Pergamon

0045-7949(94)00565-6

Computers & Structures Vol. 56, No. 4, pp. 605–613, 1995

0045-7949/95 $9.50 + 0.00

BI-LANCZOS WITH PARTIAL ORTHOGONALIZATION

H. I. van der Veen†‡ and **K. Vuik†**

†Faculty of Technical Mathematics and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands

‡TNO Bouw, Department of Computational Mechanics, P.O. Box 49, 2600 AA Delft, The Netherlands

(*Received* 28 *February* 1994)

Abstract—In theory the bi-Lanczos algorithm is an attractive possibility for solving the unsymmetric eigenproblem. In practice, however, it turns out to be unstable due to loss of orthogonality of the iteration vectors. In this paper we discuss the possibilities of reorthogonalizing the bi-Lanczos iteration vectors. To save part of the advantage of the bi-Lanczos method over the other most commonly used Krylov subspace based method of Arnoldi (bi-Lanczos lacks the growth of the number of vector operations per iteration), we propose partial reorthogonalization. In that case reorthogonalization takes place if and only if too much orthogonality is lost, so that the results are accurate enough, whereas the algorithm is still less time consuming than Arnoldi. The theory presented here is an extension of the theory available for partial reorthogonalization of the symmetric Lanczos algorithm.

1. INTRODUCTION

Large unsymmetric eigenproblems play an important role in a variety of applied sciences. In fluid and structural mechanics, for example, such problems arise quite often. One could think of bifurcation problems where the smallest eigenvector is an indication of the direction in which the solution should be continued after bifurcation. The characteristic direction of a certain flow can be analyzed by considering the eigenvalues and eigenvectors of a certain matrix. Furthermore, stable and efficient eigenproblem algorithms are of vital importance in buckling analysis and for the computation of dynamic responses.

To solve the non-symmetric eigenproblem as part of finite element analysis, iterative methods like the Krylov subspace based methods bi-Lanczos [7] and Arnoldi [5, p. 501] are most commonly used. An iterative method computes eigenvalues and eigenvectors in increasing accuracy, so that if interest is directed to a small dominant set of the eigenvalues and eigenvectors, which is the case for most finite element applications, it is possible to achieve satisfying results long before all eigenvalues and eigenvectors have been computed.

Although the method of Arnoldi is very robust and is widely used, it can become quite expensive, because of growing cost of work and memory per iteration, due to explicit reorthogonalization of the iteration vectors. Bi-Lanczos would be an attractive alternative, but unfortunately it loses orthogonality too often, which leads to ghost eigenvalues (non-realistic multiple eigenvalues) at the expense of internal eigenvalues. The bi-Lanczos method may not even converge at all, because of too great loss of orthogonality [14, p. 51]. Loss of orthogonality is noticed even in symmetric problems. Nevertheless, we are unaware of a publication on bi-Lanczos with reorthogonalization techniques. Breakdown can be a problem too for the bi-Lanczos method, but besides the fact that for smaller problems this hardly ever occurs, much research had been done in this subject already. A result of this research is for example the "Look Ahead" (bi-)Lanczos algorithm [4].

This paper discusses the possibilities of reorthogonalization of the bi-Lanczos method. Full reorthogonalization is not attractive, because that would result in a method more time-consuming than Arnoldi. Instead we consider a simple form of partial reorthogonalization proposed by Nour-Omid for the Lanczos method [6, pp. 565–600]. We will start with the description of how to reorthogonalize the iteration vectors, and continue with the necessary computations to obtain a successful partial reorthogonalization bi-Lanczos algorithm, which we have called the bilapo algorithm.

2. REORTHOGONALIZATION OF THE BI-LANCZOS METHOD

The bi-Lanczos method realizes a projection of an $n \times n$ matrix $\mathbf{A}$ on the Krylov space $\mathbf{K}_k$ $(k \leqslant n)$ spanned by the power basis $\{\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}^{k-1}\mathbf{v}_1\}$, where $\mathbf{v}_1$ is an initial vector. The reduction results in the construction of a tridiagonal non-symmetric matrix $\mathbf{T}_k$ called the Ritz matrix. After $k = n$ steps this reduction can be summarized by: $\mathbf{A}\mathbf{V}_n = \mathbf{V}_n\mathbf{T}_n$ and $\mathbf{A}^T\mathbf{W}_n = \mathbf{W}_n\mathbf{T}_n^T$. The columns of $\mathbf{W}_n$ and $\mathbf{V}_n$ are

bi-orthogonal: $\mathbf{W}_n^T\mathbf{V}_n = \mathbf{I}_n$. The Ritz matrix $\tilde{\mathbf{T}}_n$ has the following form:

$$\tilde{\mathbf{T}}_n = \begin{bmatrix} \alpha_1 & \gamma_1 & & & 0 \\ \beta_1 & \alpha_2 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \gamma_{n-1} \\ 0 & & & \beta_{n-1} & \alpha_n \end{bmatrix}. \quad (1)$$

The process can be derived by equating columns in the relations $\mathbf{AV}_k = \mathbf{V}_{k+1}\mathbf{T}_{k+1}$ and $\mathbf{A}^T\mathbf{W}_k = \mathbf{W}_{k+1}\mathbf{T}_{k+1}^T$ and $\mathbf{W}_{k+1}^T\mathbf{V}_{k+1} = \mathbf{I}_{k+1}$. The matrices $\mathbf{T}_{k+1}$ and $\mathbf{T}_{k+1}^T$ have the form:

$$\mathbf{T}_{k+1} = \begin{bmatrix} \tilde{\mathbf{T}}_k & \\ 0 \cdots 0 & \beta_k \end{bmatrix} \mathbf{T}_{k+1}^T = \begin{bmatrix} & 0 \\ \tilde{\mathbf{T}}_k & \vdots \\ & 0 \\ & \gamma_k \end{bmatrix}. \quad (2)$$

To satisfy the bi-orthogonality property, the initial vectors $\mathbf{v}_1$ and $\mathbf{w}_1$ must be chosen such that $|\mathbf{w}_1^T\mathbf{v}_1| = 1$. The norm ($\|\cdot\|$) used in this paper is the l_2-norm. The bi-Lanczos method is given by code 1.

Loss of bi-orthogonality of the iteration vectors $\mathbf{v}$ and $\mathbf{w}$ can take serious forms. To re-establish bi-orthogonality at step k, for example, we project the vectors $\mathbf{v}_k$ and $\mathbf{w}_k$ onto spaces spanned by the old iterations vectors, namely the column space of $\mathbf{W}_{k-1}$ and $\mathbf{V}_{k-1}$. We use for this the projection matrices $\mathbf{W}_{k-1}\mathbf{V}_{k-1}^T$ and $\mathbf{V}_{k-1}\mathbf{W}_{k-1}^T$. A specific matrix $\mathbf{P}$ is a projection matrix if and only if it is idempotent [3, p. 3]. $\mathbf{P}^2 = \mathbf{P}$. This property can be proven easily for the matrices $\mathbf{W}_{k-1}\mathbf{V}_{k-1}^T$ and $\mathbf{V}_{k-1}\mathbf{W}_{k-1}^T$:

$$(\mathbf{W}_{k-1}\mathbf{V}_{k-1}^T)^2 = \mathbf{W}_{k-1}\mathbf{V}_{k-1}^T \times \mathbf{W}_{k-1}\mathbf{V}_{k-1}^T = \mathbf{W}_{k-1}\mathbf{V}_{k-1}^T \quad (3)$$

and

$$(\mathbf{V}_{k-1}\mathbf{W}_{k-1}^T)^2 = \mathbf{V}_{k-1}\mathbf{W}_{k-1}^T \times \mathbf{V}_{k-1}\mathbf{W}_{k-1}^T = \mathbf{V}_{k-1}\mathbf{W}_{k-1}^T \quad (4)$$

because of the bi-orthogonality property $\mathbf{V}_{k-1}^T\mathbf{W}_{k-1} = \mathbf{W}_{k-1}^T\mathbf{V}_{k-1} = \mathbf{I}_{k-1}$. But these matrices do not define an orthogonal projection because they are not symmetric. The matrix $\mathbf{W}_{k-1}\mathbf{V}_{k-1}^T$ projects the column space of $\mathbf{W}_{k-1}$ onto itself: $\mathbf{W}_{k-1}\mathbf{V}_{k-1}^T\mathbf{W}_{k-1} = \mathbf{W}_{k-1}$, and similarly for $\mathbf{V}_{k-1}\mathbf{W}_{k-1}^T$ and $\mathbf{V}_{k-1}$. Wilkinson [15, pp. 391–393] shows that $\mathbf{W}_{k-1}\mathbf{V}_{k-1}^T$ can be used to re-establish bi-orthogonality of the left iteration vectors $\mathbf{w}_k$, whereas $\mathbf{V}_{k-1}\mathbf{W}_{k-1}^T$ is used to re-establish bi-orthogonality of the right iteration vectors $\mathbf{v}_k$.

The implication of these projections in practice is the following. Suppose one wishes to reorthogonalize a vector $\mathbf{w}_k$ to reestablish bi-orthogonality. Reorthogonalization of $\mathbf{w}_k$ can then be written as:

$$\tilde{\mathbf{w}}_k = \mathbf{w}_k - \mathbf{W}_{k-1}\mathbf{V}_{k-1}^T\mathbf{w}_k \quad (5)$$

$$= \mathbf{w}_k - [(\mathbf{v}_1^T\mathbf{w}_k)\mathbf{w}_1 + (\mathbf{v}_2^T\mathbf{w}_k)\mathbf{w}_2 + \cdots + (\mathbf{v}_{k-1}^T\mathbf{w}_k)\mathbf{w}_{k-1}]. \quad (6)$$

In a similar way the proper formulas for the reorthogonalization of $\mathbf{v}_k$ can be established. The result is

$$\tilde{\mathbf{v}}_k = \mathbf{v}_k - [(\mathbf{w}_1^T\mathbf{v}_k)\mathbf{v}_1 + (\mathbf{w}_2^T\mathbf{v}_k)\mathbf{v}_2 + \cdots + (\mathbf{w}_{k-1}^T\mathbf{v}_k)\mathbf{v}_{k-1}]. \quad (7)$$

If the iteration vectors are not normalized, $\mathbf{v}_k^T\mathbf{w}_k \neq 1$, then obviously the reorthogonalization should be performed as follows:

$$\tilde{\mathbf{w}}_k = \mathbf{w}_k - \sum_{i=1}^{k-1} \frac{(\mathbf{v}_i^T\mathbf{w}_k)\mathbf{w}_i}{\mathbf{v}_i^T\mathbf{w}_i}$$

and

$$\tilde{\mathbf{v}}_k = \mathbf{v}_k - \sum_{i=1}^{k-1} \frac{(\mathbf{w}_i^T\mathbf{v}_k)\mathbf{v}_i}{\mathbf{w}_i^T\mathbf{v}_i}. \quad (8)$$

To be certain that the orthogonalization is correct, we must show that after reorthogonalization of $\mathbf{v}_k$ and $\mathbf{w}_k$ it is still true that $\mathbf{V}_{k-1}^T\tilde{\mathbf{w}}_k = \mathbf{W}_{k-1}^T\tilde{\mathbf{v}}_k = 0$, if we suppose (induction) that the bi-orthogonality property holds until the $(k-1)$th iteration. Therefore, premultiply the first part of eqn (8) by $\mathbf{v}_j^T$ and the second part by $\mathbf{w}_j^T$, for $j < k$:

$$\mathbf{v}_j^T\tilde{\mathbf{w}}_k = \mathbf{v}_j^T\mathbf{w}_k - \sum_{i=1}^{k-1} \frac{(\mathbf{v}_i^T\mathbf{w}_k)\mathbf{v}_j^T\mathbf{w}_i}{\mathbf{v}_i^T\mathbf{w}_i} = \mathbf{v}_j^T\mathbf{w}_k - \mathbf{v}_j^T\mathbf{w}_k = 0 \quad (9)$$

$$\mathbf{w}_j^T\tilde{\mathbf{v}}_k = \mathbf{w}_j^T\mathbf{v}_k - \sum_{i=1}^{k-1} \frac{(\mathbf{w}_i^T\mathbf{v}_k)\mathbf{w}_j^T\mathbf{v}_i}{\mathbf{w}_i^T\mathbf{v}_i} = \mathbf{w}_j^T\mathbf{v}_k - \mathbf{w}_j^T\mathbf{v}_k = 0, \quad (10)$$

which is what we wanted to find.

Summarizing, the projection technique to re-establish semi-orthogonality includes the performance of two summations of inner products taken and requires twice as many floating point operations per iteration as the reorthogonalization of Gram–Schmidt in, for example, the Arnoldi method. Furthermore, the reorthogonalization must be performed during two successive iterations. Otherwise it would be useless, since the recurrence for the next iteration vector is

constructed from the last *two* iteration vectors:

$$\beta_k \mathbf{v}_{k+1} = \mathbf{A}\mathbf{v}_k - \alpha_k \mathbf{v}_k - \gamma_{k-1}\mathbf{v}_{k-1}$$

and

$$\gamma_k \mathbf{w}_{k+1} = \mathbf{A}^T\mathbf{w}_k - \alpha_k \mathbf{w}_k - \beta_{k-1}\mathbf{w}_{k-1}. \quad (11)$$

Concluding, bi-Lanczos can be attractive compared to Arnoldi only if reorthogonalization is performed less than every four iterations. That is why we will now turn our attention to partial reorthogonalization techniques. The next section covers recent results on the subject of partial reorthogonalization of the bi-Lanczos method and this paper is concluded with the performance of some tests.

3. PARTIAL REORTHOGONALIZATION OF BI-LANCZOS

The following results on partial reorthogonalization of bi-Lanczos are obtained by investigating and generalizing theory known for the symmetric Lanczos method on the subject of partial [13] and selective [9] reorthogonalization. We begin with a brief summary of the contents of these papers.

The bi-Lanczos method reduces to the Lanczos method if the matrix $\mathbf{A}$, of which one wants to find the eigenvalues, is symmetric. In that case the column spaces $\mathbf{V}_k$ and $\mathbf{W}_k$ are equal and thus $\mathbf{V}_k^T\mathbf{V}_k = \mathbf{I}_k$. The orthogonal matrix $\mathbf{V}_k$ is usually denoted by $\mathbf{Q}_k = [\mathbf{q}_1, \ldots, \mathbf{q}_k]$ and the resulting tridiagonal matrix $\mathbf{T}$ is symmetric. Full reorthogonalization of the iteration vectors $\mathbf{q}_k$ guarantees the stringent requirement of $|\mathbf{q}_i^T\mathbf{q}_k| < n\epsilon$ with $i < k$, i.e. orthogonality at roundoff level. However, the effort of explicit reorthogonalization of the new Lanczos vector $\mathbf{q}_{k+1}$ against all previous Lanczos vectors is not necessary. Numerical results [11] and theoretical considerations in connection with the various reorthogonalization methods [8] have shown that a more relaxed condition, namely $|\mathbf{q}_i^T\mathbf{q}_k| < \sqrt{\epsilon}$ is sufficient to permit the computation of eigenvalues without the appearance of spurious duplicate copies (copies of eigenvalues that turn up because of loss of orthogonality are referred to as spurious or ghost eigenvalues). The more relaxed condition $|\mathbf{q}_i^T\mathbf{q}_k| < \sqrt{\epsilon}$ is referred to in the literature as the semi-orthogonality condition. The analysis and numerical results can be summarized with the following.

Theorem 12

Let $\mathbf{T}_k$ be the tridiagonal matrix computed by the Lanczos algorithm, where by some means the Lanczos vectors are kept semi-orthogonal. Then $\mathbf{T}_k$ is, up to roundoff, the orthogonal projection of $\mathbf{A}$ onto span $(\mathbf{Q}_k)$.

Proof of this theorem can be found in Simon [12], theorem 2.5.

These results considering the symmetric Lanczos method have motivated us to investigate partial reorthogonalization of the bi-Lanczos method. In this paper we will restrict ourselves to numerical experiments. However, in the near future we hope to substantiate our results with theoretical considerations.

We will describe one way to maintain semi-orthogonality called partial reorthogonalization. This method is analogous to the method proposed by Simon [13] for the symmetric Lanczos algorithm.

To monitor loss of bi-orthogonality among the iteration vectors of the bi-Lanczos method, we set up relations for the elements of $\boldsymbol{\Omega}_k = \mathbf{W}_k^T\mathbf{V}_k$: $\omega_{i,j} = \mathbf{w}_i^T\mathbf{v}_j$. Explicit computation of the inner products $\mathbf{w}_i^T\mathbf{v}_j$ is far too expensive, because for every iteration we would need to compute one more inner product. In theory $\boldsymbol{\Omega}_k$ should be the identity matrix, but in practice, the off-diagonal elements depend on the machine precision ϵ. In the kth iteration the residual vectors of the bi-Lanczos algorithm are equal to:

$$\mathbf{r}_k = \beta_k \mathbf{v}_{k+1} = \mathbf{A}\mathbf{v}_k - \alpha_k \mathbf{v}_k - \gamma_{k-1}\mathbf{v}_{k-1} \quad (13)$$

$$\mathbf{p}_k = \gamma_k \mathbf{w}_{k+1} = \mathbf{A}^T\mathbf{w}_k - \alpha_k \mathbf{w}_k - \beta_{k-1}\mathbf{w}_{k-1}. \quad (14)$$

Premultiply eqn (13) by $\mathbf{w}_i^T$ and eqn (14) by $\mathbf{v}_i^T$ so that the relations are found for the elements of matrix $\boldsymbol{\Omega}_k$:

$$\mathbf{w}_i^T\beta_k\mathbf{y}_{k+1} = \mathbf{w}_i^T\mathbf{A}\mathbf{v}_k - \alpha_k\mathbf{w}_i^T\mathbf{v}_k - \gamma_{k-1}\mathbf{w}_i^T\mathbf{v}_{k-1} \quad (15)$$

$$\mathbf{v}_i^T\gamma_k\mathbf{w}_{k+1} = \mathbf{v}_i^T\mathbf{A}^T\mathbf{w}_k - \alpha_k\mathbf{v}_i^T\mathbf{w}_k - \beta_{k-1}\mathbf{v}_i^T\mathbf{w}_{k-1}. \quad (16)$$

Rearrange the terms and fill in the elements of $\boldsymbol{\Omega}_k$:

$$\mathbf{w}_i^T\mathbf{A}\mathbf{v}_k = \beta_k\omega_{i,k+1} + \alpha_k\omega_{i,k} + \gamma_{k-1}\omega_{i,k-1} \quad (17)$$

$$\mathbf{v}_i^T\mathbf{A}^T\mathbf{w}_k = \gamma_k\omega_{k+1,i} + \alpha_k\omega_{k,i} + \beta_{k-1}\omega_{k-1,i}. \quad (18)$$

Similar equations can be set up for the ith iteration in relation to the kth step:

$$\mathbf{w}_k^T\mathbf{A}\mathbf{v}_i = \beta_i\omega_{k,i+1} + \alpha_i\omega_{k,i} + \gamma_{i-1}\omega_{k,i-1} \quad (19)$$

$$\mathbf{v}_k^T\mathbf{A}^T\mathbf{w}_i = \gamma_i\omega_{i+1,k} + \alpha_i\omega_{i,k} + \beta_{i-1}\omega_{i-1,k}. \quad (20)$$

The matrix $\boldsymbol{\Omega}_k$ is not symmetric. But the left hand side of eqn (20) is equal to the left hand side of eqn (17), since both are the inner product of $\mathbf{A}\mathbf{v}_k$ and $\mathbf{w}_i$. Equation (18) can be related to eqn (19) in the same way, and the related equations can be subtracted from each other. This leads to:

$$\beta_k\omega_{i,k+1} + \gamma_{k-1}\omega_{i,k-1} + (\alpha_k - \alpha_i)\omega_{i,k} - \gamma_i\omega_{i+1,k} - \beta_{i-1}\omega_{i-1,k} = 0 \quad (21)$$

$$\beta_i\omega_{k,i+1} + \gamma_{i-1}\omega_{k,i-1} + (\alpha_i - \alpha_k)\omega_{k,i} - \gamma_k\omega_{k+1,i} - \beta_{k-1}\omega_{k-1,i} = 0. \quad (22)$$

Rewriting the equations, such that $\omega_{k+1,i}$ and $\omega_{i,k+1}$ can be easily computed:

$$\omega_{i,k+1} = [-\gamma_{k-1}\omega_{i,k-1} + (\alpha_i - \alpha_k)\omega_{i,k} + \gamma_i\omega_{i+1,k} + \beta_{i-1}\omega_{i-1,k}]/\beta_k \quad (23)$$

$$\omega_{k+1,i} = [-\beta_{k-1}\omega_{k-1,i} + (\alpha_i - \alpha_k)\omega_{k,i} + \beta_i\omega_{k,i+1} + \gamma_{i-1}\omega_{k,i-1}]/\gamma_k. \quad (24)$$

The recurrences (23) and (24) should be updated for $k > 1$, $i < k$. The diagonal elements of Ω_k are set equal to one to start with, and the off-diagonal elements $\omega_{k,k+1}$ and $\omega_{k+1,k}$ are set equal to $n\epsilon$. Again, ϵ denotes the machine precision and n the order of matrix **A**. This enables us to compute all elements of Ω_k.

As an example, suppose the iteration is at step $k + 1 = 4$. We then need to compute the last column and last row of Ω_4, that is $\omega_{1,4}$, $\omega_{2,4}$, $\omega_{4,1}$ and $\omega_{4,2}$,

$$\Omega_4 = \begin{bmatrix} 1 & n\epsilon & \omega_{1,3} & \omega_{1,4} \\ n\epsilon & 1 & n\epsilon & \omega_{2,4} \\ \omega_{3,1} & n\epsilon & 1 & n\epsilon \\ \omega_{4,1} & \omega_{4,2} & n\epsilon & 1 \end{bmatrix}. \quad (25)$$

These particular elements can be computed with recurrences (23) and (24). The elements of the last column follow from eqn (23):

$$\omega_{1,4} = [-\gamma_2\omega_{1,2} + (\alpha_1 - \alpha_3)\omega_{1,3} + \gamma_1\omega_{2,3}]/\beta_3 \quad (26)$$

$$\omega_{2,4} = [-\gamma_2\omega_{2,2} + (\alpha_2 - \alpha_3)\omega_{2,3} + \gamma_2\omega_{3,3} + \beta_1\omega_{1,3}]/\beta_3 \quad (27)$$

$$= [(\alpha_2 - \alpha_3)\omega_{2,3} + \beta_1\omega_{1,3}]/\beta_3. \quad (28)$$

The parameters $\omega_{1,2}$ and $\omega_{2,3}$ are equal to $n\epsilon$. The elements of the last row are computed with eqn (24):

$$\omega_{4,1} = [-\beta_2\omega_{2,1} + (\alpha_1 - \alpha_3)\omega_{3,1} + \beta_1\omega_{3,2}]/\gamma_3 \quad (29)$$

$$\omega_{4,2} = [-\beta_2\omega_{2,2} + (\alpha_2 - \alpha_3)\omega_{3,2} + \beta_2\omega_{3,3} + \gamma_1\omega_{3,1}]/\gamma_3 \quad (30)$$

$$= [(\alpha_2 - \alpha_3)\omega_{3,2} + \gamma_1\omega_{3,1}]/\gamma_3 \quad (31)$$

with $\omega_{2,1}$ and $\omega_{3,2}$ equal to $n\epsilon$.

If either $\omega_{i,k+1}$ or $\omega_{k+1,i}$, $i < j$, exceeds $\sqrt{n\epsilon}$, the algorithm pauses and performs reorthogonalization of iteration vectors v_{k+1} and w_{k+1} against all previous computed iteration vectors. The bi-orthogonality property is re-established and the ωs of the last row and column are set equal to $n\epsilon$. This is repeated in the next iteration to ensure complete recovery of bi-orthogonality. The bi-Lanczos algorithm then continues. Note that we use $\sqrt{n\epsilon}$ as reorthogonalization criterion. The use of $\sqrt{n\epsilon}$ instead of $\sqrt{\epsilon}$ seems logical considering the reset values of $n\epsilon$ after reorthogonalization. Some numerical experiments (see next section) have shown that the results computed with criterion $\sqrt{n\epsilon}$ are accurate, while fewer pauses are made by the algorithm.

At this point we have established all ingredients of the bilapo (*Bi-La*nczos with *P*artial *O*rthogonalization) algorithm. The proper semi-Fortran code is given by code 2.

A theoretical comparison

To compare bi-Lanczos and Arnoldi [1, 10] we consider the number of vector operations (copies, vector updates, inner products) and matrix–vector multiplications to be performed for each method, per iteration. The Arnoldi method needs per iteration only one matrix–vector multiplication, and approximately $(k^2 + k)/2$ vector operations due to full reorthogonalization. The bi-Lanczos method, on the other hand, needs two matrix–vector multiplications. If a reorthogonalization is necessary, $(k^2 + k)$ vector operations will need to be executed: $(k^2 + k)/2$ for both iteration vectors. Moreover, this reorthogonalization must be repeated in the next iteration, which means another $(k + 1)^2 + (k + 1)$ vector operations to ensure a successful reorthogonalization.

Although the number of vector operations needed to perform full reorthogonalization of the iteration vectors of the bi-Lanczos method seems to be quite large, it must be kept in mind that the reorthogonalization is needed only when loss of orthogonality has really occurred. This is in contrast to the Arnoldi method where full reorthogonalization, although cheaper, is performed every iteration step. The question is now, when is the bi-Lanczos method with partial reorthogonalization cheaper and when is it more expensive than the method of Arnoldi. Since the number of reorthogonalizations depends on the problem, there is unfortunately no straightforward answer to this question. Instead we will discuss two experiments in the next section. These experiments were performed on an Iris Silicon Graphics workstation. The problems are solved with a finite element software package Diana (*Di*placements *ana*lysis), maintained by and developed at TNO-Bouw, Building and Construction Research, in Rijswijk, The Netherlands.

4. EXPERIMENTS

4.1. *Two-dimensional convection–diffusion*

Consider a linear steady flow in a fully rectangular area (1×1 m^2, Fig. 1). The domain is covered with a grid consisting of 10×10 quadrilateral linear

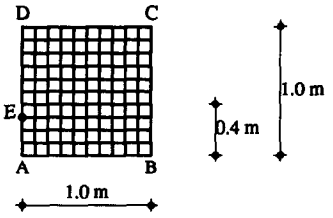

Fig. 1. Convection–diffusion grid.

elements. The two-dimensional convection–diffusion equation which underlies this problem can be written as:

$$-\epsilon\Delta\phi + \mathbf{c}\nabla\phi = 0, \tag{32}$$

where ϵ is the diffusion coefficient and $\mathbf{c}$ the convection vector. The convection term causes non-symmetry because of the discretization of the first-order derivatives. The finite element discretization most commonly used is the Galerkin method. It is used here without upwind terms so that the solution may contain undesired numerical wiggles. The element mesh leads to 90 linear equations to be solved and thus a matrix of order 90. The small size of this problem enables us to compute the eigenvalues with Eispack subroutines (public domain). They can be used as a reference for the bi-Lanczos method.

As characteristics of the flow we have chosen:

ED, BC, CD:	$\phi = 0$ m	potential
AE:	$\phi = 1$ m	potential
diffusion	0.001	$m^2 s^{-1}$
convection	(1.0, 0.2)	$m s^{-1}$.

Figure 2 shows all eigenvalues computed with Eispack. We have compared the results of Eispack with the results of the ordinary bi-Lanczos method (Fig. 3) and of the bilapo algorithm (Fig. 4). The bi-Lanczos method shows a significant loss of orthogonality, but the bilapo algorithm gives good approximations. A dot in the figures means a relative residual norm less than 10^{-4} and a plus a relative residual

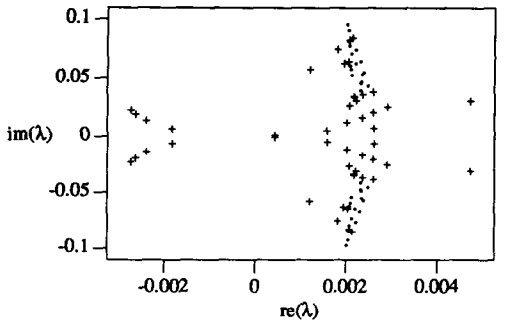

Fig. 3. Bi-Lanczos eigenvalues test 1a.

norm greater than 10^{-4}. These norms are computed with:

$$\text{relative residual norm} = \frac{\|\mathbf{Ax} - \lambda\mathbf{x}\|}{\|\lambda\mathbf{x}\|}, \tag{33}$$

where λ is an approximation of the eigenvalue of $\mathbf{A}$, and $\mathbf{x}$ an approximation of the corresponding eigenvector.

In Figs 5 and 6 we give a quantitative comparison of the relative residual norms after 90 iterations for the bi-Lanczos and bilapo method. As these graphs show, the results computed with the bilapo method are more accurate than those computed with bi-Lanczos.

To obtain these accurate results, the bilapo algorithm pauses six times for reorthogonalization. If one of the ωs exceeds the stop criterion $\sqrt{n\epsilon}$, full reorthogonalization takes place. This has occurred at the 35, 48, 61, 69, 78 and 86th iteration. Full reorthogonalization is necessary one iteration after these stops too. The time that elapses to compute 90 eigenvalues is shown in Table 1 and Fig. 7. We have compared these times with the elapsed time for an implementation of the simple Bi-Lanczos method and the method of Arnoldi.

Figure 7 shows clearly the quadratic growth of the CPU time used for the Arnoldi method because of the reorthogonalization of the iteration vectors at every step, and a linear growth in CPU time for the simple bi-Lanczos method. The bilapo method is still more expensive. However, the differences between the bi-

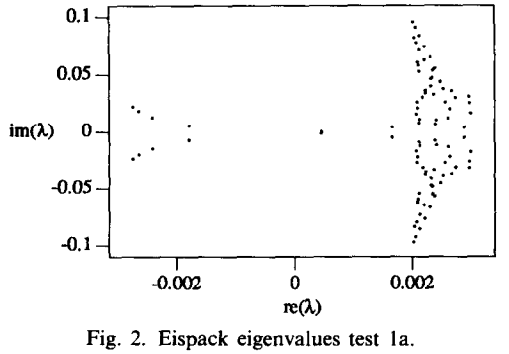

Fig. 2. Eispack eigenvalues test 1a.

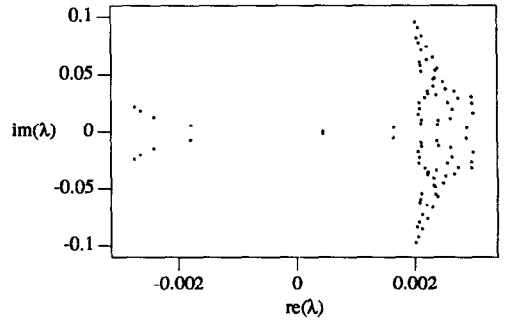

Fig. 4. Bilapo eigenvalues test 1a.

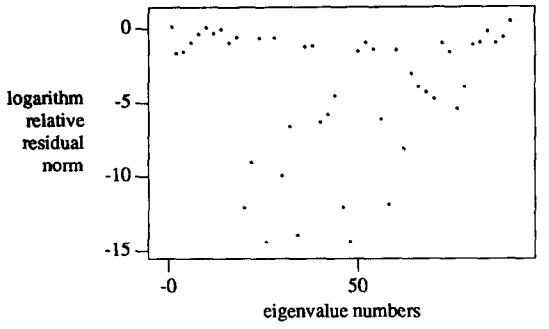

Fig. 5. Logarithm of the bi-Lanczos residuals test 1a.

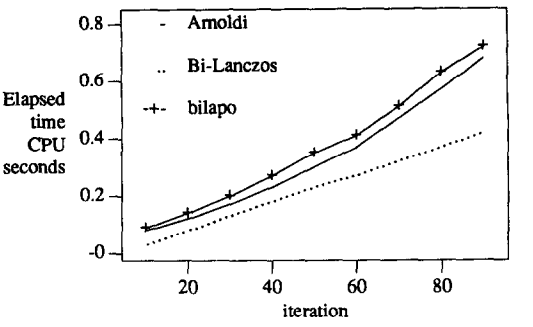

Fig. 7. Elapsed time test 1a; Arnoldi, bi-Lanczos and bilapo.

Lanczos method and Arnoldi is not very large for this small problem.

After this test we investigated the influence of the stop criterion on the number of stops made by bilapo and on the accuracy of the final results. Instead of $\sqrt{n\epsilon}$ we set the stop criterion on $\sqrt{\sqrt{n\epsilon}}$. Figures 8 and 9 show the eigenvalues and their accuracy after the iteration is completed. The accuracy of bilapo is still satisfactory for practical purposes. The bilapo algorithm stopped only four times, at iteration 44, 64, 77 and 89. The total elapsed time was now 0.63 CPU s. The bilapo method is now more attractive because it is less time consuming (Fig. 10).

4.2. *Non-associated plasticity*

A concentrated load is applied normal to the free surface of a semi-infinite body. Due to its axial symmetry, only a 2D mesh was constructed in the cross-section of the body (0.5 m horizontal × 0.4 m vertical, Fig. 11). This has resulted in a matrix of order 2698. The body is supported along 1–2 (displacement in y-direction $u_y = 0$), 2–3 ($u_x = 0$) and 4–1 ($u_x = 0$). The characteristic properties of this problem are:

Young's (elasticity) modulus E	0.96 N m^2
Poisson's ratio ν	0.20
cohesion	0.001 N
friction angle ϕ	20.0°
dilatancy angle ψ	0.0°.

The data are related according to the Drucker–Prager and Mohr–Coulomb yield criteria [2]. Furthermore, a vertical displacement of magnitude $u_y = -1.0$ mm is imposed on the top side, equally distributed over the middle portion of the area (in Fig. 11 over 0.05 m of the upper left corner). The displacement is imposed in small increments until a total amount of -1.0 mm results. The resulting deformation is plastic: it cannot be recovered by unloading the system and it is therefore permanent. Because of a difference in ϕ and ψ a non-associated plasticity deformation arises. This results in a non-symmetric tangent stiffness matrix.

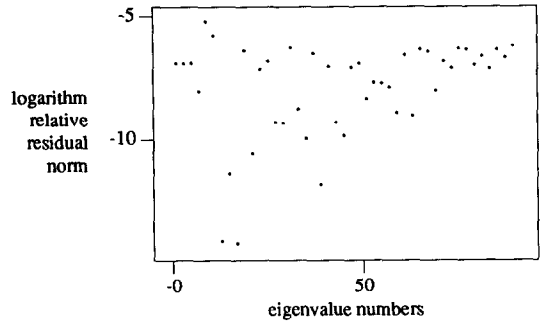

Fig. 6. Logarithm of the bilapo residuals test 1a.

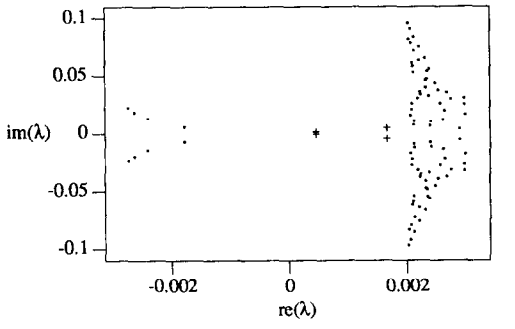

Fig. 8. Bilapo eigenvalues test 1b.

Table 1. CPU time used (s), test 1a

Method	Iterations 10	20	30	40	50	60	70	80	90
Bi-Lanczos	0.03	0.08	0.13	0.18	0.23	0.27	0.32	0.37	0.42
Arnoldi	0.08	0.12	0.17	0.23	0.30	0.37	0.47	0.57	0.68
Bilapo	0.09	0.14	0.20	0.27	0.35	0.41	0.51	0.63	0.72

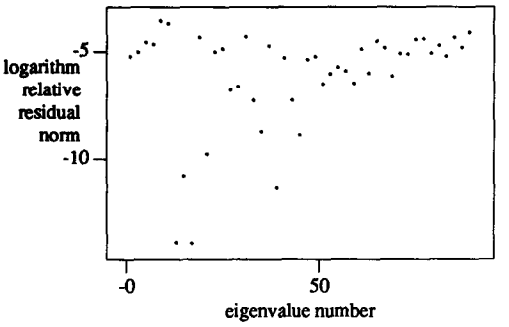

Fig. 9. Logarithm of the bilapo residuals test 1b.

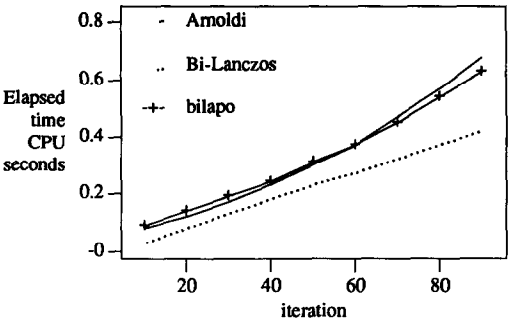

Fig. 10. Elapsed time with more loose stop criterion; Arnoldi, bi-Lanczos and bilapo.

The amount and moment of deformation must be supplied by a hardening/softening (cohesion increase/decrease) table. The data in this table are heuristic values, and must be adjusted until a realistic deformation graph results.

We have compared the results produced by Arnoldi and bilapo (Table 2) by computing the five smallest of $n = 2698$ eigenvalues. These eigenvalues are real. The first two columns of Table 2 show the eigenvalues computed by Arnoldi and their relative residual norms (33). "Bilapo 1" denotes the original algorithm with a stop (reorthogonalization) criterion of $\sqrt{n\epsilon}$ for the reorthogonalization of the iteration vectors. Again we have analyzed the consequences of

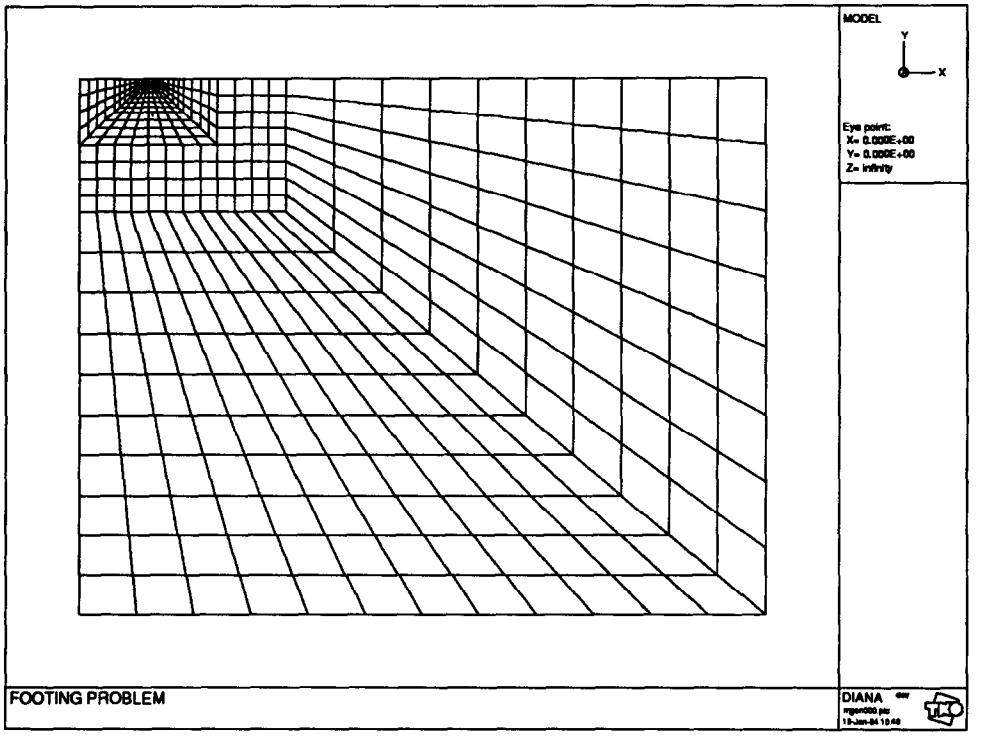

Fig. 11. Non-associated plasticity mesh.

Table 2. The five smallest eigenvalues and their norms

Arnoldi		bilapo 1	bilapo 2
eigenv	norm	eigenv	eigenv
0.1000×10^{-2}	0.4030×10^{-6}	0.1000×10^{-2}	0.1000×10^{-2}
0.2629×10^{-2}	0.9063×10^{-5}	0.2629×10^{-2}	0.2629×10^{-2}
0.3853×10^{-2}	0.1511×10^{-4}	0.3853×10^{-2}	0.3853×10^{-2}
0.6223×10^{-2}	0.8446×10^{-4}	0.6223×10^{-2}	0.6223×10^{-2}
0.8986×10^{-2}	0.2442×10^{-2}	0.8986×10^{-2}	0.8987×10^{-2}

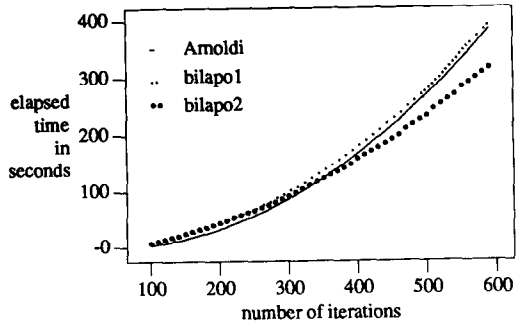

Fig. 12. Elapsed time for Arnoldi, and the two bilapo variations test 2.

a different reorthogonalization criterion of $\sqrt{\sqrt{n}\epsilon}$. The columns under "bilapo 2" show the eigenvalues computed by this criterion. The agreement with the method of Arnoldi is satisfactory for both bilapo methods.

Figure 12 shows the CPU time that it takes for the three methods to compute the five smallest eigenvalues. The original bilapo method requires about the same time as Arnoldi but the variation (bilapo2) is clearly less time consuming.

5. CONCLUSIONS AND DISCUSSION

We have shown in this paper that the newly developed bilapo method can be as attractive as the well-known Arnoldi method. At the start of the iteration it is slightly more time consuming because of an extra matrix–vector multiplication per iteration compared to the Arnoldi method. However, because the reorthogonalization of the iteration vectors is not performed every iteration, the bilapo method approaches the time required by Arnoldi at the end. If a more relaxed criterion is used the bilapo algorithm is even faster than Arnoldi.

The research on this method is still going on, and there are many aspects that should be investigated. A cheaper matrix–vector multiplication, for example, would speed up the bilapo algorithm more than the Arnoldi method. Tuning of the parameters like the reorthogonalization criterion could work to the advantage of the bilapo method as well, without a significant loss of accuracy. Furthermore, for the bilapo method it is not necessary to reorthogonalize against all iteration vectors (as is done in the present version) if loss of orthogonality has occurred. This loss may be caused only by a certain subset of the iteration vectors, so reorthogonalization is necessary only with respect to the vectors belonging to this subset.

For larger problems the time-cost of the reorthogonalization in Arnoldi will become a more serious drawback. The bilapo method could then become a very attractive alternative.

$v_0 = w_0 = 0$; $|v_1^T w_1| = 1$; $r_0 = v_1$; $p_0 = w_1$
$\beta_0 = 1$; $k = 0$
Do While ($\beta_k \neq 0$)
  $\gamma_k = r_k^T p_k / \beta_k$
  if ($\gamma_k \neq 0$) $v_{k+1} = r_k/\beta_k$; $w_{k+1} = p_k/\gamma_k$ *else breakdown*
  $k = k + 1$
  $\alpha_k = w_k^T A v_k$
  $r_k = A v_k - \alpha_k v_k - \gamma_{k-1} v_{k-1}$
  $p_k = A^T w_k - \alpha_k w_k - \beta_{k-1} w_{k-1}$
  $\beta_k = \|r_k\|$
End While

Fig. 13. The bi-Lanczos method [5, p. 503] (code 1).

$v_0 = w_0 = 0$; $|v_1^T w_1| = 1$; $r_0 = v_1$; $p_0 = w_1$
$\beta_0 = 1$; $k = 0$; *stopped* = *.false.*
Do While ($\beta_k \neq 0$)
  if *stopped then*
    $\tilde{p}_k = p_k - \Sigma_{i=1}^{k} (v_i^T p_k) p_i$ and $\tilde{r}_k = r_k - \Sigma_{i=1}^{k} (w_i^T r_k) r_i$
    $\beta_k = \|r_k\|$; if ($\beta_k = 0$) *breakdown*
    reset all ω*'s*; *stopped* = *.false.*
  end if
  if $k > 1$ *then*
    update ω*'s* (*see equations 23 , 24*)
    if *max.el.* (ω) > $\sqrt{n\epsilon}$ *then*
      $\tilde{p}_k = p_k - \Sigma_{i=1}^{k} (v_i^T p_k) p_i$ and $\tilde{r}_k = r_k - \Sigma_{i=1}^{k} (w_i^T r_k) r_i$
      $\beta_k = \|r_k\|$; if ($\beta_k = 0$) *breakdown*
      stopped = *.true.*
    end if
  end if
  $\gamma_k = r_k^T p_k / \beta_k$
  if ($\gamma_k \neq 0$) $v_{k+1} = r_k/\beta_k$; $w_{k+1} = p_k/\gamma_k$ *else breakdown*
  $k = k + 1$
  $\alpha_k = w_k^T A v_k$
  $r_k = A v_k - \alpha_k v_k - \gamma_{k-1} v_{k-1}$
  $p_k = A^T w_k - \alpha_k w_k - \beta_{k-1} w_{k-1}$
  $\beta_k = \|r_k\|$
End While

Fig. 14. The bilapo method (code 2).

Acknowledgements—The authors would like to thank Pier Nauta and Martin van Gijzen for their practical and theoretical support.

REFERENCES

1. W. E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. appl. Math.* **9**, 17–29 (1951).
2. R. de Borst, Computational methods in non-linear solid mechanics. TU-Delft, Report 25–2–90–5–04 (1991).
3. F. Chatelin, *Spectral Approximation of Linear Operations*. Academic Press, New York (1983).
4. R. W. Freund, M. H. Gutknecht and N. M. Nachtigal, An implementation of the look-ahead Lanczos algorithm for non-hermitian matrices. *SIAM J. Sci. Comput.* **14**, 137–156 (1993).
5. G. H. Golub and C. F. van Loan, *Matrix Computations*, 2nd Edn. Johns Hopkins University Press, Baltimore, MD (1989).

6. T. J. R. Hughes, *The Finite Element Method (Linear Static and Dynamic Finite Element Anlysis)*. Prentice-Hall, Englewood Cliffs, NJ (1987).
7. C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. natl Bur. Stand.* **45,** 255–282 (1950).
8. C. C. Paige, Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Math. Appl.* **18,** 341–349 (1976).
9. B. N. Parlett and D. S. Scott, The Lanczos algorithm with selective orthogonalization. *Math. Comput.* **33,** 217–238 (1979).
10. Y. Saad, Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices *Lin. Alg. Appl.* **34,** 269–295 (1980).
11. D. S. Scott, Analysis of the symmetric Lanczos process. Ph.D. Thesis, Department of Mathematics, University of California, Berkeley, CA (1987).
12. H. D. Simon, The Lanczos Algorithm for Solving Symmetric Linear Systems. Report PAM-74, Center for Pure and Applied Mathematics, University of California, Berkeley, CA (1982).
13. H. D. Simon, The Lanczos algorithm with partial reorthogonalization. *Math. Comput.* **42,** 115–142 (1984).
14. H. I. van der Veen, Eigenproblem algorithms for large and sparse non-symmetric matrices. TNO-report 93-NM-R1260, Rijswijk (1993).
15. J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford (1965).