can be used to reach the situation given in Eq. (2) [16]. The basic equations used in the subspace iteration method are, for k = 1, 2, ...,

$$\mathbf{K}\overline{\mathbf{X}}_{k+1} = \mathbf{M}\mathbf{X}_k \tag{5}$$

$$\mathbf{K}_{k+1} = \overline{\mathbf{X}}_{k+1}^T\mathbf{K}\overline{\mathbf{X}}_{k+1} \tag{6}$$

$$\mathbf{M}_{k+1} = \overline{\mathbf{X}}_{k+1}^T\mathbf{M}\overline{\mathbf{X}}_{k+1} \tag{7}$$

$$\mathbf{K}_{k+1}\mathbf{Q}_{k+1} = \mathbf{M}_{k+1}\mathbf{Q}_{k+1}\boldsymbol{\Lambda}_{k+1} \tag{8}$$

$$\mathbf{X}_{k+1} = \overline{\mathbf{X}}_{k+1}\mathbf{Q}_{k+1} \tag{9}$$

In practice, it is effective to order the iteration vectors in $\mathbf{X}_k$ naturally from the first to the last columns such that these correspond to increasing eigenvalues. Then the first vector in $\mathbf{X}_k$ corresponds to the eigenvector approximation of $\boldsymbol{\varphi}_1$ and the qth vector to the eigenvector approximation of $\boldsymbol{\varphi}_q$. The calculated approximations to the eigenvalues are given in $\boldsymbol{\Lambda}_{k+1}$.

There are three distinct steps of solution.

First, the q starting iteration vectors in $\mathbf{X}_1$ are established, $q > p$, where $\mathbf{X}_1$ is a matrix of dimension $n \times q$.

Second, the iteration is performed, using Eqs. (5)–(9), for $k = 1,2,\ldots$, until the convergence tolerance is satisfied. Let $\lambda_i^{(k+1)}$ be the approximation for λ_i calculated in the kth iteration, $k \geqslant 2$, then we have convergence to an accuracy of $2s$ digits in the required eigenvalues when for $tol = 10^{-2s}$

$$\left[1 - \frac{\left(\lambda_i^{(k+1)}\right)^2}{\left(\mathbf{q}_i^{(k+1)}\right)^T\mathbf{q}_i^{(k+1)}}\right]^{1/2} \leqslant tol; \quad i = 1,\ldots,p \tag{10}$$

where $\mathbf{q}_i^{(k+1)}$ is the i*th* vector in the matrix $\mathbf{Q}_{k+1}$ corresponding to $\lambda_i^{(k+1)}$ [16]. The eigenvector approximations will only be accurate to s digits. The theoretical convergence rates of these vectors are $\frac{\lambda_i}{\lambda_{q+1}}$, with hence a higher convergence rate to an eigenvector corresponding to a smaller eigenvalue [14,16]. While theoretical, these convergence rates are usually also observed in practice.

Third, the Sturm sequence check is carried out to ensure that the lowest p eigenpairs, that is $(\lambda_i, \boldsymbol{\varphi}_i)$ for $i = 1,\ldots p$, have indeed been calculated [1,16]. In case the Sturm sequence check is not passed, it is usually best to continue the iteration with a larger number of iteration vectors. In practice, this condition is hardly encountered provided a large enough number of iteration vectors is employed.

Using the above equations, it is crucial to establish effective starting iteration vectors, considering the *quality* and the *number of vectors*. The *quality* of the starting iteration vectors is important, because theory tells that if the subspace spanned by these vectors contains the exact eigenvectors, then a single iteration will calculate the exact eigenvalues and vectors sought. Nevertheless, in the present paper, we choose to use the simple algorithm of Ref. [1], see also Ref. [16], to establish the starting iteration vectors because we want to focus on other aspects of the solution scheme.

However, it should be noted that starting iteration vectors of much better quality may be generated or known from a previous solution. The eigenvectors just computed can be used, for example, in optimization problems of structures when the frequencies are calculated as the structure changes [16], in solving random eigenvalue problems when using Monte Carlo simulations [13], or in computational biology when evaluating the frequencies and mode shapes of proteins on conformational pathways [19]. In these cases, the use of the calculated eigenvectors of the previous solution as the starting vectors of the next solution can be very effective.

When these conditions do not apply and the order of the matrices n is large, particularly good quality starting iteration vectors may be generated, for example, using a reduction technique or the method of component mode synthesis [16]. Using substructuring smaller systems would be solved, even only approximately, and the solutions of those would be used to establish good starting iteration vectors in $\mathbf{X}_1$ for the complete system solution. This approach can be quite effective if 'typical smaller systems' can be identified which in the complete system repeat themselves, so that the eigensolution of a small problem can be used a multiple number of times in establishing the starting iteration vectors.

Whichever algorithm is used to establish the starting iteration vectors, an effective *number of vectors* q is important because the convergence rate to an eigenvector is given by $\frac{\lambda_i}{\lambda_{q+1}}$. In general, if q is small, but of course larger than p, we need a relatively large number of iterations to converge, while if q is large, we only need a few iterations to converge but in this case each iteration requires more computations. Hence, the use of an effective value of q is desirable and we address the calculation of such value in the next section.

Please cite this article in press as: Bathe K.J. The subspace iteration method – Revisited. Comput Struct (2012), http://dx.doi.org/10.1016/j.compstruc.2012.06.002

2.2. An algorithm to calculate q

A fundamental observation regarding eigenvalues of structures is that the values frequently increase in some known functional form, or at least in some approximate "guessed" functional form.

If we know the functional form, we can use this fact to increase the effectiveness of the subspace iteration method by finding an effective value of q. In the analysis given next, we follow the derivation used in Ref. [19] where it was assumed that the eigenvalues vary linearly in magnitude (as it is approximately the case for proteins).

Assume that the functional form is given by the simple distribution

$$\lambda_k = \lambda_1 + a_1(k-1) + a_2(k-1)^2 + a_3(k-1)^3 + a_4(k-1)^4 \tag{11}$$

where the a_1, a_2, a_3, and a_4 are given values, and $k = 1, 2, \ldots$ The quadratic term corresponds, for example, to the simple problem of a tensioned string, but the complete formula is easily used for structural analysis.

With the mentioned ordering of iteration vectors used in $\mathbf{X}_k$, the last iteration vector to converge to an eigenvector of interest is the pth vector in $\mathbf{X}_k$ and its rate of convergence is $\frac{\lambda_p}{\lambda_{q+1}}$. Let $\boldsymbol{\varepsilon}^{(i)}$ be the vector difference between the pth eigenvector and its approximation after i iterations, then we have the norm relationship

$$\|\boldsymbol{\varepsilon}^{(i)}\| = (\lambda_p/\lambda_{q+1})^i \|\boldsymbol{\varepsilon}^{(0)}\| \tag{12}$$

where $\boldsymbol{\varepsilon}^{(0)}$ is the initial error vector. To reach s digits of accuracy in the eigenvector approximation we therefore need to have that (considering the infinity norm)

$$(\lambda_p/\lambda_{q+1})^i \|\boldsymbol{\varepsilon}^{(0)}\| \leqslant 10^{-s} \tag{13}$$

and hence we require l iterations for the vector to converge, where l is given by

$$l = \frac{\ln[10^{-s}/\|\boldsymbol{\varepsilon}^{(0)}\|]}{\ln[\lambda_p/\lambda_{q+1}]} \tag{14}$$

We next use that the eigenvalue magnitudes increase as given in Eq. (11), which directly gives

$$\frac{\lambda_p}{\lambda_{q+1}} = \frac{\lambda_1 + a_1(p-1) + a_2(p-1)^2 + a_3(p-1)^3 + a_4(p-1)^4}{\lambda_1 + a_1(q) + a_2(q)^2 + a_3(q)^3 + a_4(q)^4} \tag{15}$$

We now assume that the norm of $\boldsymbol{\varepsilon}^{(0)}$ is the same irrespective of what value of q is used, that p and q are large, and that λ_1 in the numerator and denominator can be neglected. If λ_1 were too large to neglect, a shift could be applied [16]. Then we directly obtain

$$l = \frac{\ln[10^{-s}/\|\boldsymbol{\varepsilon}^{(0)}\|]}{\ln\left(\frac{p + c_1p^2 + c_2p^3 + c_3p^4}{q + c_1q^2 + c_2q^3 + c_3q^4}\right)} \tag{16}$$

where $c_1 = a_2/a_1$, $c_2 = a_3/a_1$, $c_3 = a_4/a_1$ (with a_1 assumed to be nonzero). However, with this relationship the number of numerical operations required for l iterations with q vectors, assuming $n \gg q$, are [16]

$$OPC = \frac{\ln[10^{-s}/\|\boldsymbol{\varepsilon}^{(0)}\|]}{\ln\left(\frac{p + c_1p^2 + c_2p^3 + c_3p^4}{q + c_1q^2 + c_2q^3 + c_3q^4}\right)}(\beta nmq + 2nq^2 + 3nq) \tag{17}$$

where n is the size and m is the 'effective' full half-bandwidth of the matrix $\mathbf{K}$, and $\beta = 2$ and 4 for a lumped mass matrix and a consistent mass matrix, respectively. In the case of the consistent mass matrix, we assume that $\mathbf{M}$ has the same bandwidth as $\mathbf{K}$ when in fact its bandwidth is, in practice, usually much smaller because no factorization of $\mathbf{M}$ is involved.

Since our only purpose is to find the best value of q for each value of p, we can use

$$OPC = \frac{\alpha}{\ln\left(\frac{p + c_1p^2 + c_2p^3 + c_3p^4}{q + c_1q^2 + c_2q^3 + c_3q^4}\right)}(\beta nmq + 2nq^2 + 3nq) \tag{18}$$

where α is a negative unknown constant of unimportant magnitude for this analysis. This expression assumes that all operations are performed in core, that is, the effort of disk accessing is not included. Minimizing the expression with respect to q, we find an approximation to the value of q that requires the least amount of computations to obtain the p eigenvalues and vectors. We call this value q_{best}. Note that this analysis does of course not tell the actual computational effort needed for the solution but only that the minimum is approximately expended when employing q_{best}.

Fig. 1 shows the value of OPC using $\alpha = -10^{-12}$ for the cases of 1 and 10 million equations and different parameters. There are many combinations of parameters that can occur and we show the results of some key cases only, but also note that if $c_1 = c_3 = 0.0$ but $c_2 = 1.0$ and if $c_1 = c_2 = 0.0$ and $c_3 = 1.0$ we have practically the same curves as in Fig. 1 but scaled down. We notice that in all cases the curves flatten around their minima at about $2p$, so that we can use $q_{best} = 2p$. Although the actual minimum is reached at a larger value, the difference in OPC is relatively small and more memory is needed for a larger q. Similar observations are made when using different reasonable values of the constants c_i, $i = 1,2,3$ and (n, m), but of course the magnitude of the function OPC changes. Hence, considering this information we can see that a reasonable value of q is given by the very simple formula

$$q = \max\{p + 8, 2p\} \tag{19}$$

where we use $p + 8$ to have enough iteration vectors when p is small. Of course, if the exact distribution of the eigenvalues is known, the minimization of OPC in Eq. (18), in appropriately modified form, could be used to obtain a better value of q than given by Eq. (19). In fact, this approach was followed in Ref. [19] and the resulting solution times scaled linearly.

It is interesting to note that in the original development of the subspace iteration procedure the same expression in the brackets was used but its minimum [1,16]. Hence – historically – an opposite value of what we now propose in Eq. (19) was given decades ago. This was so, because the experience at that time was based on seeking a much smaller number of eigenpairs (typically less than 20 pairs only) than in today's practice and the computers were much less powerful, in terms of computational speed and memory available. Of course, the analysis given above was not performed, but in particular, with the large computer memory available at present, a much larger number of iteration vectors can now be used effectively. These iteration vectors can in particular be processed efficiently in the parallel processing considered next.

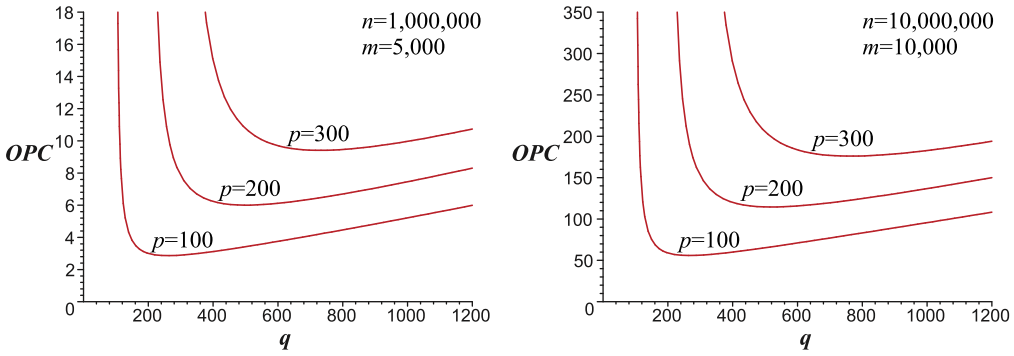
3. Shared and distributed parallel processing

The subspace iteration scheme lends itself very well to parallel processing. In essence, the following arrangements are used effectively in the parallelization of the computations.

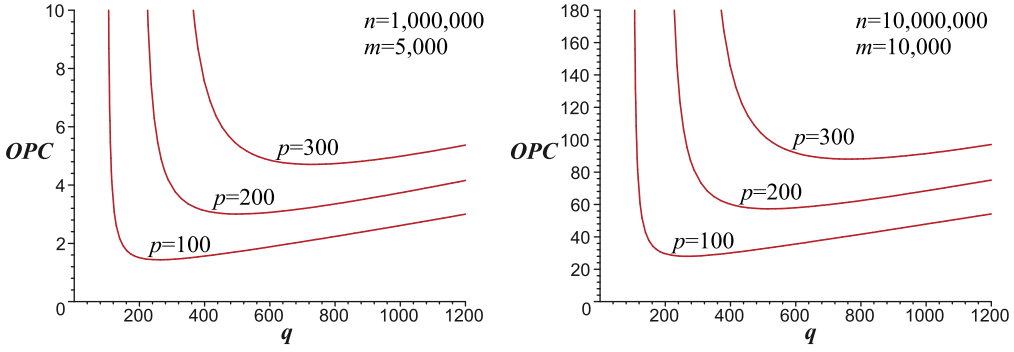
Assume that we have N nodes with M cores on each node. A relatively small but quite powerful computing environment might have N = 4 and M = 8. Hence, here we would use one root node and 3 additional nodes, with a total of 32 cores.

Considering Eqs. (5)–(7) and (9), in which most of the computational effort is expended, the q iteration vectors are partitioned horizontally by N and vertically by M. Hence, for the example given, the matrix $\mathbf{X}_k$ being of dimension $n \times q$ is "sliced" as shown in Fig. 2. The coefficient matrices $\mathbf{K}$ and $\mathbf{M}$ are also automatically

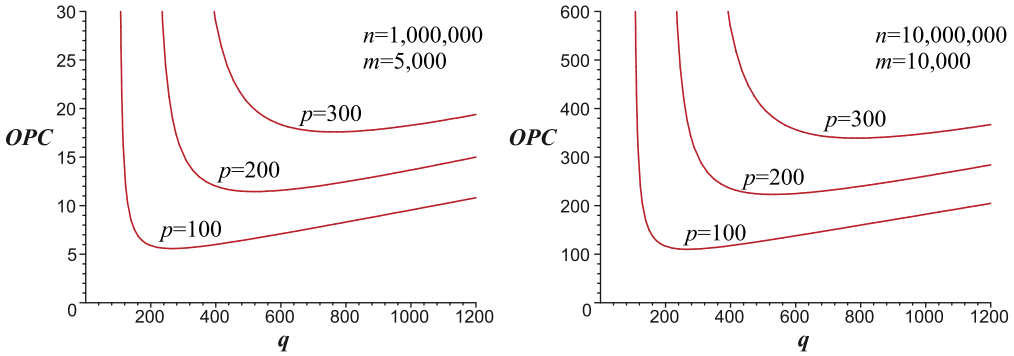
Please cite this article in press as: Bathe K.J. The subspace iteration method – Revisited. Comput Struct (2012), http://dx.doi.org/10.1016/j.compstruc.2012.06.002



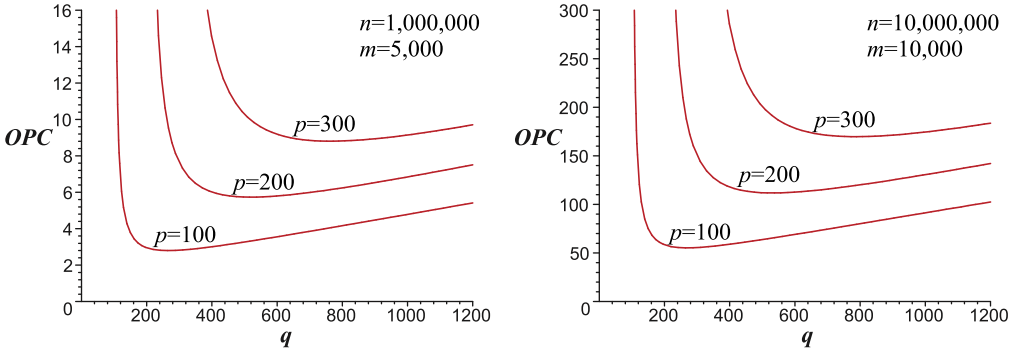
(a) The case of lumped mass matrix and $c_i = 0.0$, for all i.



(b) The case of lumped mass matrix and $c_1 = 1.0$ and $c_2 = c_3 = 0.0$



(c) The case of consistent mass matrix and $c_i = 0.0$, for all i.



(d) The case of consistent mass matrix and $c_1 = 1.0$ and $c_2 = c_3 = 0.0$

Fig. 1. The number of numerical operations *OPC* as a function of q for different cases of (n, m) and p; Eq. (18) is used with $\alpha = -10^{-12}$.

Please cite this article in press as: Bathe K.J. The subspace iteration method – Revisited. Comput Struct (2012), http://dx.doi.org/10.1016/j.compstruc.2012.06.002

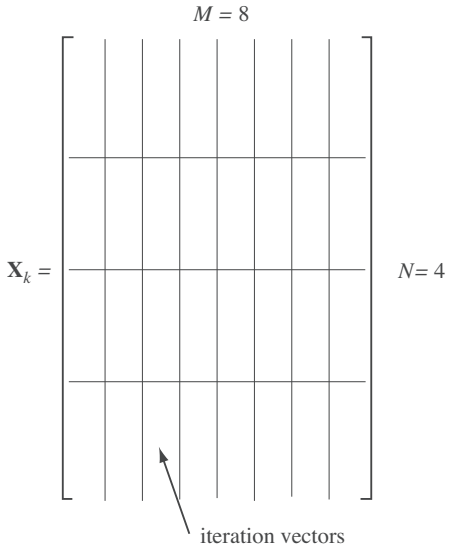


Fig. 2. The slicing of the matrices used in parallel processing; N = number of nodes working in DMP; M = number of cores on each node.

rearranged to have minimum coupling for the horizontal slicing used for the vectors.

In the DMP computations, each node operates on the horizontal slices of the matrices, whereas in the SMP computations of each node the vertical slices are used. This is a natural way to proceed and we note that these computations can be performed simultaneously and in a very efficient manner.

4. Illustrative solutions

In the illustrative solutions, the formula for q given in Eq. (19) was used, and since many eigenpairs are computed $q = 2p$. As mentioned already, the simple way of generating the starting iteration vectors given in Refs. [1,16] was used. Also, no acceleration techniques, as published for example in Refs.[4–10], were employed. The solutions were obtained using an HP cluster consisting of 4 nodes, each with dual quad-core processors running at 2.67 GHz connected by InfiniBand. The memory available was 48 GB on the root node and 24 GB on each of the other 3 nodes. The eigenvalues were calculated (with the default value s = 3) to 6 or more digits of accuracy. The first solution example is a 'constructed' benchmark test, and the next two examples are actual industry cases solved with ADINA, see Tech Briefs [20]. In all cases the consistent mass matrix was used.

4.1. Solution of beam problem

Fig. 3 shows the problem solved. The solution was obtained using 3D 8-node brick elements, with a mesh of 12 × 12 elements in the cross-section and 10,000 element layers along the length. This mesh results into 5,069,493 degrees of freedom. Fig. 4 gives the solution times used.

Of course, this is a somewhat academic problem, but the solution times are interesting. In fact, Fig. 4 shows that the solution times increase almost linearly with the number of calculated frequencies and mode shapes.

4.2. Solution of wheel assemblage

Fig. 5 shows the wheel assemblage, which was modeled with 3D brick elements and shell elements, leading to a total of 2,441,812 degrees of freedom and about 5,000 contact segments.

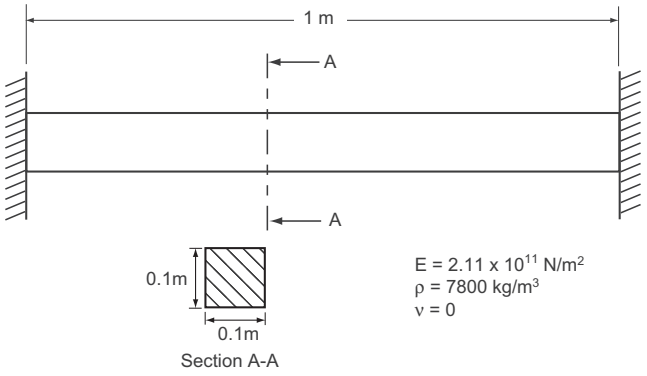


Fig. 3. Beam problem solved for lowest frequencies and mode shapes.

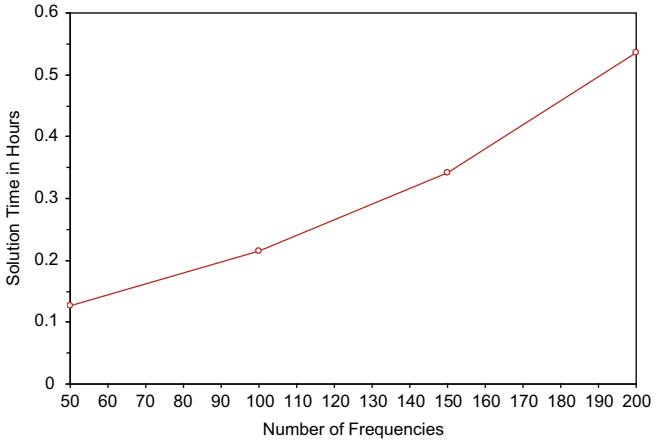


Fig. 4. Solution times for frequencies and mode shapes of the beam problem.

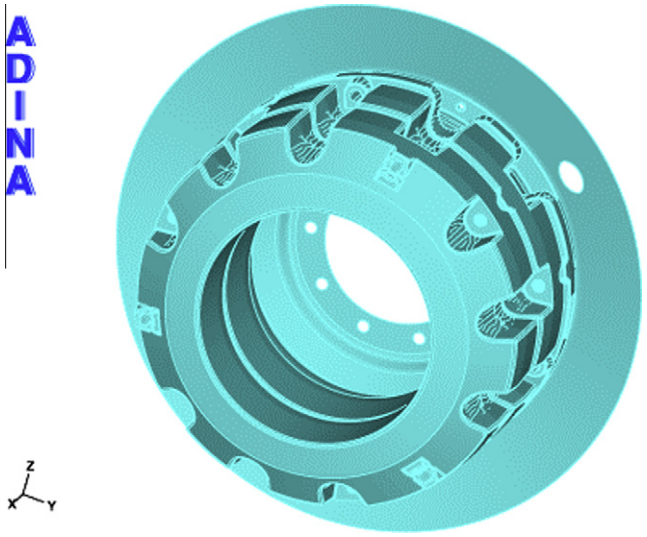


Fig. 5. Bolted wheel assembly.

The solution times used to solve the model are given in Fig. 6. Here we should also note that the solution times scale practically linearly.

Please cite this article in press as: Bathe K.J. The subspace iteration method – Revisited. Comput Struct (2012), http://dx.doi.org/10.1016/j.compstruc.2012.06.002

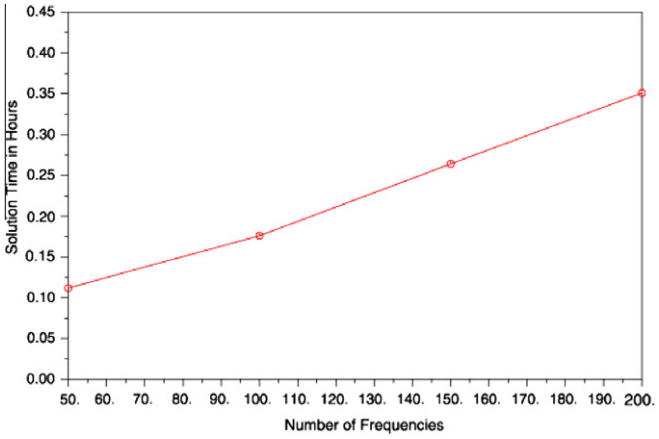


Fig. 6. Solution times for frequencies and mode shapes of the bolted wheel assembly model.

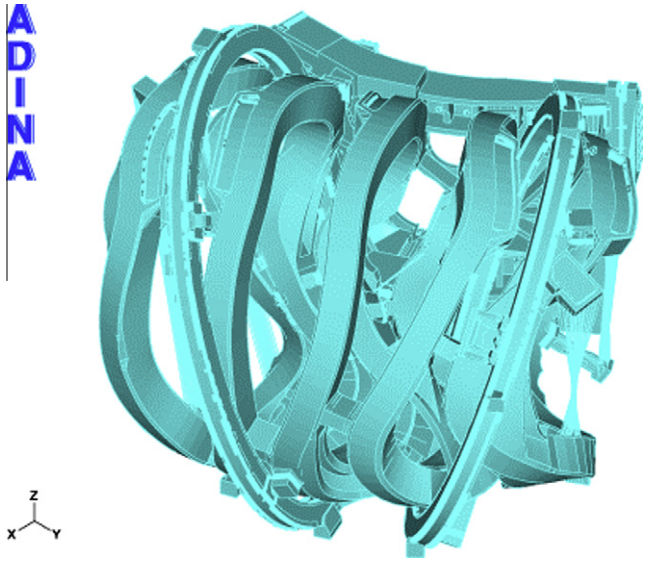


Fig. 7. Coil and support structure of a plasma fusion device.

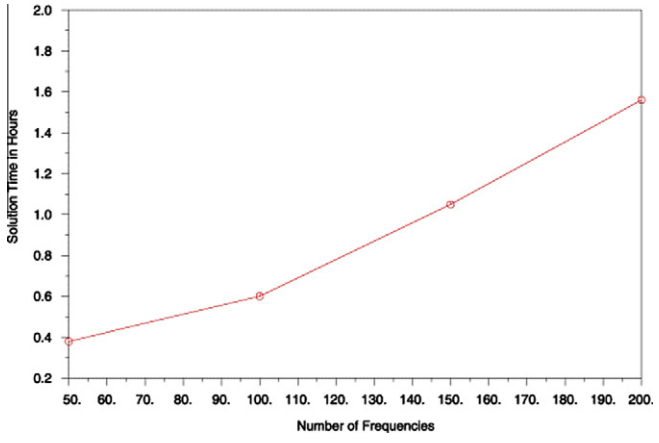


Fig. 8. Solution times for frequencies and mode shapes of the plasma fusion device model.

4.3. Solution of plasma fusion device

Fig. 7 shows the model of the plasma fusion device solved. This model was presented and studied in Ref. [21] and is a valuable model to identify the performance of the subspace iteration method.

The finite element model of the coil and support structure results into 5,865,765 degrees of freedom using mainly 3D brick elements and about 120,000 contact segments.

Fig. 8 gives the solution times used. Here too we see approximately a linear increase in the solution times.

5. Concluding remarks

The objective in this paper was to present some recent developments regarding the subspace iteration method for the solution of frequencies and mode shapes of structural systems. The paper can be thought of as an addendum to the work published about 4 decades ago, in which the original development of the method was presented [1].

We discussed in this paper that the use of an effective number of iteration vectors q to find p eigenpairs is very important and we established the simple formula that we are using at present. We also briefly discussed the use of parallel processing. With these schemes we gave some solution times and found that for the structural problems solved these times scaled, approximately, linearly with the number of eigenpairs sought. Of course, the actual scaling seen is in general problem-dependent.

Since the hardware environments for computations have improved tremendously during the last decades, with regard to processing speed and memory available, and solution requirements in finite element analyses have greatly increased, it is only natural that computational procedures, proposed some years ago, need to be brought up to date for maximum efficiency – like addressed in this paper for the subspace iteration method.

References

[1] K.J. Bathe, Solution Methods for Large Generalized Eigenvalue Problems in Structural Engineering, Report UCSESM 71-20, Department of Civil Engineering, University of California, Berkeley, 1971.
[2] Bathe KJ, Wilson EL. Solution methods for eigenvalue problems in structural mechanics. Int. J. Numer. Methods Eng. 1973;6:213–26.
[3] Bathe KJ, Wilson EL. Large eigenvalue problems in dynamic analysis. ASCE J. Eng. Mech. Div. 1972;98:1471–85.
[4] Akl F, Dilger WH, Irons BM. Over-relaxation and subspace iteration. Int. J. Numer. Methods Eng. 1979;14:629–30.
[5] Bathe KJ, Ramaswamy S. An accelerated subspace iteration method. J. Comput. Methods Appl. Mech. Eng. 1980;23:313–31.
[6] Wilson EL, Itoh T. An eigensolution strategy for large systems. Comput. Struct. 1983;16:259–65.
[7] Qian YY, Dhatt G. An accelerated subspace method for generalized eigenproblems. Comput. Struct. 1995;54:1127–34.
[8] Jung HJ, Kim MC, Lee IW. An improved subspace iteration method with shifting. Comput. Struct. 1999;70:625–33.
[9] Wang X, Zhou J. An accelerated subspace iteration method for generalized eigenproblems. Comput. Struct. 1999;71:293–301.
[10] Zhao QC, Chen P, Peng WB, Gong YC, Yuan MW. Accelerated subspace iteration with aggressive shift. Comput. Struct. 2007;85:1562–78.
[11] Bauer FL. Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischer Eigenwertprobleme. Zeitschrift für Angewandte Mathematik und Physik 1957;8:214–35.
[12] Rutishauser H. Computational aspects of F.L. Bauer's simultaneous iteration method. Numer. Math. 1969;13:4–13.
[13] Pradlwarter HJ, Schuëller GI, Szekely GS. Random eigenvalue problems for large systems. Comput. Struct. 2002;80:2415–24.
[14] Bathe KJ. Convergence of Subspace Iteration. In: Bathe KJ, Oden JT, Wunderlich W, editors. Formulations and Computational Algorithms in Finite Element Analysis. M.I.T. Press; 1977.
[15] Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Nat. Bur. Stand. 1950;45:255–82.
[16] Bathe KJ. Finite Element Procedures. Prentice Hall; 1996.
[17] Paige CC. Computational variants of the Lanczos method for the eigenproblem. IMA J. Appl. Math. 1972;10:373–81.
[18] 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.

Please cite this article in press as: Bathe K.J. The subspace iteration method – Revisited. Comput Struct (2012), http://dx.doi.org/10.1016/j.compstruc.2012.06.002

[19] Sedeh RS, Bathe M, Bathe KJ. The subspace iteration method in protein normal mode analysis. J. Comput. Chem. 2010;31:66–74.
[20] ADINA, www.adina.com.
[21] Jaksic N, van Eeten P, Bykov V, Schauer F. Analysis of the magnet support structure for the plasma fusion experiment Wendelstein 7-X. Comput. Struct. 2011;89:1177–91.

Please cite this article in press as: Bathe K.J. The subspace iteration method – Revisited. Comput Struct (2012), http://dx.doi.org/10.1016/j.compstruc.2012.06.002