TRANSPOSE-FREE MULTIPLE LANCZOS AND ITS APPLICATION IN PADÉ APPROXIMATION

MAN-CHUNG YEUNG* AND DANIEL BOLEY†

Abstract. A transpose-free two-sided nonsymmetric Lanczos method is developed for multiple starting vectors on both the left and right. The method is mathematically equivalent to the two-sided methods without look-ahead or deflation steps, but avoids the use of the transpose of the system matrix. The method is applied to the computation of the matrix Padé approximation to a linear dynamical system. The result is a method which can be labeled Transpose-Free Matrix Padé Via Lanczos (TFMPVL). Under certain circumstances, TFMPVL will actually reduce the total number of matrix-vector products needed. It is illustrated with some numerical examples.

Key words. transpose-free Lanczos method, model reduction, Padé approximation, MPVL method, TFMPVL method.

1. Introduction. In [1], Aliaga *et al.* proposed a Lanczos-type method that extends the classical Lanczos process for single starting vectors to multiple starting vectors. For convenience, we will refer to this Lanczos method as a *Two-Sided Lanczos procedure*. Given a square matrix $\mathbf{A} \in \mathcal{C}^{N\times N}$ and two blocks $\widehat{\mathbf{U}} \in \mathcal{C}^{N\times n}$ and $\widehat{\mathbf{V}} \in \mathcal{C}^{N\times m}$ of left and right starting vectors, the Two-Sided Lanczos algorithm employs matrix-vector products involving both $\mathbf{A}$ and $\mathbf{A}^H$, and generates (i) two sequences of biorthogonal basis vectors ("Krylov sequences") for the left and right block Krylov subspaces induced by the given data; (ii) band matrices $\bar{\mathbf{H}}$ and $\tilde{\mathbf{H}}$ that respectively constitute oblique projections of the matrices $\mathbf{A}$ and $\mathbf{A}^H$ onto the Krylov spaces. The remarkable feature of the algorithm is that, with a built-in deflation procedure and employing the look-ahead technique, it can handle the most general case of left and right block Krylov subspaces with arbitrary sizes of the starting blocks, while all previously proposed multiple starting Lanczos procedures are restricted to left and right starting blocks of identical sizes.

The Two-Sided Lanczos procedure has been used in applications in a variety of areas. For example, it is useful in the solution of linear systems with multiple right-hand sides and in the computation of approximate eigenvalues of a large matrix $\mathbf{A} \in \mathcal{C}^{N\times N}$. In the area of multi-input multi-output (MIMO) time-invariant linear dynamical systems, the Matrix Padé Via Lanczos (MPVL) method (see [12, 14], for instance) has been proposed to compute Padé approximation of transfer functions of the form

$$\mathbf{F}(\theta) = \widehat{\mathbf{U}}^H(\mathbf{I} - \theta\mathbf{A})^{-1}\widehat{\mathbf{V}} \tag{1.1}$$

in a stable manner using the Two-Sided Lanczos algorithm, where $\mathbf{A}, \mathbf{I} \in \mathcal{C}^{N\times N}$, $\widehat{\mathbf{U}} \in \mathcal{C}^{N\times n}$ and $\widehat{\mathbf{V}} \in \mathcal{C}^{N\times m}$.

In this paper, we propose a transpose-free version of the Two-Sided Lanczos procedure which computes the band matrix $\bar{\mathbf{H}}$ without accessing $\mathbf{A}^H$. A transpose-free version can be useful in certain situations involving sparse matrices, for which the cost of computing the matrix-vector product can be very different from the cost of

*Dept. 3036, 1000 East University Avenue, Laramie, WY 82071. E-mail: myeung@uwyo.edu. This research was supported by A & S Basic Research Grants during the 2001/02 academic year, University of Wyoming, and in part by NSF grant DMS-0314152.

†Computer Science and Engineering Department, University of Minnesota, Minneapolis, MN 55455. E-mail: boley@cs.umn.edu. This research was supported in part by NSF grant 0208621.

computing the vector-matrix product, depending on the storage format, particularly when attempting to parallelize the process [24, §3.5 & §11.5]. It also can be useful if the operator is not represented by an explicit matrix $\mathbf{A}$, but rather by a procedure derived from a differential operator which yields matrix-vector products but not vector-matrix products.

It is well known that the classical Lanczos process is intimately related to biconjugate gradient (BiCG) method for solving nonsymmetric systems of linear equations [19, 24]. Transpose-free versions, e.g., CGS [26], BiCGSTAB [28] for single starting vectors and ML(k)BiCGSTAB [29] for multiple left starting vectors, are methods derived from BiCG which avoid the need for matrix-vector products involving $\mathbf{A}^H$. In this paper, we extend techniques of avoiding matrix-vector multiplies with $\mathbf{A}^H$ in the context of solving systems of linear equations to handle the current case of multiple starting vectors on both the left and right and obtain a method which we label Transpose-free Multiple Lanczos Procedure (TFMLP). In our discussion, we assume for simplicity that no deflation or look-ahead steps occur in the Two-Sided Lanczos procedure, so TFMLP is actually a transpose-free version of the limited Two-Sided method. Moreover, we will point out how a variation of TFMPVL is closely related to the band Arnoldi procedure [17] when $n = N$.

To illustrate the method, we use the particular application of computing Padé approximations of transfer functions for Multi-Input-Multi-Output (MIMO) time-invariant linear dynamical systems (1.1). In this application, the ordinary Two-Sided Lanczos procedure leads to the so-called "Matrix Padé Via Lanczos" (MPVL) method [12, 14]. Correspondingly, a transpose-free MPVL method (TFMPVL) can be developed from the TFMLP method. We will see that the Transpose-Free version not only avoids computation with the adjoint operator, but can also reduce the total number of matrix-vector products, in the case when the number of output (left) vectors exceeds the number of input (right) vectors. Though transpose-free algorithms in the context of linear systems of equations are often less stable than the two-sided algorithm counterparts, we can stablize the method by introducing extra starting vectors in the spirit of ML(k)BiCGSTAB.

The close relationship between the Lanczos process, Padé approximants, moment matching, Asymptotic Waveform Evaluation, and Hankel system of equations has been explored extensively in the literature, see e.g. [8, 15, 18] and the recent survey [17]. We will give some specifics of this connection relevant to this paper in §5 after we have introduced the TFMLP method.

The rest of this paper is organized as follows. In §2 we review the Two-Sided Lanczos algorithm and introduce our notation, in §3 we derive our transpose-free Two-Sided Lanczos procedure, in §4 we show that TFMLP is closely related to the band Arnoldi procedure, in §5 and §6 we show how to use this Lanczos procedure to compute the Padé approximant, and in §7 we illustrate the methods with some numerical experiments. We end the paper with a preliminary discussion of the issue of deflation in §8 and some conclusions in §9.

2. Lanczos Procedure for Multiple Starting Vectors. Aliaga, Boley, Freund and Hernández (ABFH) [1] recently developed a Two-Sided Lanczos-type procedure that handles multiple starting vectors. Let $\mathbf{A} \in \mathcal{C}^{N \times N}$ and let n left starting vectors $\widehat{\mathbf{u}}_1, \widehat{\mathbf{u}}_2, \cdots, \widehat{\mathbf{u}}_n \in \mathcal{C}^N$ and m right starting vectors $\widehat{\mathbf{v}}_1, \widehat{\mathbf{v}}_2, \cdots, \widehat{\mathbf{v}}_m \in \mathcal{C}^N$ be

# of left Krylov vector $\mathbf{p}_k$	k	1 2 3	4 5 6	7 8 9	10 11 12	$\cdots$
power of $\mathbf{A}^H$: *grade*	$g_n(k)$	0 0 0	1 1 1	2 2 2	3 3 3	$\cdots$
applied to left starting vector#	$r_n(k)$	1 2 3	1 2 3	1 2 3	1 2 3	$\cdots$

FIG. 2.1. *Simple illustration of the* grades *for the left vectors (2.2a).*

given. Define index functions

$$\begin{aligned} g_n(k) &= \lfloor (k-1)/n \rfloor\,, \qquad & r_n(k) &= k - n\, g_n(k)\,, \\ g_m(k) &= \lfloor (k-1)/m \rfloor\,, \qquad & r_m(k) &= k - m\, g_m(k)\,, \end{aligned} \tag{2.1}$$

where $k = 1, 2, \ldots$ and $\lfloor \cdot \rfloor$ rounds its argument to the nearest integer towards minus infinity. Note that

$$g_n(jn+i) = j \quad \text{and} \quad r_n(jn+i) = i$$

if we write $k = jn + i$ with $j \geq 0$ and $1 \leq i \leq n$. A similar property holds with the other two index functions.

Now, we set

$$\text{(a)} \quad \mathbf{p}_k = \left(\mathbf{A}^H\right)^{g_n(k)} \widehat{\mathbf{u}}_{r_n(k)}\,, \qquad \text{(b)} \quad \mathbf{q}_k = \mathbf{A}^{g_m(k)} \widehat{\mathbf{v}}_{r_m(k)}\,. \tag{2.2}$$

The index $g_n(k)$ and $g_m(k)$ are called the *grades* of $\mathbf{p}_k$ and $\mathbf{q}_k$ respectively. They are non-decreasing step functions of k. Fig. 2.1 illustrates the grade for the left vectors $\mathbf{p}_k$ with $n = 3$ left starting vectors.

The Two-Sided Lanczos procedure generates two sequences $\{\mathbf{u}_{k'}\}_{k'=1,2,\cdots}$ and $\{\mathbf{v}_k\}_{k=1,2,\cdots}$ of vectors such that

$$\begin{aligned} \mathbf{u}_{k'} &\in \mathcal{G}_{k'}\left(\mathbf{A}^H, \widehat{\mathbf{U}}\right) \quad \text{and} \quad \mathbf{u}_{k'} \perp \mathcal{G}_{k'-1}\left(\mathbf{A}, \widehat{\mathbf{V}}\right), \\ \mathbf{v}_k &\in \mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right) \quad \text{and} \quad \mathbf{v}_k \perp \mathcal{G}_{k-1}\left(\mathbf{A}^H, \widehat{\mathbf{U}}\right), \end{aligned} \tag{2.3}$$

where

$$\begin{aligned} \mathcal{G}_{k'}\left(\mathbf{A}^H, \widehat{\mathbf{U}}\right) &\stackrel{\text{def}}{=} \mathit{span}\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{k'}\} \\ \mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right) &\stackrel{\text{def}}{=} \mathit{span}\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_k\}. \end{aligned} \tag{2.4}$$

The subspaces $\mathcal{G}_{k'}\left(\mathbf{A}^H, \widehat{\mathbf{U}}\right)$ and $\mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right)$ are referred to as the left block Krylov subspace and the right block Krylov subspace, respectively.

In the following we present some theory regarding the existence of the vector sequences $\{\mathbf{u}_{k'}\}$, $\{\mathbf{v}_k\}$ based on [1, 12, 16], but using our own notation. The existence of two such sequences $\{\mathbf{u}_{k'}\}$ and $\{\mathbf{v}_k\}$ of vectors can be guaranteed if the following matrices

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{p}_1^H\mathbf{q}_1 & \mathbf{p}_1^H\mathbf{q}_2 & \cdots & \mathbf{p}_1^H\mathbf{q}_k \\ \mathbf{p}_2^H\mathbf{q}_1 & \mathbf{p}_2^H\mathbf{q}_2 & \cdots & \mathbf{p}_2^H\mathbf{q}_k \\ \vdots & \vdots & & \vdots \\ \mathbf{p}_k^H\mathbf{q}_1 & \mathbf{p}_k^H\mathbf{q}_2 & \cdots & \mathbf{p}_k^H\mathbf{q}_k \end{bmatrix}, \quad \text{for all } k = 1, 2, \ldots, \nu \tag{2.5}$$

are all nonsingular for some ν.

LEMMA 2.1. *If all the leading principal submatrices of $\mathbf{W}_\nu$ are nonsingular, then there exist two sets $\{\mathbf{u}_{k'}\}_{k'=1}^{\nu}$ and $\{\mathbf{v}_k\}_{k=1}^{\nu}$ of linearly independent vectors which satisfy properties (2.3). Moreover,*

$$span\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{k'}\} = \mathcal{G}_{k'}\left(\mathbf{A}^H, \widehat{\mathbf{U}}\right), \qquad span\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k\} = \mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right),$$

where $k', k = 1, 2, \ldots, \nu$.

Proof. In fact, if we express $\mathbf{v}_k$ as

$$\mathbf{v}_k = \gamma_1^{(k)}\mathbf{q}_1 + \gamma_2^{(k)}\mathbf{q}_2 + \cdots + \gamma_{k-1}^{(k)}\mathbf{q}_{k-1} + \mathbf{q}_k, \tag{2.6}$$

then (2.3) is equivalent to $\mathbf{W}_{k-1}\boldsymbol{\gamma}^{(k)} + \mathbf{b} = 0$ where $\boldsymbol{\gamma}^{(k)} = [\gamma_1^{(k)}, \gamma_2^{(k)}, \cdots, \gamma_{k-1}^{(k)}]^T$ and $\mathbf{b} = [\mathbf{p}_1^H\mathbf{q}_k, \mathbf{p}_2^H\mathbf{q}_k, \cdots, \mathbf{p}_{k-1}^H\mathbf{q}_k]^T$. Furthermore, the $\mathbf{v}_k$ defined by (2.6) satisfies $\mathbf{v}_k \not\perp \mathbf{p}_k$ for $k \le \nu$ and hence $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_\nu$ are linearly independent. The same arguments can be applied to the vectors $\mathbf{u}_{k'}$. ∎

From the definition of $\mathbf{q}_k$, we note that $\mathbf{q}_k = \mathbf{A}\mathbf{q}_{k-m}$ for $k > m$. Applying (2.6) to itself recursively, we can write $\mathbf{v}_k$ $(k > m)$ in terms of the previous $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{k-1}$ as follows,

$$\mathbf{v}_k = \mathbf{A}\mathbf{v}_{k-m} - \bar{h}_{k-1}^{(k-m)}\mathbf{v}_{k-1} - \bar{h}_{k-2}^{(k-m)}\mathbf{v}_{k-2} - \cdots - \bar{h}_1^{(k-m)}\mathbf{v}_1\,, \tag{2.7}$$

where $\bar{h}$'s are some scalars. A similar equation for vectors $\mathbf{u}_{k'}$ $(k' > n)$ is also available.

LEMMA 2.2. *The vectors $\mathbf{v}_k$ and $\mathbf{u}_{k'}$ in Lemma 2.1 with $k > m$ and $k' > n$ can be expressed with $m + n + 1$ term recursion relationships of the forms*

$$\mathbf{v}_k = \mathbf{A}\mathbf{v}_{k-m} - \bar{h}_{k-1}^{(k-m)}\mathbf{v}_{k-1} - \bar{h}_{k-2}^{(k-m)}\mathbf{v}_{k-2} - \cdots - \bar{h}_{\bar{m}_{k-m}}^{(k-m)}\mathbf{v}_{\bar{m}_{k-m}} \tag{2.8}$$

and

$$\mathbf{u}_{k'} = \mathbf{A}^H\mathbf{u}_{k'-n} - \tilde{h}_{k'-1}^{(k'-n)}\mathbf{u}_{k'-1} - \tilde{h}_{k'-2}^{(k'-n)}\mathbf{u}_{k'-2} - \cdots - \tilde{h}_{\tilde{m}_{k'-n}}^{(k'-n)}\mathbf{u}_{\tilde{m}_{k'-n}},$$

where $\bar{m}_i = \max(i-n, 1)$ and $\tilde{m}_i = \max(i-m, 1)$.

Proof. Noting that $\mathbf{v}_i \perp span\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{i-1}\}, \mathbf{v}_i \not\perp \mathbf{p}_i$ and $\mathbf{p}_i^H\mathbf{A}\mathbf{v}_{k-m} = \mathbf{p}_{i+n}^H\mathbf{v}_{k-m} = 0$ for $i \le k-m-n-1$, and examining in turn

$$\mathbf{p}_i^H\mathbf{v}_k = \mathbf{p}_i^H\mathbf{A}\mathbf{v}_{k-m} - \bar{h}_{k-1}^{(k-m)}\mathbf{p}_i^H\mathbf{v}_{k-1} - \bar{h}_{k-2}^{(k-m)}\mathbf{p}_i^H\mathbf{v}_{k-2} - \cdots - \bar{h}_1^{(k-m)}\mathbf{p}_i^H\mathbf{v}_1$$

for $i = 1, 2, \ldots, k-m-n-1$, we find all the coefficients in (2.7) zero except $\bar{h}_{k-1}^{(k-m)}, \bar{h}_{k-2}^{(k-m)}, \ldots, \bar{h}_{\bar{m}_{k-m}}^{(k-m)}$. ∎

Set $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k]$ and set $\bar{\mathbf{H}} = \left(\bar{h}_{ij}\right)_{i=1,\cdots,\nu;j=1,\cdots,\nu-m}$, the $\nu \times (\nu - m)$ band matrix with $\bar{h}_{j+m,j} = 1$; $\bar{h}_{ij} = \bar{h}_i^{(j)}$ for $\bar{m}_j \le i \le j + m - 1$; with $\bar{h}_{ij} = 0$ otherwise. Then the recurrence relations (2.8) can be written in matrix form as

$$\mathbf{A}\mathbf{V}_{\nu-m} = \mathbf{V}_\nu\bar{\mathbf{H}}\,. \tag{2.9}$$

Similar results can be drawn for the vectors $\mathbf{u}_{k'}$. We collect these results into the following.

THEOREM 2.3. *Let vectors $\widehat{\mathbf{u}}_1, \widehat{\mathbf{u}}_2, \cdots, \widehat{\mathbf{u}}_n, \widehat{\mathbf{v}}_1, \widehat{\mathbf{v}}_2, \cdots, \widehat{\mathbf{v}}_m$ be given starting vectors and let $\mathbf{p}_k$'s, $\mathbf{q}_k$'s, $\mathcal{G}_{k'}\left(\mathbf{A}^H, \widehat{\mathbf{U}}\right)$, $\mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right)$ and $\mathbf{W}_\nu$ be defined as in (2.2), (2.4),*

(2.5). If all the leading principal submatrices of $\mathbf{W}_\nu$ *are nonsingular, then there exist two sets* $\{\mathbf{u}_{k'}\}_{k'=1}^{\nu}$ *and* $\{\mathbf{v}_k\}_{k=1}^{\nu}$ *of linearly independent vectors, an* $\nu \times (\nu - m)$ *band matrix* $\bar{\mathbf{H}}$ *whose upper bandwidth is* n*, lower bandwidth is* m *and in which all the entries in its lowest diagonal are equal to 1 (*$\bar{h}_{j+m,j} = 1$*), and an* $\nu \times (\nu - n)$ *band matrix* $\widetilde{\mathbf{H}}$*, whose upper bandwidth is* m*, lower bandwidth is* n *and in which all the entries in its lowest diagonal are equal to 1 (*$\tilde{h}_{j+n,j} = 1$*), such that*

$$(2.10) \quad span\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{k'}\} = \mathcal{G}_{k'}\left(\mathbf{A}^H, \widehat{\mathbf{U}}\right), \quad span\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k\} = \mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right)$$

and

$$\mathbf{u}_{k'}^H \mathbf{v}_k \begin{cases} \neq 0 & \text{if } k' = k \\ = 0 & \text{if } k' \neq k \end{cases}$$

for all $k', k = 1, 2, \ldots, \nu$*. Moreover,*

$$\mathbf{A}^H \mathbf{U}_{\nu-n} = \mathbf{U}_\nu \widetilde{\mathbf{H}} \quad \text{and} \quad \mathbf{A}\mathbf{V}_{\nu-m} = \mathbf{V}_\nu \bar{\mathbf{H}},$$

where $\mathbf{U}_{k'} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{k'}]$ *and* $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k]$.

The Two-Sided Lanczos procedure of [1] is a procedure which computes the quantities $\mathbf{U}_\nu, \mathbf{V}_\nu, \widetilde{\mathbf{H}}$ and $\bar{\mathbf{H}}$ in Theorem 2.3. The procedure of [1] also handles the cases where some $\mathbf{W}_k$ is singular with a complete deflation and look-ahread process, but in our transpose-free algorithm, we will postpone discussion of such complications to a later section.

3. Transpose-free Lanczos Procedure for Multiple Starting Vectors. The implementation of the Two-Sided Lanczos procedure involves matrix-vector multiplications with $\mathbf{A}^H$. As mentioned in §1, a number of articles in the literature have discussed Lanczos implementations without accessing $\mathbf{A}^H$, see, for instance, [10, 13, 19, 20, 21, 24, 25, 26, 28, 29], mostly in the context of solving systems of linear equations. Techniques of avoiding matrix-vector multiplies with $\mathbf{A}^H$ in the classical Lanczos procedure can be generalized to the current case. In this section, we will give a new variant of a "limited" Two-Sided Lanczos procedure which computes the $\bar{\mathbf{H}}$ in Theorem 2.3 without using $\mathbf{A}^H$. To simplify the derivation, we will suppose the assumption of Theorem 2.3 holds so that the deflation or look-ahead features in the full Two-Sided algorithm are not needed.

We continue to use the notation introduced in §2. Also, we assume $m \leq n$ in the following derivation. Now, we consider the equation (2.8),

$$\mathbf{v}_{k+m} = \mathbf{A}\mathbf{v}_k - \bar{h}_{\bar{m}_k}^{(k)} \mathbf{v}_{\bar{m}_k} - \bar{h}_{\bar{m}_k+1}^{(k)} \mathbf{v}_{\bar{m}_k+1} - \cdots - \bar{h}_{k+m-1}^{(k)} \mathbf{v}_{k+m-1},$$

where $k = 1, 2, \cdots, \nu - m$. Because of the property (2.3), the coefficients $\bar{h}_i^{(k)}$ are determined by

$$\bar{h}_i^{(k)} = \frac{\mathbf{p}_i^H \mathbf{A}\mathbf{v}_k - \mathbf{p}_i^H \sum_{j=\bar{m}_k}^{i-1} \bar{h}_j^{(k)} \mathbf{v}_j}{\mathbf{p}_i^H \mathbf{v}_i}, \qquad i = \bar{m}_k, \bar{m}_k + 1, \ldots, k + m - 1.$$

Thus, we have the following naive procedure to compute the right Lanczos vectors $\{\mathbf{v}_k\}$ and matrix $\bar{\mathbf{H}}$, while computing only the left Krylov vectors $\{\mathbf{p}_k\}$ instead of the left Lanczos vectors $\{\mathbf{u}_k\}$.

Lanczos Procedure Version 1.

1. Compute vectors $\{\mathbf{v}_k\}_{k=1}^m$ such that $\mathbf{v}_k \perp span\{\widehat{\mathbf{u}}_1, \cdots, \widehat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\widehat{\mathbf{v}}_1, \cdots, \widehat{\mathbf{v}}_k\}$, and compute $\mathbf{p}_k = (\mathbf{A}^H)^{g_n(k)}\widehat{\mathbf{u}}_{r_n(k)}$, $k = 1, \ldots, m$, according to (2.2a).
2. For $k = 1, 2, \cdots, \nu - m$
3. $\quad \bar{m}_k = max\{k - n, 1\}$;
4. $\quad$ For $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
5. $\quad\quad \bar{h}_i^{(k)} = \dfrac{\mathbf{p}_i^H \mathbf{A}\mathbf{v}_k - \mathbf{p}_i^H \sum_{j=\bar{m}_k}^{i-1} \bar{h}_j^{(k)} \mathbf{v}_j}{\mathbf{p}_i^H \mathbf{v}_i}$;
6. $\quad$ End
7. $\quad \bar{h}_{k+m}^{(k)} = 1$;
8. $\quad \mathbf{v}_{k+m} = \mathbf{A}\mathbf{v}_k - \sum_{i=\bar{m}_k}^{k+m-1} \bar{h}_i^{(k)} \mathbf{v}_i$;
9. $\quad$ Compute $\mathbf{p}_{k+m} = (\mathbf{A}^H)^{g_n(k+m)}\widehat{\mathbf{u}}_{r_n(k+m)}$ according to (2.2a).
10. End

It is obvious that Version 1 can be simplified by noting that the summations in the numerators of Line 5 are just partial sums of the summation in Line 8. So, we can accumulate the summations one term at a time into a temporary vector $\mathbf{v}_{tmp}$, and use the partial sums stored in $\mathbf{v}_{tmp}$ directly in Lines 5 and 8 as they are generated. This avoids effectively having to accumulate the summations multiple times and also eliminates one redundant matrix-vector product $\mathbf{A}\mathbf{v}_k$. Since $m \leq n$, we have that $g_n(k) = 0$ and $r_n(k) = k$ during the initial stage of processing the starting vectors when $1 \leq k \leq m$. We also introduce a scalar variable c_k defined by $c_k = \mathbf{p}_k^H \mathbf{v}_k$ to save repeatly computing the inner product $\mathbf{p}_i^H \mathbf{v}_i$. Thus, we arrive at the following version.

Lanczos Procedure Version 2.

1. Compute vectors $\{\mathbf{v}_k\}_{k=1}^m$ such that $\mathbf{v}_k \perp span\{\widehat{\mathbf{u}}_1, \cdots, \widehat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\widehat{\mathbf{v}}_1, \cdots, \widehat{\mathbf{v}}_k\}$.
2. Compute $\mathbf{p}_k = \widehat{\mathbf{u}}_k$ for $k = 1, \ldots, m$.
3. Compute $c_k = \mathbf{p}_k^H \mathbf{v}_k$ for $k = 1, \ldots, m$.
4. For $k = 1, 2, \cdots, \nu - m$
5. $\quad \bar{m}_k = max\{k - n, 1\}$;
6. $\quad \mathbf{v}_{tmp} = \mathbf{A}\mathbf{v}_k$;
7. $\quad$ For $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
8. $\quad\quad \bar{h}_i^{(k)} = \mathbf{p}_i^H \mathbf{v}_{tmp}/c_i$;
9. $\quad\quad \mathbf{v}_{tmp} = \mathbf{v}_{tmp} - \bar{h}_i^{(k)} \mathbf{v}_i$;
10. $\quad$ End
11. $\quad \bar{h}_{k+m}^{(k)} = 1$;
12. $\quad \mathbf{v}_{k+m} = \mathbf{v}_{tmp}$;
13. $\quad \mathbf{p}_{k+m} = (\mathbf{A}^H)^{g_n(k+m)}\widehat{\mathbf{u}}_{r_n(k+m)}$;
14. $\quad c_{k+m} = \mathbf{p}_{k+m}^H \mathbf{v}_{k+m}$;
15. End

Based on the above Lanczos version, we are now ready to give a transpose-free procedure to compute $\bar{\mathbf{H}}$. In order to remove the $\mathbf{A}^H$ which is used to calculate $\mathbf{p}_{k+m}$ in Line 13, we introduce an auxiliary vector $\boldsymbol{\psi}_k$ defined by

$$\boldsymbol{\psi}_k = \mathbf{A}^{g_n(k)} \mathbf{v}_k \tag{3.1}$$

for $k = 1, 2, \ldots$. A note of terminology: $\boldsymbol{\psi}_k$ represents the k-Krylov vector indirectly, in that it has an extra factor of $\mathbf{A}^{g_n(k)}$. We will say that the *grade* of $\boldsymbol{\psi}_k$ is $g_n(k)$,

and is set to match the grade of the corresponding *left* vector $\mathbf{p}_k$.

By combining (3.1) with (2.2a), we can fold powers of $\mathbf{A}$ from the left vectors into the right vectors:

$$\mathbf{p}_k^H \mathbf{v}_k = \left(\widehat{\mathbf{u}}_{r_n(k)}^H \mathbf{A}^{g_n(k)}\right) \mathbf{v}_k = \widehat{\mathbf{u}}_{r_n(k)}^H \left(\mathbf{A}^{g_n(k)} \mathbf{v}_k\right) = \widehat{\mathbf{u}}_{r_n(k)}^H \boldsymbol{\psi}_k. \tag{3.2}$$

The inner product in line 14 can be written as an inner product of $\widehat{\mathbf{u}}_{r_n(k+m)}$ and $\boldsymbol{\psi}_{k+m}$. Likewise, the inner product in line 8 can be written as the inner product between $\widehat{\mathbf{u}}_{r_n(i)}$ and a temporary vector $\boldsymbol{\psi}_{tmp}$, but the grade of $\boldsymbol{\psi}_{tmp}$ must be adjusted to match that of the corresponding vector $\mathbf{p}_i$. That is, we must have that $\boldsymbol{\psi}_{tmp} = \mathbf{A}^{g_n(i)} \mathbf{v}_{tmp}$, to match the condition

$$\mathbf{p}_i^H \mathbf{v}_{tmp} = \left(\widehat{\mathbf{u}}_{r_n(i)}^H \mathbf{A}^{g_n(i)}\right) \mathbf{v}_{tmp} = \widehat{\mathbf{u}}_{r_n(i)}^H \left(\mathbf{A}^{g_n(i)} \mathbf{v}_{tmp}\right) = \widehat{\mathbf{u}}_{r_n(i)}^H \boldsymbol{\psi}_{tmp}.$$

Assuming we can keep the grade of $\boldsymbol{\psi}_{tmp}$ adjusted correctly, we can eliminate any explicit reference to the left Krylov vectors $\{\mathbf{p}_k\}$ or the transposed operator $\mathbf{A}^H$. Hence we can eliminate lines 2 and 13 where the left Krylov vectors are computed.

It remains to show how $\boldsymbol{\psi}_k$ can be recursively computed from the previous $\boldsymbol{\psi}$'s, and how the grade of the temporary vector $\boldsymbol{\psi}_{tmp}$ can be adjusted on the fly. Our goal is to compute $\bar{\mathbf{H}}$ and the right vectors $\boldsymbol{\psi}$ and optionally the vectors $\mathbf{v}$.

To keep track of the grade of the temporary vector $\boldsymbol{\psi}_{tmp}$, we introduce a variable $grade = g_n(\bar{m}_k)$. Examining Version 2, within each pass through the outer k–loop (lines 4–15), we initialize

$$\boldsymbol{\psi}_{tmp} = \mathbf{A}^{g_n(k)-1} \mathbf{v}_{tmp} = \mathbf{A}^{g_n(k)} \mathbf{v}_k = \boldsymbol{\psi}_k. \tag{3.3}$$

When we first enter the inner i–loop (lines 7–10), the grade of $\boldsymbol{\psi}_{tmp}$ must match the that of the first $\mathbf{p}$ vector we apply in line 8, namely $\mathbf{p}_{\bar{m}_k}$. But $k - n \le \bar{m}_k \le k$ by the definition in line 5, hence $g_n(k) - 1 \le g_n(\bar{m}_k) \le g_n(k)$. Hence the initial grade of $\boldsymbol{\psi}_{tmp}$ is either correct or must be incremented by 1. As we progress through the i–loop, we access the left vectors in increasing order of grade, so occasionally we will have to increment the grade again. If we started with n left vectors, then this grade would have to be incremented only once every n passes through the i–loop, and if $n \ge m$, this incrementing happens at least once, but no more than two times. So we add code to the i–loop to increment the grade of $\boldsymbol{\psi}_{tmp}$ when it is necessary.

At the end of each pass through the k–loop, we generate a new vector $\boldsymbol{\psi}_{k+m} (= \boldsymbol{\psi}_{tmp})$ in line 12. The grade of $\boldsymbol{\psi}_{tmp}$ at the end of the pass through the k–loop is the grade it has when we leave the inner i–loop, namely $g_n(k+m-1)$. Hence we need to add code to possibly increment the grade one last time so that the grade of the stored vector $\boldsymbol{\psi}_{k+m}$ is $g_n(k+m)$. Since the vector $\boldsymbol{\psi}_{k+m}$ is computed at the end of the k–loop, we can use the space to be occupied by $\boldsymbol{\psi}_{k+m}$ to store the value of $\boldsymbol{\psi}_{tmp}$.

Combining all these manipulations yields the Transpose-free Multiple Lanczos Procedure (TFMLP).

ALGORITHM 3.1. **Transpose-free Multiple Lanczos Procedure (TFMLP)** *Given m right starting vectors $\{\widehat{\mathbf{v}}_k\}_{k=1}^m$ and n left starting vectors $\{\widehat{\mathbf{u}}_{k'}\}_{k'=1}^n$ with $m \le n$. Suppose the assumption of Theorem 2.3 holds. The following algorithm computes the matrix $\bar{\mathbf{H}} = \{\bar{h}_{ij}\}$ (and optionally $\mathbf{V}_\nu$) in Theorem 2.3 where $\bar{h}_{ij} = \bar{h}_i^{(j)}$.*

1. *Compute vectors $\{\mathbf{v}_k\}_{k=1}^m$ such that $\mathbf{v}_k \perp span\{\widehat{\mathbf{u}}_1, \cdots, \widehat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\widehat{\mathbf{v}}_1, \cdots, \widehat{\mathbf{v}}_k\}$.*
2. *For $k = 1, 2, \ldots, m$, do: set $\boldsymbol{\psi}_k = \mathbf{v}_k$ and compute $c_k = \widehat{\mathbf{u}}_k^H \mathbf{v}_k$.*

3. For $k = 1, 2, 3, \cdots$
4. $\quad \bar{m}_k = max\{k - n, 1\}$;
5. $\quad grade = g_n(\bar{m}_k)$;
6. $\quad \boldsymbol{\psi}_{k+m} = \mathbf{A}^{1+grade-g_n(k)}\boldsymbol{\psi}_k$ *(and optionally* $\mathbf{v}_{k+m} = \mathbf{A}\mathbf{v}_k$*);*
7. $\quad$ *For* $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
8. $\quad\quad$ *If* $g_n(i) > grade$
9. $\quad\quad\quad \boldsymbol{\psi}_{k+m} = \mathbf{A}\boldsymbol{\psi}_{k+m}$;
10. $\quad\quad\quad grade = grade + 1$;
11. $\quad\quad$ *End*
12. $\quad\quad \bar{h}_i^{(k)} = \widehat{\mathbf{u}}_{r_n(i)}^H \boldsymbol{\psi}_{k+m} \,/\, c_i$;
13. $\quad\quad \boldsymbol{\psi}_{k+m} = \boldsymbol{\psi}_{k+m} - \bar{h}_i^{(k)}\boldsymbol{\psi}_i$ *(and optionally* $\mathbf{v}_{k+m} = \mathbf{v}_{k+m} - \bar{h}_i^{(k)}\mathbf{v}_i$*);*
14. $\quad$ *End*
15. $\quad \bar{h}_{k+m}^{(k)} = 1$;
16. $\quad$ *If* $g_n(k + m) > grade$
17. $\quad\quad \boldsymbol{\psi}_{k+m} = \mathbf{A}\boldsymbol{\psi}_{k+m}$;
18. $\quad$ *End*
19. $\quad c_{k+m} = \widehat{\mathbf{u}}_{r_n(k+m)}^H \boldsymbol{\psi}_{k+m}$;
20.End

Note that in the above algorithm, we have expressed all the quantities necessary to carry out the iteration, namely the $\bar{h}$'s, $\boldsymbol{\psi}$'s, without using the $\mathbf{v}$ vectors. The $\mathbf{v}$ vectors are needed only to define the new $\mathbf{v}$ vectors, and hence need to be computed only if the user needs them explicitly. In addition, we note the extra code needed to adjust the grades of $\boldsymbol{\psi}_{tmp}$ (actually stored in $\boldsymbol{\psi}_{k+m}$) appears in lines 8–11 and 16–18.

It is often the case in practice that the norms $\|\boldsymbol{\psi}_k\|_2$ become very large or very small as Algorithm 3.1 progresses, and as a result, the matrix $\bar{\mathbf{H}}$ obtained by the algorithm can become very close to singular. So, it is necessary either to normalize the vectors $\boldsymbol{\psi}_k$ or to balance $\bar{\mathbf{H}}$ in order to make the algorithm more practicable.

For that purpose, let $\boldsymbol{\Lambda}_\nu = diag\{\lambda_1, \lambda_2, \cdots, \lambda_\nu\}$ be a nonsingular diagonal matrix and let

$$\widehat{\mathbf{H}} = \boldsymbol{\Lambda}_\nu \, \bar{\mathbf{H}} \, \boldsymbol{\Lambda}_{\nu-m}^{-1} \qquad \text{and} \qquad \widetilde{\mathbf{V}}_\nu = \mathbf{V}_\nu \boldsymbol{\Lambda}_\nu^{-1}. \tag{3.4}$$

Because of equation (2.9), $\widehat{\mathbf{H}}$ and $\widetilde{\mathbf{V}}_\nu$ are related by

$$\mathbf{A}\widetilde{\mathbf{V}}_{\nu-m} = \widetilde{\mathbf{V}}_\nu \widehat{\mathbf{H}} \, .$$

We now modify Algorithm 3.1 to an algorithm which directly computes the matrix $\widehat{\mathbf{H}}$ (and optionally $\widetilde{\mathbf{V}}_\nu$). To do so, we redefine the variables $c_k, \bar{h}_i^{(k)}$ and $\boldsymbol{\psi}_k$ in Algorithm 3.1 as follows

$$\widehat{h}_i^{(k)} \stackrel{\text{def}}{=} \lambda_i \bar{h}_i^{(k)} \lambda_k^{-1}; \;\; \boldsymbol{\phi}_k \stackrel{\text{def}}{=} \boldsymbol{\psi}_k / \lambda_k; \;\; b_k \stackrel{\text{def}}{=} c_k / \lambda_k; \;\; (\text{optionally } \tilde{\mathbf{v}}_k \stackrel{\text{def}}{=} \mathbf{v}_k / \lambda_k). \tag{3.5}$$

With these new definitions, Algorithm 3.1 becomes

ALGORITHM 3.2. **Scaled Transpose-free Multiple Lanczos Procedure (Scaled TFMLP)** *Given m right starting vectors $\{\widehat{\mathbf{v}}_k\}_{k=1}^m$ and n left starting vectors $\{\widehat{\mathbf{u}}_{k'}\}_{k'=1}^n$ with $m \le n$. Suppose the assumption of Theorem 2.3 holds. The following algorithm computes the $\nu \times (\nu - m)$ band matrix $\widehat{\mathbf{H}} = (\widehat{h}_{ij})$ with $\widehat{h}_{ij} = \widehat{h}_i^{(j)}$ (and optionally the matrix $\widetilde{V}_\nu$) described in equation (3.4).*

1. *Compute vectors $\{\mathbf{v}_k\}_{k=1}^m$ such that $\mathbf{v}_k \perp span\{\widehat{\mathbf{u}}_1, \cdots, \widehat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\widehat{\mathbf{v}}_1, \cdots, \widehat{\mathbf{v}}_k\}$.*

2. For $k = 1, 2, \ldots, m$, do: define λ_k, set $\boldsymbol{\phi}_k = \mathbf{v}_k \,/\, \lambda_k$
(and optionally $\tilde{\mathbf{v}}_k = \boldsymbol{\phi}_k$), and compute $b_k = \widehat{\mathbf{u}}_k^H \boldsymbol{\phi}_k$.
3. For $k = 1, 2, 3, \ldots$
4. $\bar{m}_k = max\{k - n, 1\}$;
5. $grade = g_n(\bar{m}_k)$;
6. $\boldsymbol{\phi}_{k+m} = \mathbf{A}^{1+grade-g_n(k)} \boldsymbol{\phi}_k$ *(and optionally* $\tilde{\mathbf{v}}_{k+m} = \mathbf{A}\tilde{\mathbf{v}}_k$*);*
7. For $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
8. If $g_n(i) > grade$
9. $\boldsymbol{\phi}_{k+m} = \mathbf{A}\boldsymbol{\phi}_{k+m}$;
10. $grade = grade + 1$;
11. End
12. $\widehat{h}_i^{(k)} = \widehat{\mathbf{u}}_{r_n(i)}^H \boldsymbol{\phi}_{k+m} \,/\, b_i$;
13. $\boldsymbol{\phi}_{k+m} = \boldsymbol{\phi}_{k+m} - \widehat{h}_i^{(k)} \boldsymbol{\phi}_i$ *(and optionally* $\tilde{\mathbf{v}}_{k+m} = \tilde{\mathbf{v}}_{k+m} - \bar{h}_i^{(k)} \tilde{\mathbf{v}}_i$*);*
14. End
15. If $g_n(k + m) > grade$
16. $\boldsymbol{\phi}_{k+m} = \mathbf{A}\boldsymbol{\phi}_{k+m}$;
17. End
18. Define $\widehat{h}_{k+m}^{(k)}$;
19. $\boldsymbol{\phi}_{k+m} = \boldsymbol{\phi}_{k+m} \,/\, \widehat{h}_{k+m}^{(k)}$; *(and optionally* $\tilde{\mathbf{v}}_{k+m} = \tilde{\mathbf{v}}_{k+m} \,/\, \widehat{h}_{k+m}^{(k)}$*);*
20. $b_{k+m} = \widehat{\mathbf{u}}_{r_n(k+m)}^H \boldsymbol{\phi}_{k+m}$;
21.End

We remark that the λ's and $\widehat{h}$'s in Lines 2 and 18 of Algorithm 3.2 can be assigned any nonzero numbers; A typical choice is to set $\lambda_k = \|\mathbf{v}_k\|_2$ in line 2 and $\widehat{h}_{k+m}^{(k)} = \|\boldsymbol{\phi}_{k+m}\|_2$ in line 18. This choice is equivalent to scaling the vectors $\boldsymbol{\phi}_k$ to unit length.

We also remark that Lines 4 - 20 compute the entries of $\widehat{\mathbf{H}}$ in the k-th column. The k-th column of $\widehat{\mathbf{H}}$ is related to the k-th column of $\bar{\mathbf{H}}$ by

$$\begin{aligned}
&[0, \cdots, 0, \widehat{h}_{k-n}^{(k)}, \cdots, \widehat{h}_{k+m}^{(k)}, 0, \cdots, 0]^T \\
&\qquad = \quad [0, \cdots, 0, \lambda_{k-n}\bar{h}_{k-n}^{(k)}\lambda_k^{-1}, \cdots, \lambda_{k+m}\bar{h}_{k+m}^{(k)}\lambda_k^{-1}, 0, \cdots, 0]^T
\end{aligned}$$

according to (3.4), where $\bar{h}_{k+m}^{(k)} = 1$. The $\{\lambda_j\}_{j=k-n}^{k+m}$ are free parameters set by the scaling choices in Algorithm 3.2. For $j \le m$, λ_j is fixed directly by the choice in line 2, and for $j > m$, $\lambda_j = \widehat{h}_j^{(j-m)} \lambda_{j-m}$ is fixed by the choices made in line 18 during successive steps.

We compare the costs for Algorithm 3.2 with that of the Two-Sided Lanczos procedure without look-ahead or deflation in Table 4.1. There may be many variants of the limited Two-Sided procedure. In §7, we construct a version of the MPVL method for the purpose of making a comparison of the TFMPVL method with MPVL. The last column of Table 4.1 reflects the cost of the Two-Sided Lanczos procedure that underlies the MPVL produced there.

In arriving at the formulas in Table 4.1, we note that the power $1 + grade - g_n(k)$ of $\mathbf{A}$ in Line 6 of Algorithm 3.2 is zero whenever $k > n$. Hence this line normally does not involve a multiplication by $\mathbf{A}$. The multiplication by $\mathbf{A}$ normally occurs only in Lines 9 and 16. In each pass through the k–loop (Lines 4 - 20), the multiplication by $\mathbf{A}$ is guaranteed to occur once and sometimes twice, leading to the average cost estimate of $1 + m/n$ matrix-vector products given in Table 4.1. Regarding the storage requirements, the data $\{\boldsymbol{\phi}_{k-n}, \cdots, \boldsymbol{\phi}_{k+m}\}$, $\{\widehat{\mathbf{u}}_1, \cdots, \widehat{\mathbf{u}}_n\}$ and $\{b_{k-n}, \cdots, b_{k+m-1}\}$ of storage are required in the process of each k–loop in addition to the matrices $\mathbf{A}$ and

$\widehat{\mathbf{H}}$.

4. A Relation to Arnoldi Procedure. In this section, we consider the case where $n = N$, that is, the number of the left starting vectors $\widehat{\mathbf{u}}$ is equal to the size of $\mathbf{A}$. Theoretically, the indices k and i on Lines 3 and 7 of Algorithm 3.2 never exceed $N - m$ and N respectively. Therefore

$$grade = 0\,, \qquad 1 + grade - g_n(k) = 1\,, \qquad r_n(i) = i\,, \qquad r_n(k+m) = k+m$$

at all stages of the algorithm. Because of these observations, Algorithm 3.2 can be simplified as

Lanczos Procedure Version 3.

1. For $k = 1, 2, \ldots, m$, do:
   (i) compute $\mathbf{v}_k$ such that $\mathbf{v}_k \perp span\{\widehat{\mathbf{u}}_1, \cdots, \widehat{\mathbf{u}}_{k-1}\}$ and
   $\mathbf{v}_k \in span\{\widehat{\mathbf{v}}_1 \cdots, \widehat{\mathbf{v}}_k\}$;
   (ii) define λ_k, set $\boldsymbol{\phi}_k = \mathbf{v}_k \,/\, \lambda_k$ and compute $b_k = \widehat{\mathbf{u}}_k^H \boldsymbol{\phi}_k$.
2. For $k = 1, 2, 3, \ldots$
3. $\quad \boldsymbol{\phi}_{k+m} = \mathbf{A}\boldsymbol{\phi}_k$;
4. $\quad$ For $i = 1, 2, \cdots, k+m-1$
5. $\quad\quad \widehat{h}_i^{(k)} = \widehat{\mathbf{u}}_i^H \boldsymbol{\phi}_{k+m} \,/\, b_i$;
6. $\quad\quad \boldsymbol{\phi}_{k+m} = \boldsymbol{\phi}_{k+m} - \widehat{h}_i^{(k)} \boldsymbol{\phi}_i$;
7. $\quad$ End
8. $\quad$ Define $\widehat{h}_{k+m}^{(k)}$;
9. $\quad \boldsymbol{\phi}_{k+m} = \boldsymbol{\phi}_{k+m} \,/\, \widehat{h}_{k+m}^{(k)}$;
10. $\quad b_{k+m} = \widehat{\mathbf{u}}_{k+m}^H \boldsymbol{\phi}_{k+m}$;
11. End

Since the grade is always zero, the vectors $\boldsymbol{\phi}_k$ are the same as the original vectors $\mathbf{v}_k$ except for scaling:

$$\boldsymbol{\phi}_{k+m} = \boldsymbol{\psi}_{k+m}/\lambda_{k+m} = \mathbf{A}^{g_n(k+m)}\mathbf{v}_{k+m}/\lambda_{k+m} = \mathbf{v}_{k+m}/\lambda_{k+m}\,,$$

where $\lambda_{k+m} = \widehat{h}_{k+m}^{(k)}\lambda_k / \bar{h}_{k+m}^{(k)} = \widehat{h}_{k+m}^{(k)}\lambda_k$ by (3.5) and the fact that $\bar{h}_{k+m}^{(k)} = 1$ from Algorithm 3.1. Properties (2.3) then imply that

$$span\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_k\} = \mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right) \tag{4.1}$$

and

$$\boldsymbol{\phi}_k \perp span\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{k-1}\} = span\{\widehat{\mathbf{u}}_1, \widehat{\mathbf{u}}_2, \cdots, \widehat{\mathbf{u}}_{k-1}\}. \tag{4.2}$$

The equation in (4.2) holds because $k \le N - m$ and $\mathbf{p}_i = \left(\mathbf{A}^H\right)^{g_n(i)} \widehat{\mathbf{u}}_{r_n(i)} = \left(\mathbf{A}^H\right)^{g_N(i)} \widehat{\mathbf{u}}_{r_N(i)} = \widehat{\mathbf{u}}_i$ by (2.2a) where $1 \le i \le N$.

We now show that Version 3 includes Arnoldi procedure as a special case, when the left vectors are obtained during the course of the algorithm. Let be given a set of starting vectors $\widehat{\mathbf{v}}_1, \widehat{\mathbf{v}}_2, \cdots, \widehat{\mathbf{v}}_m$. A band Arnoldi procedure (see, for instance, §6 of [17]) generates a sequence of vectors $\{\mathbf{v}_k\}_{k=1,2,\ldots}$ such that

$$span\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k\} = \mathcal{G}_k\left(\mathbf{A}, \widehat{\mathbf{V}}\right) \quad \text{and} \quad \mathbf{v}_i^H \mathbf{v}_j = \begin{cases} 1\,, & i = j \\ 0\,, & i \ne j \end{cases} \tag{4.3}$$

TABLE 4.1

Average cost per k–loop of Algorithm 3.2 and its total storage requirement, compared to the Two-Sided Lanczos Procedure without deflation or look-ahead.

Item	Average count Algorithm 3.2 without $\mathbf{v}$'s	Average count Algorithm 3.2 with $\mathbf{v}$'s	Average count Two-Sided Lanczos without deflation or look-ahead
Matrix vector product	$1+\frac{m}{n}$	$2+\frac{m}{n}$	2
Saxpy	$m+n+1$	$2(m+n)+1$	$2(m+n)$
Inner product	$m+n+1$	$m+n+1$	$2(m+n+1)$
Average Storage beyond $\mathbf{A}$, $\widehat{\mathbf{H}}$	$(m+2n+1)N$ $+\,m+n$	$(2m+3n+2)N$ $+\,m+n$	$2(m+n+1)N$ $+\,2(m+n)$

if no deflation appears, where the block Krylov subspace $\mathcal{G}_k\left(\mathbf{A},\widehat{\mathbf{V}}\right)$ is defined by (2.4). We remark that the band Arnoldi process [17] generates the Arnoldi vectors one single vector at a time. The Arnoldi vectors can also be generated one block at a time via a block Arnoldi process [9, 8], using a different way to decide deflations.

Version 3 involves two sets of vectors, $\{\boldsymbol{\phi}_k\}_{k=1,2,\cdots}$ and $\{\widehat{\mathbf{u}}_k\}_{k=1,2,\cdots,N}$. If we consider the set $\{\widehat{\mathbf{u}}_k\}$ to be a set of free parameters and define $\lambda_k=\|\mathbf{v}_k\|_2$, $\widehat{\mathbf{u}}_k=\boldsymbol{\phi}_k$ on Line 1(ii), $\widehat{h}^{(k)}_{k+m}=\|\boldsymbol{\phi}_{k+m}\|_2$ on Line 8 and $\widehat{\mathbf{u}}_{k+m}=\boldsymbol{\phi}_{k+m}$ on Line 10, then the vectors $\boldsymbol{\phi}$ computed by Version 3 are the Arnoldi vectors and Version 3 is a band Arnoldi procedure. In fact, the vectors $\boldsymbol{\phi}$ computed are unit vectors and satisfy the Arnoldi properties (4.3) because of (4.1), (4.2) and $\widehat{\mathbf{u}}_i=\boldsymbol{\phi}_i$.

5. A Transpose-free Version of the MPVL Method. In this section, we present one application of the transpose-free multiple Lanczos procedure of Algorithm 3.2.

We consider the task of model reduction via Padé approximation on a multi-input multi-output (MIMO) linear dynamical system

$$\mathbf{C}\frac{d\mathbf{x}}{dt}=-\mathbf{G}\mathbf{x}(t)+\mathbf{R}\mathbf{w}(t),\qquad \mathbf{y}(t)=\mathbf{L}^H\mathbf{x}(t),$$

where $\mathbf{C},\mathbf{G}\in\mathcal{C}^{N\times N}$, $\mathbf{R}\in\mathcal{C}^{N\times m}$, $\mathbf{L}\in\mathcal{C}^{N\times n}$, and $\mathbf{w}(t),\mathbf{y}(t)$ and $\mathbf{x}(t)$ are vector-valued functions of length m,n and N, respectively. For the sake of simplicity, we assume the initial condition $\mathbf{x}(0)=0$. See, for instance, [11, 12, 14, 15].

Corresponding to this system is the matrix-valued transfer function $\mathbf{F}(z)$ mapping the input $\mathbf{W}(z)$ to the output $\mathbf{Y}(z)$ in frequency domain:

$$\mathbf{Y}(z)=\mathbf{L}^H(z\mathbf{C}+\mathbf{G})^{-1}\mathbf{R}\cdot\mathbf{W}(z)\equiv\mathbf{F}(z)\cdot\mathbf{W}(z). \tag{5.1}$$

To compute $\mathbf{F}(z)$, write $z=z_0+\theta$. Then we can express the transfer function in terms of the matrices and expand it in a power series around $z=z_0$:

$$\begin{aligned}\mathbf{F}(z) &= \mathbf{L}^H(z\mathbf{C}+\mathbf{G})^{-1}\mathbf{R}\\ &= \mathbf{L}^H(z_0\mathbf{C}+\mathbf{G}+\theta\mathbf{C})^{-1}\mathbf{R}\\ &= \mathbf{L}^H(\mathbf{I}+\theta(z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{C})^{-1}(z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{R}\\ &= \widehat{\mathbf{U}}^H(\mathbf{I}-\theta\mathbf{A})^{-1}\widehat{\mathbf{V}}\\ &= \sum_{k=0}^{\infty}\mathbf{M}_k\theta^k,\end{aligned} \tag{5.2}$$

where

$$\mathbf{A} = -(z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{C}, \quad \widehat{\mathbf{U}} = \mathbf{L}, \quad \widehat{\mathbf{V}} = (z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{R}, \quad \mathbf{M}_k = \widehat{\mathbf{U}}^H\mathbf{A}^k\widehat{\mathbf{V}}. \tag{5.3}$$

The matrix coefficients $\mathbf{M}_k$'s in the power series are often called the *moments* or *Markov parameters*. Our goal is to seek a new lower order system

$$\frac{d\breve{\mathbf{x}}}{dt} = \breve{\mathbf{A}}\breve{\mathbf{x}} + \breve{\mathbf{V}}\mathbf{w}(t), \qquad \breve{\mathbf{y}}(t) = \breve{\mathbf{U}}^H\breve{\mathbf{x}}(t) \tag{5.4}$$

with frequency domain description

$$\breve{\mathbf{Y}}(z) = \breve{\mathbf{U}}^H(\mathbf{I}-\theta\breve{\mathbf{A}})^{-1}\breve{\mathbf{V}}\cdot\mathbf{W}(z) = \sum_{k=0}^{\infty}\breve{\mathbf{M}}_k\,\theta^k\cdot\mathbf{W}(z),$$

that approximates the original (5.1) in the sense that as many terms $\breve{\mathbf{M}}_k$ as possible agree with the corresponding terms $\mathbf{M}_k$ in the original power series. Specifically, we seek a Padé approximant defined in the following definition.

DEFINITION 5.1. *An l-th Padé approximant $\mathbf{f}_l(\theta)$ of $\mathbf{F}(\theta+z_0)$ is defined to be a function of the form*

$$\mathbf{f}_l(\theta) = \breve{\mathbf{U}}^H(\mathbf{I}-\theta\breve{\mathbf{A}})^{-1}\breve{\mathbf{V}} \tag{5.5}$$

whose Taylor expansion about $\theta = 0$ matches as many leading terms of the Taylor expansion (5.2) of $\mathbf{F}(\theta+z_0)$ as possible, where $\breve{\mathbf{U}} \in \mathcal{C}^{l\times n}, \breve{\mathbf{V}} \in \mathcal{C}^{l\times m}, \breve{\mathbf{A}} \in \mathcal{C}^{l\times l}$ and $\mathbf{I}$ is the $l\times l$ identity matrix. See, for instance, [14, 15].

With the given blocks $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ of (5.3) as the n left starting vectors and m right starting vectors respectively in the Two-Sided Lanczos procedure, we obtain data $\mathbf{U}_\nu, \mathbf{V}_\nu, \bar{\mathbf{H}}$ and $\widetilde{\mathbf{H}}$. Because of (2.10), there exist matrices $\boldsymbol{\eta} \in \mathcal{C}^{n\times n}$ and $\boldsymbol{\rho} \in \mathcal{C}^{m\times m}$ such that

$$\widehat{\mathbf{U}} = \mathbf{U}_n\boldsymbol{\eta} \qquad \text{and} \qquad \widehat{\mathbf{V}} = \mathbf{V}_m\boldsymbol{\rho}. \tag{5.6}$$

Let $\bar{\mathbf{H}}_k$ be the $k\times k$ principal block of the matrix $\bar{\mathbf{H}}$, $\mathbf{0}_{k'\times k}$ denote the $k'\times k$ zero matrix and set

$$\mathbf{D}_k = \mathbf{U}_k^H\mathbf{V}_k\,. \tag{5.7}$$

Then the following theorem [14, 15] provides us an l-th Padé approximant.

THEOREM 5.2. *Let $\max\{m,n\} \le l$. Then,*

$$\mathbf{f}_l(\theta) = \begin{bmatrix} \mathbf{D}_n^H\boldsymbol{\eta} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I}-\theta\bar{\mathbf{H}}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix} \tag{5.8}$$

is an l-th Padé approximant of the function $\mathbf{F}(\theta+z_0)$ and

$$\mathbf{f}_l(\theta) = \mathbf{F}(\theta+z_0) + O(\theta^{\lfloor l/m\rfloor+\lfloor l/n\rfloor})$$

on the disc $\{\theta : |\theta| < 1/\delta\}$ where $\delta = \max\{\delta(\mathbf{A}), \delta(\bar{\mathbf{H}}_l)\}$ and $\delta(\mathbf{M})$ is the spectral radius of a matrix $\mathbf{M}$.

In [12, 14], the MPVL method was proposed to compute $\mathbf{f}_l(\theta)$ based on Theorem 5.2 using the Two-Sided Lanczos algorithm. The MPVL method consists of two steps: (a) the Two-Sided Lanczos procedure is carried out for l steps to obtain data $\bar{\mathbf{H}}_l, \mathbf{D}_n, \boldsymbol{\eta}$

and $\boldsymbol{\rho}$, then (b) the state-space realization (5.4) for l-th order Padé approximant is formed based on (5.8):

$$\frac{d\tilde{\mathbf{x}}}{dt} = \bar{\mathbf{H}}_l \tilde{\mathbf{x}}(t) + \begin{bmatrix} \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix} \mathbf{w}(t), \qquad \tilde{\mathbf{y}}(t) = \begin{bmatrix} \mathbf{D}_n^H \boldsymbol{\eta} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \tilde{\mathbf{x}}(t).$$

Observe that the Two-Sided Lanczos procedure generates not only the data $\bar{\mathbf{H}}_l, \mathbf{D}_n, \boldsymbol{\eta}, \boldsymbol{\rho}$ but $\mathbf{U}_l, \mathbf{V}_l, \widetilde{\mathbf{H}}_l$ as well. However, the data $\mathbf{U}_l, \mathbf{V}_l$ and $\widetilde{\mathbf{H}}_l$ do not contribute directly to compute $\mathbf{f}_l(\theta)$ in (5.8). Instead, they are used only to obtain the matrix $\bar{\mathbf{H}}_l$ in the Lanczos procedure itself. The question then arises as to whether or not it is possible in the Two-Sided Lanczos procedure to bypass the computations of $\mathbf{U}_l, \mathbf{V}_l$ and $\widetilde{\mathbf{H}}_l$ and still generate the quantities that are related to (5.8). The transpose-free procedure of Algorithm 3.1 or 3.2 provides an answer to this question. In the following, we will derive a transpose-free version of the MPVL method from Algorithm 3.2.

We first express the $\mathbf{f}_l(\theta)$ of (5.8) in terms of the quantities computed by Algorithm 3.2. Since $\widehat{\mathbf{U}} = \mathbf{U}_n \boldsymbol{\eta}$ and $\mathbf{D}_n = \mathbf{U}_n^H \mathbf{V}_n$ from (5.6) and (5.7), we have

$$\mathbf{f}_l(\theta) \;=\; \begin{bmatrix} \mathbf{V}_n^H \widehat{\mathbf{U}} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I} - \theta \bar{\mathbf{H}}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix}.$$

If we let $\widehat{\mathbf{H}}_l$ and $\boldsymbol{\Lambda}_l$ denote the $l \times l$ principal blocks of the matrices $\widehat{\mathbf{H}}$ and $\boldsymbol{\Lambda}_\nu$ respectively, then we have $\widehat{\mathbf{H}}_l = \boldsymbol{\Lambda}_l \bar{\mathbf{H}}_l \boldsymbol{\Lambda}_l^{-1}$ from (3.4) and therefore

$$\begin{aligned} \mathbf{f}_l(\theta) \;&=\; \begin{bmatrix} \mathbf{V}_n^H \widehat{\mathbf{U}} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I} - \theta \boldsymbol{\Lambda}_l^{-1} \widehat{\mathbf{H}}_l \boldsymbol{\Lambda}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix} \\ &=\; \begin{bmatrix} \boldsymbol{\Lambda}_n^{-1} \mathbf{V}_n^H \widehat{\mathbf{U}} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I} - \theta \widehat{\mathbf{H}}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\Lambda}_m \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix} \qquad (5.9) \\ &=\; \begin{bmatrix} \boldsymbol{\Omega}_n^H \widehat{\mathbf{U}} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I} - \theta \widehat{\mathbf{H}}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\Lambda}_m \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix}, \end{aligned}$$

where $\boldsymbol{\Omega}_n = [\mathbf{v}_1/\lambda_1, \cdots, \mathbf{v}_n/\lambda_n] = [\boldsymbol{\psi}_1/\lambda_1, \cdots, \boldsymbol{\psi}_n/\lambda_n] = [\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_n]$ by (3.1) and (3.5).

We are now ready to present a transpose-free implementation of the MPVL method in the following algorithm. We remark that the initial $m \times m$ matrix $\boldsymbol{\rho}$ can be computed via a modified two-sided Gram-Schmidt-type process [12, 22].

ALGORITHM 5.3. **Transpose-free MPVL (TFMPVL)** *Given m right starting vectors $\{\widehat{\mathbf{v}}_k\}_{k=1}^m$ and n left starting vectors $\{\widehat{\mathbf{u}}_{k'}\}_{k'=1}^n$ with $m \le n$. Suppose the assumption of Theorem 2.3 holds. The following algorithm computes an l-th Padé approximant $\mathbf{f}_l(\theta)$ of the transfer function $\mathbf{F}(\theta + z_0)$ described in Theorem 5.2.*

1. *Compute $\boldsymbol{\rho}$ via a modified two-sided Gram-Schmidt-type process*
2. *Run l steps of the transpose-free Lanczos process with multiple starting vectors (Algorithm 3.2) to obtain $\boldsymbol{\Lambda}_m, \boldsymbol{\Omega}_n, \boldsymbol{\rho}$ and $\widehat{\mathbf{H}}_l$.*
3. *Compute $\mathbf{f}_l(\theta)$ according to (5.9).*

Algorithm 5.3 requires about $(1 + m/n)l$ matrix-vector products with $\mathbf{A}$ to get the l-th Padé approximant $\mathbf{f}_l(\theta)$. When $n > m$, it is cheaper than MPVL to get $\mathbf{f}_l(\theta)$ (see Table 4.1).

6. An Augmented version of TFMPVL. The derivation of Algorithm 5.3 is based on Theorem 2.3. If the assumptions of the theorem hold, breakdown will not occur and deflation (see, for instance, [1]) will not be needed within the first ν steps when we run Algorithm 3.2. An ν-th Padé approximant $\mathbf{f}_\nu(\theta)$ is therefore guaranteed. In practice, however, breakdown and deflation are unavoidable — especially, deflation is guaranteed to occur at some point. So, they must be handled eventually. In the following, we propose a use of Algorithm 5.3 which empirical evidence suggests can help avoid *near-breakdown* and *inexact* deflation, but not the *exact* situation (see the remarks at the end of this section and see [1, pp.6-8] for the meanings of *near-breakdown* and *inexact* deflation). This idea can also be applied to the MPVL method, but the effect is not as pronounced as for TFMPVL from our experiments (see §7).

About Algorithm 5.3, we can augment the input/output data $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ before using it by adding some random vectors, say $\mathbf{r}_{\widehat{U}} \in \mathcal{C}^{N\times n_0}$ and $\mathbf{r}_{\widehat{V}} \in \mathcal{C}^{N\times m_0}$, as follows

$$\widehat{\mathbf{U}}_{aug} = [\mathbf{r}_{\widehat{U}}, \widehat{\mathbf{U}}], \qquad \widehat{\mathbf{V}}_{aug} = [\mathbf{r}_{\widehat{V}}, \widehat{\mathbf{V}}]\,. \tag{6.1}$$

Then, the matrix-valued function $\mathbf{F}(\theta + z_0)$ of (5.2) which we want to estimate is just the diagonal block at the lower-right corner of the matrix-valued function

$$\mathbf{F}_{aug}(\theta + z_0) = \widehat{\mathbf{U}}_{aug}^H(\mathbf{I} - \theta\mathbf{A})^{-1}\widehat{\mathbf{V}}_{aug}\,.$$

We now apply Algorithm 5.3 to find a Padé approximant $\mathbf{f}_l(\theta)$ to the function

$$\widehat{\mathbf{U}}_{aug}^H(\mathbf{I} - \theta\mathbf{A})^{-1}\widehat{\mathbf{V}}_{aug}$$

and then use the lower-right $n\times m$ corner block of $\mathbf{f}_l(\theta)$ as an approximant to $\mathbf{F}(\theta+z_0)$. We call this approximation approach an *Augmented TFMPVL*.

We remark that the augmented method does not eliminate *exact deflation*. Suppose the left (or right) starting vectors $\widehat{\mathbf{U}}$ (or $\widehat{\mathbf{V}}$) lies in an invariant subspace of small dimension. Then *exact* deflation in the Lanczos process will occur after very few steps (at most the dimension of the invariant subspace). Suppose we prepend some random vectors in front on the left of $\widehat{\mathbf{U}}$ (or $\widehat{\mathbf{V}}$). Then we will be able to generate larger Krylov spaces, but the original starting vectors will still lie in some invariant subspace, and we will still suffer *exact* deflation after only a few "block" steps. Hence the deflation problem is postponed, but not eliminated.

The augmented method does not eliminate *exact breakdown* either. When one of the leading principal submatrices $\mathbf{W}_k$ (defined in §2) is singular, *exact* breakdown is likely to occur. For clarity, we let $\mathbf{W}_k(\mathbf{B}_l, \mathbf{B}_r)$ denote the corresponding $\mathbf{W}_k$-matrix associated with left starting block $\mathbf{B}_l$ and right starting block $\mathbf{B}_r$. It is then easy to see that $\mathbf{W}_k(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})$ is a submatrix of $\mathbf{W}_{k'}(\widehat{\mathbf{U}}_{aug}, \widehat{\mathbf{V}}_{aug})$ for some $k' \geq k$. For $\mathbf{W}_{k'}(\widehat{\mathbf{U}}_{aug}, \widehat{\mathbf{V}}_{aug})$, that adding extra random vectors $\mathbf{r}_{\widehat{U}}$, $\mathbf{r}_{\widehat{V}}$ does not guarantee its nonsingularity. Let us go to the extreme case, for example, where $\mathbf{W}_k(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}) = \mathbf{0}$, $m_0 = 0$ and $n_0 = 1$. It is then obvious that $\mathbf{W}_{k'}(\widehat{\mathbf{U}}_{aug}, \widehat{\mathbf{V}}_{aug})$ is singular. Hence Lanczos process starting with $\widehat{\mathbf{U}}_{aug}$ and $\widehat{\mathbf{V}}_{aug}$ can encounter *exact* breakdown at step k'.

However, empirical evidence shown in the next section seems to indicate that the augmented TFMPVL can have a mitigating effect on *near-breakdown* and *inexact deflation*. Inexact deflation depends on the conditioning among the Lanczos vectors generated, and the presence of random vectors generally reduces that conditioning, enhancing the numerical independence among the vectors. In addition the empirical

TABLE 7.1
Collection of some information of the experiments in Example 1 of §7.

Title	Size	Vectors		Exp.	Lanc	Padé	Time (secs)		#mat-vecs		Fig.
		$\widehat{\mathbf{U}}_{aug}$	$\widehat{\mathbf{V}}_{aug}$	point	steps	order	TF	MP	TF	MP	#
	N	n+n_0	m+m_0	z_0	l						
PEEC	306	1+0	1+0	$5\pi 10^9 i$	60	120	11.87	12.26	120	120	7.1(a)
PEEC	306	1+5	1+0	$5\pi 10^9 i$	60	70	10.34	12.26	70	115	7.1(b)

results in the next section illustrate that the use of random vectors can actually help improve the approximation even with a lower order approximation. The Lanczos vectors would form a basis for the projection from the original system to the Padé approximant, and this experimental evidence suggests that the projection arising from an augmented algorithm is better conditioned.

7. Numerical Experiments. In this section, we present some examples to illustrate the effectiveness of Algorithm 5.3. In all the experiments, we defined the parameters λ_k and $\widehat{h}_{k+m}^{(k)}$ in Algorithm 3.2 as follows,

$$\lambda_k = \|v_k\|_2, \qquad \widehat{h}_{k+m}^{(k)} = \|\boldsymbol{\phi}_{k+m}\|_2,$$

With this choice of the parameters, we can normalize the vectors $\boldsymbol{\phi}$, which could become very large or small in implementing the algorithm. All the experiments were performed in Matlab Version 6.0.0.88 Release 12 and some information has been collected in Tables 7.1, 7.2 and 7.3.

For the purpose of making a comparison of TFMPVL with MPVL, we produced one version of the MPVL method of our own without deflation and look-ahead. This version of MPVL just combined a scaled Two-Sided Lanczos procedure and formula (5.9). In the scaled Two-Sided Lanczos procedure, the Lanczos vectors $\mathbf{u}$ and $\mathbf{v}$ (see Theorem 2.3) are scaled to unit vectors. We remark that this version of MPVL does not reduce to the PVL method proposed in [11] since we did not compute or use the eigenvalues of $\widehat{\mathbf{H}}$.

In the implementations of TFMPVL and MPVL, we did not explicitly form the matrix $\mathbf{A}$ in (5.3). Instead, we computed the matrix-vector products involving $\mathbf{A}$ and $\mathbf{A}^H$ with (respectively):

$$\mathbf{A}\mathbf{v} = -(z_0\mathbf{C} + \mathbf{G})^{-1}(\mathbf{C}\mathbf{v}) \tag{7.1}$$

$$\mathbf{A}^H\mathbf{v} = -\mathbf{C}^H\left[(z_0\mathbf{C} + \mathbf{G})^{-H}\mathbf{v}\right]. \tag{7.2}$$

Example 1. This is the same example used in [2, 3, 11] from a three-dimensional electromagnetic problem model via PEEC (partial element equivalent circuit) [23]. It is regarded as a benchmark and difficult test problem. The matrices $\mathbf{C}$ and $\mathbf{G}$ have order 306 and both $\mathbf{L}$ and $\mathbf{R}$ are column vectors. Hence, the transfer function $\mathbf{F}(z)$ in (5.2) is a scalar-valued function.

In [2, 3, 11], the magnitude of $\mathbf{F}(z)$ with $z = 2\pi wi$ was approximated over the frequency interval $1 \le w \le 5 \times 10^9$ with a 60-th Padé approximant $\mathbf{f}_{60}(\theta)$ in Theorem 5.2 obtained by the PVL method, where $\theta = 2\pi wi - z_0$. The numerical results therein illustrated that the approximation produced by PVL was indistinguishable from the true $|\mathbf{F}(2\pi wi)|$.

We repeated the above experiment with the expansion point $z_0 = 5\pi 10^9\sqrt{-1}$. TFMPVL (Algorithm 5.3) and MPVL were employed to compute $\mathbf{f}_{60}(\theta)$. We plotted the results in Fig. 7.1(a) and observed that the approximation by MPVL was almost indistinguishable from the exact curve, but that made by TFMPVL was not accurate in the high and low frequency w-regions. Theoretically, both PVL and TFMPVL produce the same $\mathbf{f}_{60}(\theta)$. Numerically, however, the TFMPVL method is less stable in this experiment.

We continued the experiment of Fig. 7.1(a). This time, we used Augmented TFMPVL and MPVL of §6 with $m_0 = 0$ and $n_0 = 5$ to compute $\mathbf{f}_{60}(\theta)$. We can see from the results plotted in Fig. 7.1(b) that the approximation by TFMPVL has been improved and matches the exact curve very well. However, the approximation by MPVL becomes worse. The corresponding relative errors $|\mathbf{F}(2\pi wi)-\mathbf{f}_{60}(\theta)|/|\mathbf{F}(2\pi wi)|$ were shown in Fig. 7.2(a).

In order to provide an explanation to the phenomenon observed, we measured the conditioning (numerical independence) among the Lanczos vectors $\mathbf{u}$ and $\mathbf{v}$ (see Theorem 2.3) and the sizes of their inner-products $\mathbf{u}_k^H\mathbf{v}_k$ generated in the experiments of Fig. 7.1(a) and Fig. 7.1(b) respectively.

Noting that Algorithm 3.2, based on which TFMPVL was built, does not directly generate $\mathbf{u}$'s or $\mathbf{v}$'s, we therefore computed the condition numbers for the $\boldsymbol{\phi}$'s instead. Moreover, we only computed the condition numbers of $\mathbf{v}$'s from the Two-Sided Lanczos procedure of MPVL as an illustration. The 2-norm condition numbers of the matrices $[\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{k+m}]$ and $[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k+m}]$ against the iteration index k were plotted in Fig. 7.3(a). With the addition of some random vectors, we can see that the conditioning among the $\boldsymbol{\phi}$'s generated in Fig. 7.1(b) has had a big improvement over that generated in Fig. 7.1(a). On the other hand, however, the conditioning among the $\mathbf{v}$'s from MPVL almost experienced no improvement.

If inner-products among Lanczos vectors were too close to zero in magnitude, a Lanczos procedure would face instability or even break down. The inner-products in Algorithm 3.2 (TFMLP) are already computed as the b_{k+m}'s. Let c_{k+m} denote the inner-product $\mathbf{u}_{k+m}^H\mathbf{v}_{k+m}$ in the Two-Sided Lanczos procedure of MPVL. In Fig. 7.2(b), we plotted the absolute values of b's and c's obtained in the experiments of Figures 7.1(a) and 7.1(b). From the figure, we can see that the $|b_{k+m}|$ related to Fig. 7.1(a) quickly dropped down to a size of 10^{-16} as k increased. However, the quick drop-down was overcome in the experiment of Fig. 7.1(b) by adding some random vectors. Surpringly, $|c_{k+m}|$ behaved reversely. It tended to become even smaller with the existence of random vectors.

The above experiments about condition numbers and inner-products illustrate that *inexact deflation* and *near-breakdown* in Algorithm 3.2 can be partially sidestepped with the addition of random vectors. For Two-Sided Lanczos procedure, however, the situation can be made even worse. The observations help to explain why Augmented TFMPVL in Fig. 7.1(b) outperformed TFMPVL in Fig. 7.1(a) and Augmented MPVL in Fig. 7.1(b) behaved not as well as MPVL in 7.1(a).

Some information of the experiments in Example 1 has been collected in Table 7.1 for a quick reference.

Example 2. The test data used in this example were from *Benchmark examples for model reduction of linear time invariant dynamical systems, The Control and Systems Library SLICOT* [7]. Information of the experiments carried out were collected in Table 7.2 which shows 1) test data we selected from the library; 2) corresponding system dimensions; 3) numbers of columns in input and output matrices; 4) expansion points

TABLE 7.2
Collection of some information of the experiments in Example 2 of §7.

Title	Size	Vectors		Exp.	Lanc	Padé	Time (secs)		#mat-vecs		Fig.
		$\hat{\mathbf{U}}_{aug}$	$\hat{\mathbf{V}}_{aug}$	point	steps	order	TF	MP	TF	MP	#
	N	n+n_0	m+m_0	z_0	l						
Tline	256	2+0	2+0	$2\pi 10^9$	30	30	1.23	1.37	60	60	7.3(b)
CD-player	120	2+0	2+0	$10^2\pi$	30	30	0.20	0.09	60	60	7.4(a)
CD-player	120	2+2	2+0	$10^2\pi$	40	30	0.31	0.14	60	78	7.4(b)
MNA1	578	9+0	9+0	$10^{10}\pi$	90	20	14.90	14.07	180	180	7.5(a)
MNA4	980	4+0	4+0	$10^{10}\pi$	40	20	23.90	23.44	80	80	7.5(b)
MNA4	980	4+4	4+0	$10^{10}\pi$	56	21	25.14	31.77	84	108	7.6(a)
MNA3	4863	22+0	22+0	$10^{10}\pi$	220	20	11613	11670	440	440	7.6(b)
MNA2	9223	18+0	18+0	$10^{10}\pi$	180	20	51535	52280	360	360	7.7(a)
MNA5	10913	9+0	9+0	$10^{10}\pi$	90	20	2688	2671	180	180	7.7(b)
MNA5	10913	9+9	9+0	$10^{10}\pi$	126	21	2822	3613	189	234	7.8(a)
ISS	270	3+0	3+0	20π	30	20	0.29	0.15	60	60	7.8(b)

TABLE 7.3
Collection of some information of the experiments in Example 3 of §7. TFMPVL and MPVL were applied to (5.2).

Title	Size	Vectors		Exp.	Lanc	Padé	Time (secs)		#mat-vecs		Fig.
		$\hat{\mathbf{U}}_{aug}$	$\hat{\mathbf{V}}_{aug}$	point	steps	order	TF	MP	TF	MP	#
	N	n+n_0	m+m_0	z_0	l						
Power Plants	421	4+0	2+0	$2\pi 10^5$	60	45	23.80	17.10	90	118	7.9(a)
Power Plants	421	4+4	2+0	$2\pi 10^5$	60	37	19.99	16.83	74	114	7.9(b)

z_0 used; 5) Lanczos steps l performed in the construction of Padé approximations; 6) Padé orders $\lfloor l/(m+m_0)\rfloor + \lfloor l/(n+n_0)\rfloor$ of the corresponding approximations (see Theorem 5.2); 7) operation times in seconds consumed by TFMPVL and MPVL; 8) actual numbers of matrix-vector multiplies performed by TFMPVL and MPVL in the computations; 9) figures in which numerical results were plotted.

Overall, the performance of TFMPVL and MPVL were quite similar on the data that we selected. In the case where $n+n_0 > m+m_0$, TFMPVL performed slightly better. The experiments illustrated that the addition of random vectors could help improve the stability of both methods. In terms of operation time, TFMPVL was about the same with MPVL when $m = n$ and faster than MPVL when $m < n$ in most cases. It is because the overall computational cost of TFMPVL is less than that of MPVL in general when $n \geq m$ (see Table 4.1).

Example 3. In [4, 5, 6], Benner *et al* collected many examples for the numerical solution of continuous-time and discrete-time algebraic Riccati equations. This third example is the 421×421 system from Example 20 of [4] which describes a problem arising in power plants. The matrix $\mathbf{C}$ is an identity matrix. Both their (output) matrix $\mathbf{L}_o$ and their (input) matrix $\mathbf{R}_o$ contain 211 columns. We generated the data by the Matlab function *carex* provided by [4].

In this example, we picked the first four and the first two columns of $\mathbf{L}_o$ and

TABLE 7.4

Collection of some information of the experiments in Example 3 of §7. TFMPVL and MPVL were applied to (7.3).

Title	Size	Vectors		Exp.	Lanc	Padé	Time (secs)		#mat-vecs		Fig.
		$\widehat{\mathbf{U}}_{aug}$	$\widehat{\mathbf{V}}_{aug}$	point	steps	order	TF	MP	TF	MP	#
	N	n+n_0	m+m_0	z_0	l						
Power Plants	421	2+2	4+0	$2\pi 10^5$	60	30	2.65	16.90	120	120	7.10(a)
Power Plants	421	2+6	4+0	$2\pi 10^5$	60	22	2.27	15.74	88	116	7.10(b)

$\mathbf{R}_o$ as our (output) matrix $\mathbf{L}$ and (input) matrix $\mathbf{R}$ respectively. We simulated the 4×2 matrix-valued function $\mathbf{F}(z)$ of (5.2) with $z = 2\pi wi$ over the frequency interval $1 \le w \le 10^{12}$. A 60-th Padé approximant $\mathbf{f}_{60}(\theta)$ in Theorem 5.2 was computed by TFMPVL of Algorithm 5.3 and our version of MPVL respectively, where $\theta = 2\pi wi - z_0$ and the expansion point $z_0 = 2\pi 10^5$. Numerical results of the approximation to the $(1,2)$th element of $\mathbf{F}(z)$ are showed in Figures 7.9(a) and 7.9(b).

We can see that TFMPVL performs better than MPVL in this experiment. With the addition of 4 random vectors (i.e., $n_0 = 4$), the approximation by TFMPVL is even better while that by MPVL becomes worse (see Fig. 7.9(b)). Noting that $n > m$, TFMPVL needs fewer matrix-vector multiplies to get $\mathbf{f}_{60}(\theta)$ compared to MPVL (see Table 4.1). An actual count of the numbers of matrix-vector multiplies is listed in the column under "#mat-vecs" in Table 7.3. One shortcoming about TFMPVL in these experiments is that the operation time consumed was longer than that by MPVL. It is not hard to find the reason. Consider the experiment of 7.9(a) for example. TFMPVL performed 90 matrix-vector multiplies with $\mathbf{A}$ and, meanwhile, MPVL performed 59 with $\mathbf{A}$ and 59 with $\mathbf{A}^H$. However, we observed that computing $\mathbf{Av}$ by (7.1) took much longer in our Matlab implementation than computing $\mathbf{A}^H\mathbf{v}$ by (7.2). As a result, the operation time consumed by TFMPVL was longer.

To reduce the operation time for TFMPVL, we can use the conjugate-transpose of $\mathbf{F}(z)$ instead:

$$\mathbf{F}(z)^H = \mathbf{R}^H(\bar{z}\,\mathbf{C}^H + \mathbf{G}^H)^{-1}\mathbf{L}. \tag{7.3}$$

The matrix-vector multiplies in TFMPVL will then all have the form $\mathbf{A}^H\mathbf{v}$. We carried out experiments according to this idea and results of the approximation to the abolute value of the $(2,1)$th element of $\mathbf{F}(z)^H$ were plotted in Figures 7.10(a) and 7.10(b) respectively. Note that the output matrix now is $\mathbf{R}$ which has 2 columns and the input matrix is $\mathbf{L}$ that contains 4 columns. Therefore, $n = 2$ and $m = 4$. Recall that TFMPVL requires $n \ge m$ as a prerequisite to implement. To solve the problem, we added some random vectors to $\widehat{\mathbf{U}}$ to increase n and used Augmented TFMPVL in the experiments. Some information has been collected in Table 7.4. From Table 7.4, we can see that the operation time taken by TFMPVL has been significantly reduced (compare the "Time" columns of Tables 7.3 and 7.4). We also applied MPVL to (7.3) for a comparison to TFMPVL.

8. Discussion: Detecting Deflation. In this section, we present some steps to take with the eventual goal of handling the deflation issue in transpose-free multiple Lanczos procedure. Since it is complicated and difficult to consider deflation and breakdown simultaneously, we only consider deflation and assume that no breakdown

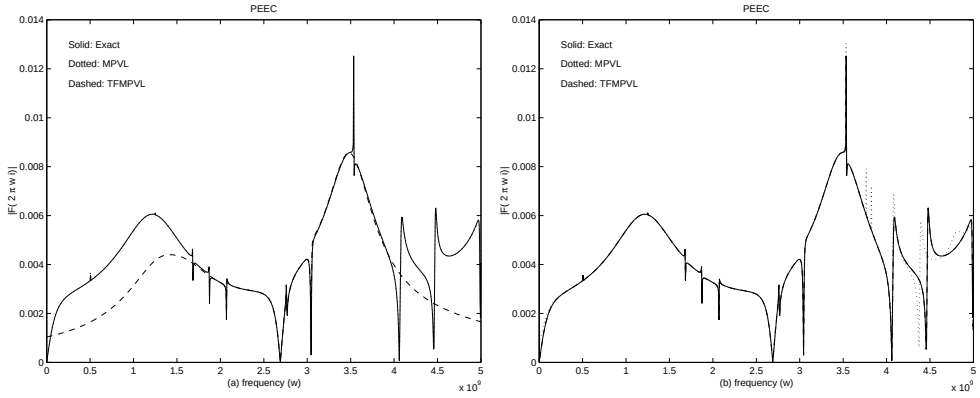

FIG. 7.1. *Graphs of* $|\mathbf{F}(2\pi wi)|$ *and* $|\mathbf{f}_{60}(\theta)|$. *Expansion point* $z_0 = 5\pi 10^9\sqrt{-1}$. *(a)* $\mathbf{f}_{60}(\theta)$ *was computed by ordinary TFMPVL and MPVL. (b)* $\mathbf{f}_{60}(\theta)$ *was computed by Augmented TFMPVL and MPVL with* $m_0 = 0$ *and* $n_0 = 5$. *Solid: exact; Dotted: MPVL; Dashed: TFMPVL.*

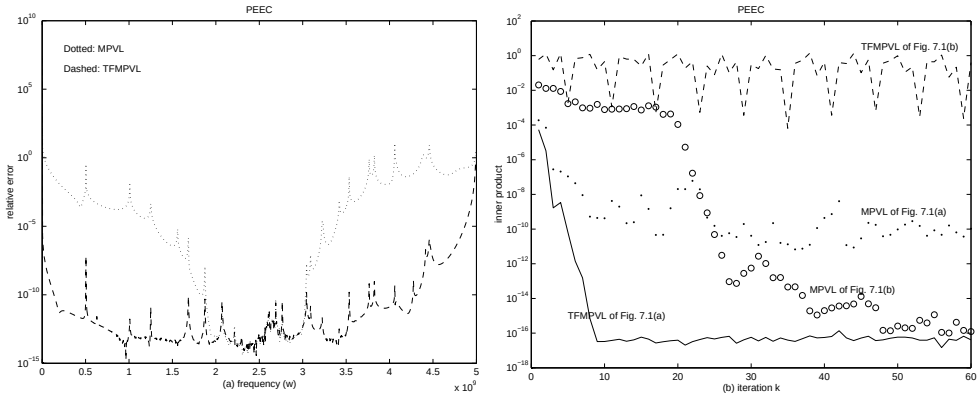

FIG. 7.2. *(a) Relative errors of the approximation in Fig. 7.1(b). Dotted: MPVL; Dashed: TFMPVL. (b) The absolute values of inner-product* b_{k+m} *in Algorithm 3.2 and the inner-product* $c_{k+m} = \mathbf{u}_{k+m}^H \mathbf{v}_{k+m}$ *in the Two-Sided Lanczos procedure used by MPVL of our version against iteration index* k. *Solid: TFMPVL in Fig. 7.1(a); Dashed: TFMPVL of Fig. 7.1(b); Point: MPVL in 7.1(a); Circle: MPVL in 7.1(b).*

will occur. Of the two issues, deflation is almost guaranteed to occur eventually whereas "look-ahead"-style breakdown occurs only in exceptional cases. For this reason, we choose to devote our discussion to deflation, and reserve our discussion of breakdown for a future paper.

In the following, we will assume no breakdown occurs. Since transpose-free multiple Lanczos procedure, say, Algorithm 3.1, only computes the data $\bar{\mathbf{H}}$ and $\boldsymbol{\psi}_k$, we have to detect through them deflations among the left Lanczos vectors (2.10)

$$\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots$$

or among the right Lanczos vectors (2.10)

$$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots$$

If some $\mathbf{v}_k$ linearly depends on previous $\mathbf{v}$-vectors, then $\mathbf{v}_k$ must be zero and therefore,

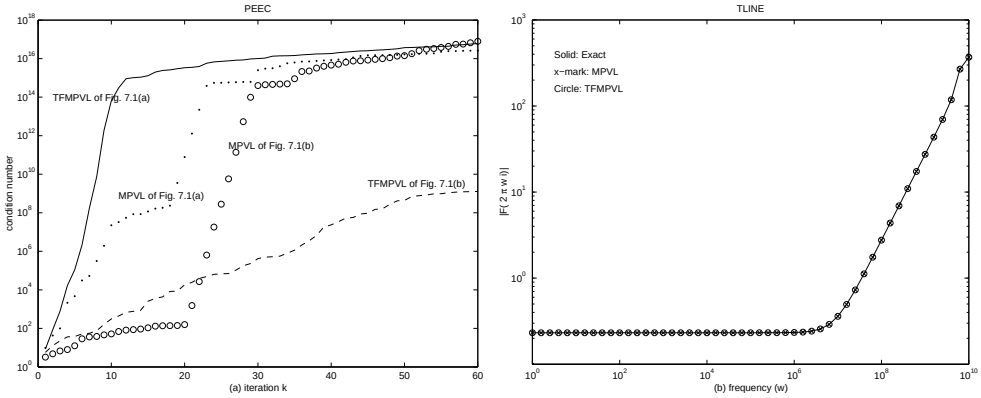

FIG. 7.3. *(a) 2-norm condition numbers of the matrix* $[\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{k+m}]$ *in Algorithm 3.2 and the matrix* $[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k+m}]$ *in the Two-Sided Lanczos procedure used by MPVL of our version against iteration index* k*. Solid: TFMPVL in Fig. 7.1(a); Dashed: TFMPVL of Fig. 7.1(b); Point: MPVL in 7.1(a); Circle: MPVL in 7.1(b). (b) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{30}(\theta)$*.* $\mathbf{f}_{30}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 2\pi 10^9$*. Solid: exact; x-mark: MPVL; Circle: TFMPVL.*

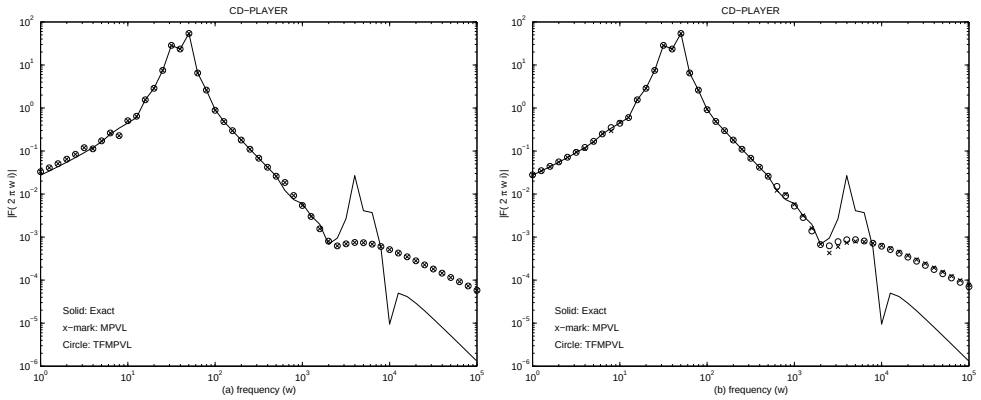

FIG. 7.4. *(a) Graphs of the absolute values of the* $(1,2)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{30}(\theta)$*.* $\mathbf{f}_{30}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^2\pi$*. (b) Graphs of the absolute values of the* $(1,2)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{40}(\theta)$*.* $\mathbf{f}_{40}(\theta)$ *was computed by Augmented TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^2\pi$*,* $m_0 = 0$ *and* $n_0 = 2$*. Solid: exact; x-mark: MPVL; Circle: TFMPVL.*

$\boldsymbol{\psi}_k = \mathbf{A}^{g_n(k)}\mathbf{v}_k = \mathbf{0}$ by (3.1). So, we can tell a deflation among the right Lanczos vectors by looking at whether or not $\boldsymbol{\psi}_k$ is $\mathbf{0}$.

To detect deflations among the left Lanczos vectors, we can look at c_{k+m} in Line 19 of Algorithm 3.1. According to Line 19,

$$c_{k+m} = \widehat{\mathbf{u}}^H_{r_n(k+m)}\boldsymbol{\psi}_{k+m} = \widehat{\mathbf{u}}^H_{r_n(k+m)}\mathbf{A}^{g_n(k+m)}\mathbf{v}_{k+m} = \mathbf{p}^H_{k+m}\mathbf{v}_{k+m}$$

by (3.1) and (2.2). For the vector $\mathbf{p}_{k+m}$, there are two situations in which it can make c_{k+m} zero: (i) $\mathbf{p}_{k+m}$ is linearly dependent on previous $\mathbf{p}$'s (deflation occurs); (ii) $\mathbf{p}_{k+m}$ is independent of previous $\mathbf{p}$'s but still $c_{k+m} = 0$ (breakdown occurs). So, if we assume no breakdown occurs, $c_{k+m} = 0$ will imply that $\mathbf{p}_{k+m}$ depends on previous $\mathbf{p}$'s. Therefore, we can tell a deflation among the left Lanczos vectors by looking at whether or not c_{k+m} is 0.

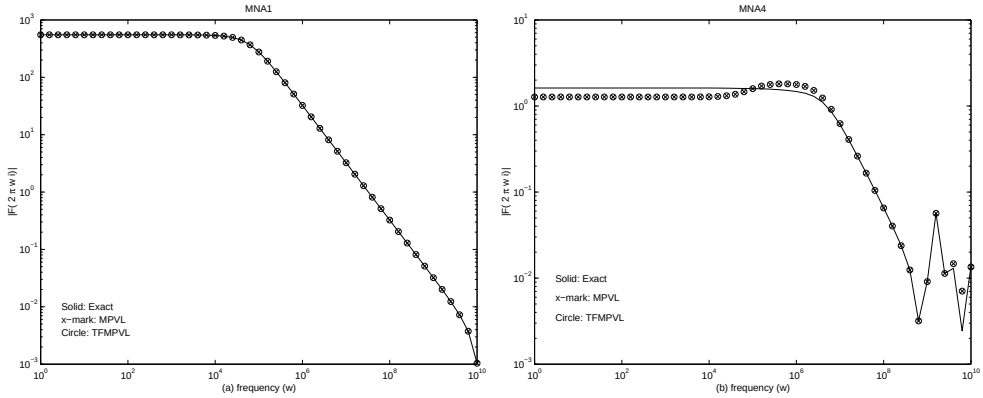

FIG. 7.5. *(a) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{90}(\theta)$. $\mathbf{f}_{90}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^{10}\pi$. *(b) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{40}(\theta)$. $\mathbf{f}_{40}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^{10}\pi$. *Solid: exact; x-mark: MPVL; Circle: TFMPVL.*

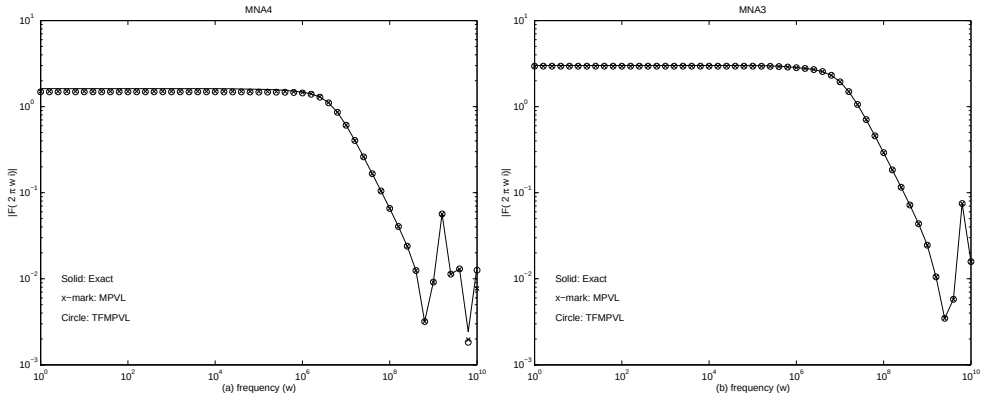

FIG. 7.6. *(a) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{56}(\theta)$. $\mathbf{f}_{56}(\theta)$ *was computed by Augmented TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^{10}\pi$, $m_0 = 0$ *and* $n_0 = 4$. *(b) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{220}(\theta)$. $\mathbf{f}_{220}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^{10}\pi$. *Solid: exact; x-mark: MPVL; Circle: TFMPVL.*

However, the difficulties of incorporating the above deflation idea into Algorithm 3.1 arise from the fact that the bookkeeping becomes difficult when continuing the algorithm after some vectors have been deleted as part of the deflation process. When deflation occurs among the right vectors, the size of the blocks of generated vectors will be reduced, changing all the indexing introduced in section 2. It will become necessary to keep the g and r indices of (2.1) for each generated vector explicitly instead of relying on the formulas (2.1). When deflation occurs on the left, it becomes necessary not only to change the indexing, but it is also necessary to preserve some vectors for future use, just as it is done in the two-sided algorithm of [1]. However, in our transpose-free method, we are not even generating the vectors that would have to be preserved, so we would have to recover the same information using other vectors. At the very least, the result will be Extra matrix-vector multiplications

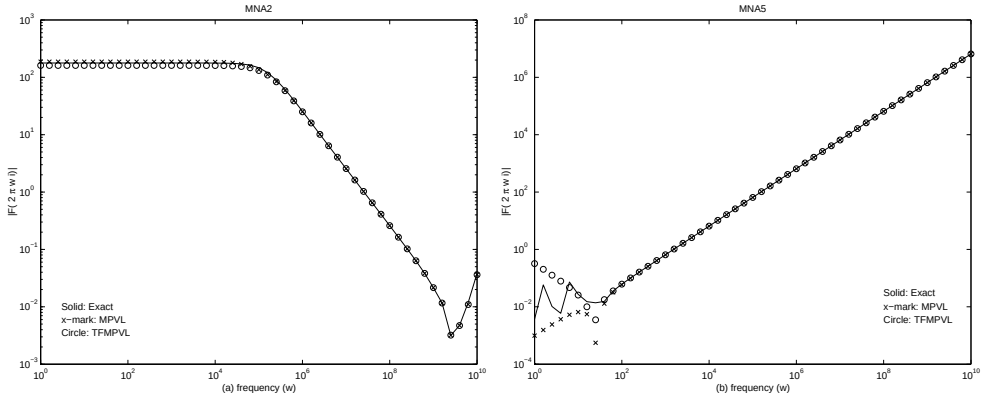

FIG. 7.7. *(a) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{180}(\theta)$. $\mathbf{f}_{180}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^{10}\pi$. *(b) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{90}(\theta)$. $\mathbf{f}_{90}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^{10}\pi$. *Solid: exact; x-mark: MPVL; Circle: TFMPVL.*

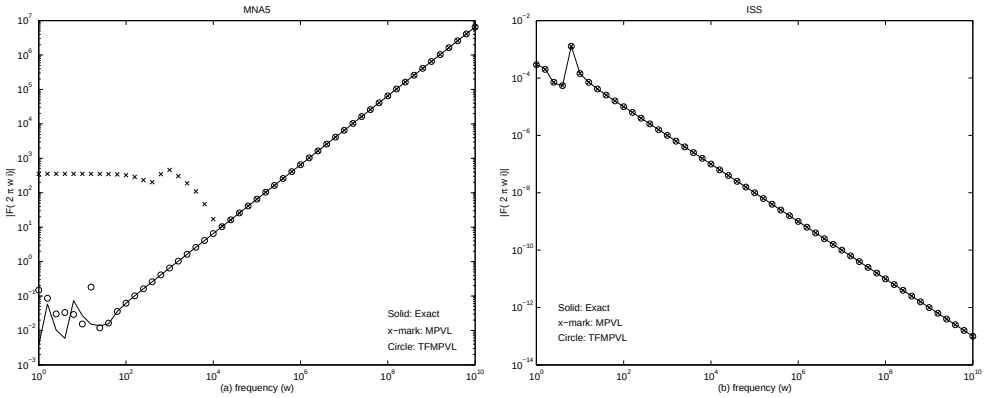

FIG. 7.8. *(a) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{126}(\theta)$. $\mathbf{f}_{126}(\theta)$ *was computed by Augmented TFMPVL and MPVL respectively. Expansion point* $z_0 = 10^{10}\pi$, $m_0 = 0$ *and* $n_0 = 9$. *(b) Graphs of the absolute values of the* $(1,1)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{30}(\theta)$. $\mathbf{f}_{30}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively. Expansion point* $z_0 = 20\pi$. *Solid: exact; x-mark: MPVL; Circle: TFMPVL.*

needed to be handled every time deflation occurs in right or even in left Lanczos vectors. The detailed algorithm to continue the transpose-free Lanczos process in the face of deflation is still under development, even under the assumption of no breakdown.

9. Concluding Remarks. We have proposed a transpose-free version of a Lanczos procedure for multiple starting vectors for the limited case of no deflation and no look-ahead Lanczos process. This version was derived from the Two-Sided Lanczos procedure and includes band Arnoldi procedure as its variant. The method has been illustrated with the problem of computing a Padé approximation to a given transfer function and has resulted in a method called TFMPVL. Besides avoiding the need for explicitly carrying the transpose of the matrix $\mathbf{A}$, TFMPVL reduces the average number of matrix-vector products per iteration from 2 (which is required by the two-

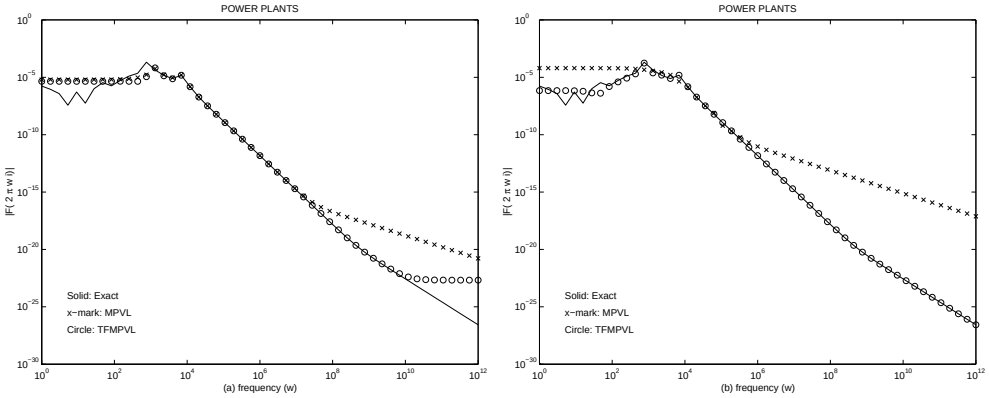

FIG. 7.9. *Graphs of the absolute values of the* $(1,2)$*th elements of* $\mathbf{F}(2\pi wi)$ *and* $\mathbf{f}_{60}(\theta)$*. Expansion point* $z_0 = 2\pi 10^5$*. (a)* $\mathbf{f}_{60}(\theta)$ *was computed by ordinary TFMPVL and MPVL respectively; (b)* $\mathbf{f}_{60}(\theta)$ *was computed by Augmented TFMPVL and MPVL with* $m_0 = 0$ *and* $n_0 = 4$ *respectively. Solid: Exact; x-mark: MPVL; Circle: TFMPVL.*

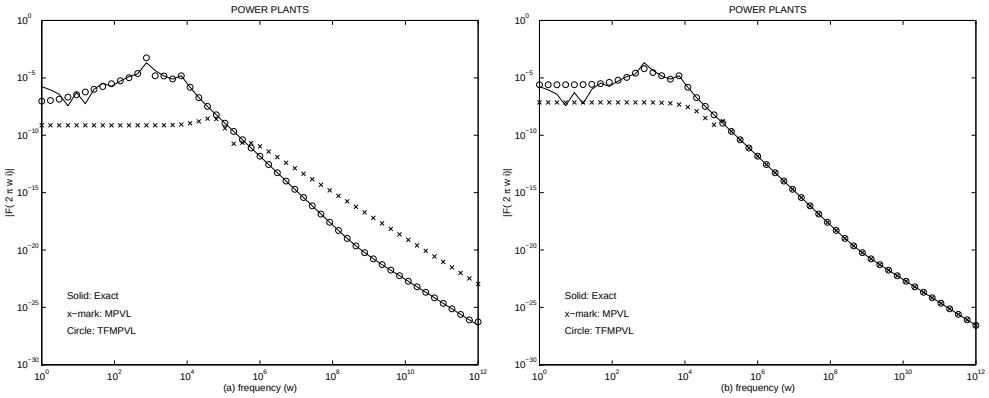

FIG. 7.10. *TFMPVL and MPVL were applied to (7.3). Graphs of the absolute values of the* $(2,1)$*th elements of* $\mathbf{F}(2\pi wi)^H$ *and* $\mathbf{f}_{60}(\theta)$ *are plotted. Expansion point* $z_0 = 2\pi 10^5$*. (a)* $\mathbf{f}_{60}(\theta)$ *was computed by Augmented TFMPVL and MPVL with* $m_0 = 0$ *and* $n_0 = 2$ *respectively; (b)* $\mathbf{f}_{60}(\theta)$ *was computed by Augmented TFMPVL and MPVL with* $m_0 = 0$ *and* $n_0 = 6$ *respectively. Solid: Exact; x-mark: MPVL; Circle: TFMPVL.*

sided MPVL method) to $1+m/n$, where m, n are the number of input, output vectors respectively. In fact, the overall computational cost of TFMPVL is less than that of MPVL (see Table 4.1). Strictly speaking, TFMPVL is a one-sided procedure. It only involves matrix-vector multiplies of the form $\mathbf{Av}$ if applied to (5.2). In the case where computing $\mathbf{A}^H\mathbf{v}$ is much faster than computing $\mathbf{Av}$, one may apply TFMPVL to (7.3) to speed up the overall computation, as we did in Example 3 of §7.

Numerical experiments indicate that, although the TFMPVL method can be less stable than the original two-sided MPVL method in general when $m = n$, its numerical properties can be as favorable as those for MPVL and sometimes even better than MPVL in the case where $m < n$. Moreover, by adding some random vectors to the starting vectors, we may avoid the possible occurrence of *near-breakdown* and *inexact* deflation. We can understand the transpose-free Lanczos method to some extent through the behavior of the TFMPVL method.

In section 8, we gave some discussion of incorporating deflation into the transpose-free algorithm and also some hints at the difficulties when we do so. This variant of this algorithm will be addressed in detail in a future paper.

Acknowledgements. The authors wish to thank Profs. Zhao J. Bai and Peter Benner for their help in setting up the experiments. We would also like to thank Prof. Lothar Reichel as well as the anonymous referees for their valuable comments on an earlier version of this paper.

REFERENCES

[1] J. Aliaga, D. Boley, R. Freund and V. Hernández, *A Lanczos-type method for multiple starting vectors*, Math. Comp. 69 (2000), pp. 1577-1601.

[2] Z. Bai, P. Feldmann and R. Freund, *How to make theoretically passive reduced-order models passive in practice* , in Proceedings of the IEEE 1998 Custom Integrated Circuits Conference, pp. 207-210. IEEE, 1998.

[3] Z. Bai and Q. Ye, *Error estimation of the Padé approximation of transfer functions via the Lanczos process*, Electronic Trans. Numer. Anal. 7, 1-17, 1998.

[4] P. Benner, A. Laub and V. Mehrmann, *A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: continuous-time case*, Technical Report SPC 95_22, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, 1995. Available from *http://www.tu-chemnitz.de/sfb393/spc95pr.html*.

[5] P. Benner, A. Laub and V. Mehrmann, *A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: Discrete-time case*, Technical Report SPC 95_23, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, 1995. Available from *http://www.tu-chemnitz.de/sfb393/spc95pr.html*.

[6] P. Benner, A. Laub and V. Mehrmann, *Benchmarks for the Numerical Solution of Algebraic Riccati Equations*, IEEE Control Systems Magazine, Vol. 7, No. 5, pp. 18-28, 1997.

[7] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel and A. Varga, *SLICOT - A Subroutine Library in Systems and Control Theory*, June 1997, NICONET Report 97-3.

[8] D. L. Boley, *Krylov Space Methods on State-Space Control Models*, Circ. Syst. & Signal Proc. 13 #6, pp 733-758, 1994.

[9] D. L. Boley and G. H. Golub, *The Lanczos-Arnoldi algorithm and controllability*, Systems & Control Letters 4, pp 317-324, 1984.

[10] T. Chan, L. de Pillis and H. van der Vorst, *Transpose-free formulations of Lanczos-type methods for nonsymmetric linear systems*, Numerical Algorithms, 17, pp.51-66, 1998.

[11] P. Feldmann and R. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. Computer-Aided Design 14 (1995), 639-649.

[12] P. Feldmann and R. Freund, *Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm*, in Proceedings of the 32nd Design Automation Conference, pp. 474–479, ACM, San Francisco, 1995.

[13] R. Freund, *A transpose-free quasi-minimum residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470-482.

[14] R. Freund, *Computation of matrix Padé approximations of transfer functions via a Lanczos-type process*, in: Approximation Theory VIII, Vol. 1: Approximation and Interpolation, ed. C. K. Chui and L. L. Schumaker, World Scientific Publishing Co., Inc., Singapore, 1995, 215-222.

[15] R. Freund, *Circuit simulation techniques based on Lanczos-type algorithms*, in Systems and Control in the Twenty-First Century, C. I. Byrnes, B. N. Datta, D. S. Gilliam, C. F. Martin, eds., pp. 171–184, Birkäuser, 1997.

[16] R. Freund, *Computation of matrix-valued formally orthogonal polynomials and applications*, Journal of Computational and Applied Mathematics, 127, pp. 173-199, 2001.

[17] R. Freund, *Model reduction methods based on Krylov subspaces* , Technical report 03/4-01. Available at *http://cm.bell-labs.com/cm/cs/doc/nam.html*.

[18] K. Gallivan, E. Grimme, D. Sorensen and P. Van Dooren, *On some modifications of the Lanczos algorithm and the relation with Padé approximations*, Mathematical Research Series, 7:87–116, 1996.

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, second edition, The Johns Hopkins University Press (Baltimore), 1989.

[20] M. H. Gutknecht, *Variants of BiCGStab for matrices with complex spectrum*, SIAM J. Sci. Comput. 14, 1020-1033, 1993.

[21] M. H. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numer., 6(1997), pp. 271-397.

[22] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J., Matrix Anal. Appl., 13 (1992), pp. 567–593.

[23] A. E. Ruehli, *Equivalent circuit models for three-dimensional multi-conductor systems*, IEEE Trans. Microwave Theory Tech., 22:216-221, 1974.

[24] Y. Saad, *Iterative methods for sparse linear systems*, 2nd edition, SIAM, 2003.

[25] G. L. G. Sleijpen and D. R. Fokkema, *BiCGSTAB(k) for linear equations involving unsymmetric matrices with complex spectrum*, Electronic Trans. Numer. Anal. 1, 11-32, 1993.

[26] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 10(1):36-52, 1989.

[27] M. Steinbuch, P. J. M. Van Groos, G. Schootstra, P. M. R. Wortelboer and O. H. Bosgra, *μ-synthesis for a compact disc player*, International Journal of Robust and Nonlinear Control, 8(2):169-189, Feb 1998.

[28] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 12:631-644, 1992.

[29] M. Yeung and T. Chan, *ML(k)BiCGSTAB: A BiCGSTAB Variant Based on Multiple Lanczos Starting Vectors*, SIAM J. Sci. Comput., Vol. 21, No. 4, pp. 1263-1290, 1999.