Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 02-031

Transpose-free Multiple Lanczos and Its Application in Pade‘ Approximation

M. Yeung and Daniel Boley

September 13, 2002

Transpose-free Multiple Lanczos and Its Application in Padé Approximation

M. Yeung* and D. 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.
AMS subject classifications: Primary 65F15; Secondary 65G05.

1 Introduction

Recently, Aliaga *et al.*[1] 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 multi-input multi-output (MIMO) Lanczos algorithm or a MIMO Lanczos procedure. Given a square matrix $\mathbf{A} \in \mathcal{C}^{N\times N}$ and two blocks $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ of left and right starting vectors, the MIMO Lanczos algorithm employs matrix-vector products involving both $\mathbf{A}$ and $\mathbf{A}^H$, and generates (i) block tridiagonal matrices $\bar{\mathbf{H}}$ and $\tilde{\mathbf{H}}$ that constitute approximations to the matrices $\mathbf{A}$ and $\mathbf{A}^H$ respectively; (ii) two sequences of biorthogonal basis vectors for the left and right block Krylov subspaces induced by the given data. 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 MIMO Lanczos procedure has received applications in varieties 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

*Department of Mathematics, P.O. Box 3036, 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.

†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 9811229.

time-invariant linear dynamical systems, the MPVL method (see [7, 9], for instance) has been proposed to compute Padé approximation of transfer functions of the form

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

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

In this paper, we propose a transpose-free version of the MIMO Lanczos procedure which computes the block tridiagonal matrix $\bar{\mathbf{H}}$ and a sequence of basis vectors for the right block Krylov subspace without accessing $\mathbf{A}^H$. It is well known that the classical Lanczos process is intimately related to bi-conjugate gradient (BiCG) method for solving nonsymmetric systems of linear equations [13, 18]. Transpose-free versions, e.g., CGS [20], BiCGSTAB [22] for single starting vectors and ML(k)BiCGSTAB [23] 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 is labeled Transpose-free Multiple Lanczos Procedure (TFMLP). In our discussion, we assume for simplicity that no deflation or look-ahead steps occur in the MIMO Lanczos procedure, so TFMLP is actually a transpose-free version of the limited MIMO method.

The MPVL method is an application of the MIMO Lanczos procedure to Padé approximation of transfer functions (1). Correspondingly, a transpose-free MPVL method (TFMPVL) will be developed from the TFMLP and in the process can actually reduce the total number of matrix-vector products necessary. Recall that, however, the transpose-free algorithms in the context of linear systems of equations are in general less stable than the two-size algorithms though ML(k)BiCGSTAB can improve the stability by increasing the number k of left starting vectors. Not unexpectedly, TFMPVL is less stable than MPVL and the advantages and disadvantages of the transpose-free algorithms that were found in the context of systems of linear equations will be carried also in the context of Padé approximants.

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. [4, 10, 12] and references therein. We will give some specifics of this connection relevant to this paper in §4 after we have introduced the TFMLP method.

The rest of this paper is organized as follows. In §2 we review the two-sided MIMO Lanczos algorithm and introduce our notation, in §3 we derive our transpose-free MIMO Lanczos procedure, in §4 and §5 we show how to use this Lanczos procedure to compute the Padé approximant, and in §6 we illustrate the methods with some numerical experiments. We finally conclude the paper in §7.

2 Lanczos Procedure for Multiple Starting Vectors

Aliaga, Boley, Freund and Hernández (ABFH) [1] recently developed a MIMO Lanczos-type procedure that handles multiple starting vectors. Let $\mathbf{A} \in \mathcal{C}^{N\times N}$ and let n left starting vectors $\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \cdots, \hat{\mathbf{u}}_n \in \mathcal{C}^N$ and m right starting vectors $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \cdots, \hat{\mathbf{v}}_m \in \mathcal{C}^N$ be given. Define index functions

$$g_n(k) = \lfloor (k-1)/n \rfloor\,, \qquad r_n(k) = k - n\,g_n(k)\,,$$

$$g_m(k) = \lfloor (k-1)/m \rfloor\,, \qquad r_m(k) = k - m\,g_m(k)\,,$$

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

$$\text{(a)} \quad \mathbf{p}_k = \left(\mathbf{A}^H\right)^{g_n(k)} \hat{\mathbf{u}}_{r_n(k)}\,, \qquad \text{(b)} \quad \mathbf{q}_k = \mathbf{A}^{g_m(k)} \hat{\mathbf{v}}_{r_m(k)}\,. \tag{2}$$

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.

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

$$\begin{aligned} &\mathbf{u}_{k'} \in \mathcal{G}_{k'}\left(\mathbf{A}^H, \hat{\mathbf{U}}\right) \quad \text{and} \quad \mathbf{u}_{k'} \perp \mathcal{G}_{k'-1}\left(\mathbf{A}, \hat{\mathbf{V}}\right), \\ &\mathbf{v}_k \in \mathcal{G}_k\left(\mathbf{A}, \hat{\mathbf{V}}\right) \quad \text{and} \quad \mathbf{v}_k \perp \mathcal{G}_{k-1}\left(\mathbf{A}^H, \hat{\mathbf{U}}\right), \end{aligned} \tag{3}$$

where $\mathcal{G}_{k'}\left(\mathbf{A}^H, \hat{\mathbf{U}}\right) \stackrel{\text{def}}{=} span\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{k'}\}$ and $\mathcal{G}_k\left(\mathbf{A}, \hat{\mathbf{V}}\right) \stackrel{\text{def}}{=} span\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_k\}$. The subspaces $\mathcal{G}_{k'}\left(\mathbf{A}^H, \hat{\mathbf{U}}\right)$ and $\mathcal{G}_k\left(\mathbf{A}, \hat{\mathbf{V}}\right)$ 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, 7, 11], 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

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

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 (3). Moreover,*

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

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

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

$$\mathbf{v}_k = \gamma_1^{(k)}\mathbf{q}_1 + \gamma_2^{(k)}\mathbf{q}_2 + \cdots + \gamma_{k-1}^{(k)}\mathbf{q}_{k-1} + \mathbf{q}_k, \tag{4}$$

then (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 (4) satisfies $\mathbf{v}_k \not\perp \mathbf{p}_k$ for $k \leq \nu$ 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 (4) 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,

$$\mathbf{v}_k = \mathbf{A}\mathbf{v}_{k-m} - \bar{h}_{k-1}^{(k-m)}\mathbf{v}_{k-1} - \bar{h}_{k-2}^{(k-m)}\mathbf{v}_{k-2} - \cdots - \bar{h}_1^{(k-m)}\mathbf{v}_1\,, \tag{5}$$

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*

$$\mathbf{v}_k = \mathbf{A}\mathbf{v}_{k-m} - \bar{h}_{k-1}^{(k-m)}\mathbf{v}_{k-1} - \bar{h}_{k-2}^{(k-m)}\mathbf{v}_{k-2} - \cdots - \bar{h}_{\bar{m}_{k-m}}^{(k-m)}\mathbf{v}_{\bar{m}_{k-m}} \tag{6}$$

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 (5) 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 (6) can be written in matrix form as

$$\mathbf{A}\mathbf{V}_{\nu-m} = \mathbf{V}_\nu\bar{\mathbf{H}}\,. \tag{7}$$

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

Theorem 2.3 *Let vectors $\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \cdots, \hat{\mathbf{u}}_n, \hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \cdots, \hat{\mathbf{v}}_m$ be given starting vectors and let $\mathbf{p}_k$'s, $\mathbf{q}_k$'s, $\mathcal{G}_{k'}\left(\mathbf{A}^H, \hat{\mathbf{U}}\right)$, $\mathcal{G}_k\left(\mathbf{A}, \hat{\mathbf{V}}\right)$ and $\mathbf{W}_\nu$ be as defined above. 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 $\bar{\mathbf{H}}$, which has an upper bandwidth n and a lower bandwidth m and all the entries $\bar{h}_{j+m,j} = 1$ in its lowest subdiagonal, and an $\nu \times (\nu-n)$ band matrix $\tilde{\mathbf{H}}$, which has an upper bandwidth m and a lower bandwidth n and all the entries $\tilde{h}_{j+n,j} = 1$ in its lowest subdiagonal, such that*

$$span\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{k'}\} = \mathcal{G}_{k'}\left(\mathbf{A}^H, \hat{\mathbf{U}}\right), \qquad span\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k\} = \mathcal{G}_k\left(\mathbf{A}, \hat{\mathbf{V}}\right) \tag{8}$$

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,

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

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 MIMO Lanczos procedure is a procedure which computes the quantities $\mathbf{U}_\nu, \mathbf{V}_\nu, \tilde{\mathbf{H}}$ and $\bar{\mathbf{H}}$ in Theorem 2.3.

3 Transpose-free Lanczos Procedure for ultiple Starting Vectors

The implementation of the MIMO 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, [5, 8, 13, 14, 15, 18, 19, 20, 22, 23], 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" MIMO Lanczos procedure which computes the $\bar{\mathbf{H}}$ and $\mathbf{V}_\nu$ 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 MIMO 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 (6),

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

where $k = 1, 2, \cdots, \nu - m$. Because of the property (3), the coefficients $\bar{h}^{(k)}_i$ are determined by

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

Thus, we have the following procedure to compute the matrices $\bar{\mathbf{H}}$ and $\mathbf{V}_\nu$.

Lanczos Procedure Version 1.

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

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. Moreover, recall that we have assumed $m \leq n$. Therefore, $g_n(k) = 0$ and $r_n(k) = k$ 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 dot 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 $\mathbf{v}_k \perp span\{\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\hat{\mathbf{v}}_1, \cdots, \hat{\mathbf{v}}_k\}$.
2 Compute $\mathbf{p}_k = \hat{\mathbf{u}}_k$ and $c_k = \mathbf{p}_k^H\mathbf{v}_k$ for $k = 1, \ldots, m$.

3. For $k = 1, 2, \cdots, \nu - m$
4. $\quad \bar{m}_k = max\{k - n, 1\}$;
5. $\quad \mathbf{v}_{tmp} = \mathbf{A}\mathbf{v}_k$;
6. $\quad$ For $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
7. $\quad\quad \bar{h}_i^{(k)} = \mathbf{p}_i^H \mathbf{v}_{tmp}/c_i$;
8. $\quad\quad \mathbf{v}_{tmp} = \mathbf{v}_{tmp} - \bar{h}_i^{(k)} \mathbf{v}_i$;
9. $\quad$ End
10. $\quad \bar{h}_{k+m}^{(k)} = 1$;
11. $\quad \mathbf{v}_{k+m} = \mathbf{v}_{tmp}$;
12. $\quad \mathbf{p}_{k+m} = (\mathbf{A}^H)^{g_n(k+m)} \hat{\mathbf{u}}_{r_n(k+m)}$;
13. $\quad c_{k+m} = \mathbf{p}_{k+m}^H \mathbf{v}_{k+m}$;
14. End

Based on the above Lanczos version, we are now ready to give a transpose-free procedure to compute $\bar{\mathbf{H}}$ and $\mathbf{V}_\nu$. In order to remove the $\mathbf{A}^H$ which is used to calculate $\mathbf{p}_{k+m}$ in Line 12, we introduce an auxiliary vector $\boldsymbol{\psi}_k$ defined by

$$\boldsymbol{\psi}_k = \mathbf{A}^{g_n(k)} \mathbf{v}_k \tag{9}$$

for $k = 1, 2, \ldots$. With (9) and recall that $\mathbf{p}_k = (\mathbf{A}^H)^{g_n(k)} \hat{\mathbf{u}}_{r_n(k)}$ by (2a), Version 2 can then be reformulated as follows.

Lanczos Procedure Version 3.

1. Compute vectors $\{\mathbf{v}_k\}_{k=1}^m$ such that $\mathbf{v}_k \perp span\{\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\hat{\mathbf{v}}_1, \cdots, \hat{\mathbf{v}}_k\}$.
2. Compute $\mathbf{p}_k = \hat{\mathbf{u}}_k$ for $k = 1, 2, \ldots, m$.
3. Set $\boldsymbol{\psi}_k = \mathbf{v}_k$ and compute $c_k = \mathbf{p}_k^H \mathbf{v}_k$ for $k = 1, 2, \ldots, m$.
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. $\quad \mathbf{A}^{g_n(\bar{m}_k)} \mathbf{v}_{tmp} = \mathbf{A}^{1+g_n(\bar{m}_k)-g_n(k)} \boldsymbol{\psi}_k$;
8. $\quad$ For $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
9. $\quad\quad \bar{h}_i^{(k)} = \hat{\mathbf{u}}_{r(i)}^H \mathbf{A}^{g_n(i)} \mathbf{v}_{tmp} \,/\, c_i$;
10. $\quad\quad \mathbf{v}_{tmp} = \mathbf{v}_{tmp} - \bar{h}_i^{(k)} \mathbf{v}_i$;
11. $\quad\quad \mathbf{A}^{g_n(i)} \mathbf{v}_{tmp} = \mathbf{A}^{g_n(i)} \mathbf{v}_{tmp} - \bar{h}_i^{(k)} \boldsymbol{\psi}_i$;
12. $\quad$ End
13. $\quad \bar{h}_{k+m}^{(k)} = 1$;
14. $\quad \mathbf{v}_{k+m} = \mathbf{v}_{tmp}$;
15. $\quad \boldsymbol{\psi}_{k+m} = \mathbf{A}^{g_n(k+m)} \mathbf{v}_{tmp}$;
16. $\quad \mathbf{p}_{k+m} = (\mathbf{A}^H)^{g_n(k+m)} \hat{\mathbf{u}}_{r(k+m)}$;
17. $\quad c_{k+m} = \hat{\mathbf{u}}_{r(k+m)}^H \boldsymbol{\psi}_{k+m}$;
18. End

Our goal is to compute $\bar{\mathbf{H}}$ and $\mathbf{V}_\nu$. In Version 3, Lines 2 and 16 compute the vectors $\mathbf{p}_k$ which are not required in finding these quantities. Since these two lines are irrelevant to the remaining part of the procedure, we can delete them from the version without changing the implementation.

We now set $grade = g_n(\bar{m}_k)$ and $\mathbf{e}_{tmp} = \mathbf{A}^{g_n(i)} \mathbf{v}_{tmp}$ in Version 3. Note that g_n is a non-decreasing step function with step size n. So, the value of $g_n(i)$ increases by 1 every n iterations in

i (Lines 8 - 12). Moreover, two new vectors $\mathbf{v}_{k+m}$ (in Line 14) and $\boldsymbol{\psi}_{k+m}$ (in Line 15) are generated in each k-iteration. Since the vectors $\mathbf{v}_{k+m}$ and $\boldsymbol{\psi}_{k+m}$ are computed at the end of the k-loop, we can store the values of $\mathbf{v}_{tmp}$ and $\mathbf{e}_{tmp}$ in the spaces occupied by them respectively to save some memory. Thus, we arrive at the following final version of our transpose-free Lanczos procedure which produces the matrices $\bar{\mathbf{H}}$ and $\mathbf{V}_\nu$.

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

1. *Compute vectors $\{\mathbf{v}_k\}_{k=1}^m$ such that $\mathbf{v}_k \perp span\{\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\hat{\mathbf{v}}_1, \cdots, \hat{\mathbf{v}}_k\}$.*
2. *For $k = 1, 2, \ldots, m$, do: set $\boldsymbol{\psi}_k = \mathbf{v}_k$ and compute $c_k = \hat{\mathbf{u}}_k^H \mathbf{v}_k$.*
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. $\quad \mathbf{v}_{k+m} = \mathbf{A}\mathbf{v}_k$;
7. $\quad \boldsymbol{\psi}_{k+m} = \mathbf{A}^{1+grade-g_n(k)}\boldsymbol{\psi}_k$;
8. $\quad$ *For* $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
9. $\quad\quad$ *If* $g_n(i) > grade$
10. $\quad\quad\quad \boldsymbol{\psi}_{k+m} = \mathbf{A}\boldsymbol{\psi}_{k+m}$;
11. $\quad\quad\quad grade = grade + 1$;
12. $\quad\quad$ *End*
13. $\quad\quad \bar{h}_i^{(k)} = \hat{\mathbf{u}}_{r(i)}^H \boldsymbol{\psi}_{k+m} \,/\, c_i$;
14. $\quad\quad \mathbf{v}_{k+m} = \mathbf{v}_{k+m} - \bar{h}_i^{(k)} \mathbf{v}_i$;
15. $\quad\quad \boldsymbol{\psi}_{k+m} = \boldsymbol{\psi}_{k+m} - \bar{h}_i^{(k)} \boldsymbol{\psi}_i$;
16. $\quad$ *End*
17. $\quad \bar{h}_{k+m}^{(k)} = 1$;
18. $\quad$ *If* $g_n(k + m) > grade$
19. $\quad\quad \boldsymbol{\psi}_{k+m} = \mathbf{A}\boldsymbol{\psi}_{k+m}$;
20. $\quad$ *End*
21. $\quad c_{k+m} = \hat{\mathbf{u}}_{r(k+m)}^H \boldsymbol{\psi}_{k+m}$;
22. *End*

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 $\bar{\mathbf{H}}$ computed in the algorithm can become very ill-conditioning. 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 any nonsingular diagonal matrix and let

$$\hat{\mathbf{H}} = \boldsymbol{\Lambda}_\nu \, \bar{\mathbf{H}} \, \boldsymbol{\Lambda}_{\nu-m}^{-1} \qquad \text{and} \qquad \tilde{\mathbf{V}}_\nu = \mathbf{V}_\nu \boldsymbol{\Lambda}_\nu^{-1}. \tag{10}$$

Because of equation (7), $\hat{\mathbf{H}}$ and $\tilde{\mathbf{V}}_\nu$ are related by

$$\mathbf{A}\tilde{\mathbf{V}}_{\nu-m} = \tilde{\mathbf{V}}_\nu \hat{\mathbf{H}} \, .$$

A typical choice of $\boldsymbol{\Lambda}_\nu$ is to set $\lambda_k = \|\boldsymbol{\psi}_k\|_2$ for $k = 1, 2, \ldots$. This choice is equivalent to normalize the vectors $\boldsymbol{\psi}_k$ in each k-loop of computation in Algorithm 3.1 (see the definition of $\boldsymbol{\phi}_k$ in (11) below).

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

$$\begin{aligned} \hat{h}_i^{(k)} &\stackrel{\text{def}}{=} \lambda_i \bar{h}_i^{(k)} \lambda_k^{-1}, & \boldsymbol{\phi}_k &\stackrel{\text{def}}{=} \boldsymbol{\psi}_k/\lambda_k, \\ \tilde{\mathbf{v}}_k &= \mathbf{v}_k/\lambda_k, & b_k &\stackrel{\text{def}}{=} c_k/\lambda_k. \end{aligned} \tag{11}$$

With these new definitions, Algorithm 3.1 becomes

Algorithm 3.2 Scaled Transpose-free Multiple Lanczos Procedure *Given m right starting vectors $\{\hat{\mathbf{v}}_k\}_{k=1}^m$ and n left starting vectors $\{\hat{\mathbf{u}}_{k'}\}_{k'=1}^n$ with $m \le n$. Suppose the assumption of Theorem 2.3 holds. The following algorithm computes the $\nu \times (\nu - m)$ band matrix $\hat{\mathbf{H}} = (\hat{h}_{ij})$ with $\hat{h}_{ij} = \hat{h}_i^{(j)}$ and the matrix $\tilde{\mathbf{V}}_\nu$ described in equation (10).*

1. *Compute vectors $\{\mathbf{v}_k\}_{k=1}^m$ such that $\mathbf{v}_k \perp span\{\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_{k-1}\}$ and $\mathbf{v}_k \in span\{\hat{\mathbf{v}}_1, \cdots, \hat{\mathbf{v}}_k\}$.*
2. *For $k = 1, 2, \ldots, m$, do: define λ_k, set $\boldsymbol{\phi}_k = \mathbf{v}_k \,/\, \lambda_k$ and $\tilde{\mathbf{v}}_k = \boldsymbol{\phi}_k$, and compute $b_k = \hat{\mathbf{u}}_k^H \boldsymbol{\phi}_k$.*
3. *For $k = 1, 2, 3, \ldots$*
4. $\quad \bar{m}_k = max\{k - n, 1\};$
5. $\quad grade = g_n(\bar{m}_k);$
6. $\quad \tilde{\mathbf{v}}_{k+m} = \mathbf{A}\tilde{\mathbf{v}}_k;$
7. $\quad \boldsymbol{\phi}_{k+m} = \mathbf{A}^{1+grade-g_n(k)}\boldsymbol{\phi}_k;$
8. $\quad$ *For* $i = \bar{m}_k, \bar{m}_k + 1, \cdots, k + m - 1$
9. $\quad\quad$ *If* $g_n(i) > grade$
10. $\quad\quad\quad \boldsymbol{\phi}_{k+m} = \mathbf{A}\boldsymbol{\phi}_{k+m};$
11. $\quad\quad\quad grade = grade + 1;$
12. $\quad\quad$ *End*
13. $\quad\quad \hat{h}_i^{(k)} = \hat{\mathbf{u}}_{r(i)}^H \boldsymbol{\phi}_{k+m} \,/\, b_i;$
14. $\quad\quad \tilde{\mathbf{v}}_{k+m} = \tilde{\mathbf{v}}_{k+m} - \hat{h}_i^{(k)} \tilde{\mathbf{v}}_i;$
15. $\quad\quad \boldsymbol{\phi}_{k+m} = \boldsymbol{\phi}_{k+m} - \hat{h}_i^{(k)} \boldsymbol{\phi}_i;$
16. $\quad$ *End*
17. $\quad$ *If* $g_n(k + m) > grade$
18. $\quad\quad \boldsymbol{\phi}_{k+m} = \mathbf{A}\boldsymbol{\phi}_{k+m};$
19. $\quad$ *End*
20. $\quad$ *Define* $\hat{h}_{k+m}^{(k)};$
21. $\quad \tilde{\mathbf{v}}_{k+m} = \tilde{\mathbf{v}}_{k+m} \,/\, \hat{h}_{k+m}^{(k)};$
22. $\quad \boldsymbol{\phi}_{k+m} = \boldsymbol{\phi}_{k+m} \,/\, \hat{h}_{k+m}^{(k)};$
23. $\quad b_{k+m} = \hat{\mathbf{u}}_{r(k+m)}^H \boldsymbol{\phi}_{k+m};$
24. *End*

We remark that (a) the λ's and $\hat{h}$'s in Lines 2 and 20 of Algorithm 3.2 can be assigned any nonzero numbers; (b) Lines 4 - 23 compute the entries of $\hat{\mathbf{H}}$ in the k-th column. The k-th column of $\hat{\mathbf{H}}$ is related to the k-th column of $\bar{\mathbf{H}}$ by

$$[0, \cdots, 0, \hat{h}_{k-n}^{(k)}, \cdots, \hat{h}_{k+m}^{(k)}, 0, \cdots, 0]^T = [0, \cdots, 0, \lambda_{k-n}\bar{h}_{k-n}^{(k)}\lambda_k^{-1}, \cdots, \lambda_{k+m}\bar{h}_{k+m}^{(k)}\lambda_k^{-1}, 0, \cdots, 0]^T$$

according to (10), 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 $j > m$, $\lambda_j = \hat{h}_j^{(j-m)}\lambda_{j-m}$ is fixed by the choices made in line 20 during successive steps.

Item	*Average count Algorithm 3.2*	*Average count Lanczos Version 2*
Matrix vector product	$2 + \frac{m}{n}$	2
Saxpy	$2(m+n)$	$m+n$
Scalar-vector product	2	0
Dot product	$m+n+1$	$m+n+1$
Storage beyond $\mathbf{A}$, $\hat{\mathbf{H}}$	$(2m+3n+2)N+m+n$	$(2m+3n+1)N+m+n$

Table 1: Average cost per k-loop of Algorithm 3.2 and its total storage requirement, compared to the Lanczos Procedure Version 2.

Regarding the computational cost of Algorithm 3.2, we list the average cost per k-iteration in Table 1. Note that the power $1 + grade - g_n(k)$ of $\mathbf{A}$ in Line 7 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 6, 10 and 18. In each pass through the k-loop (Lines 4 - 23), the multiplication by $\mathbf{A}$ is guaranteed to occur twice and sometimes thrice, leading to the average cost estimate of $2 + m/n$ given in Table 1. About storage, the data $\{\mathbf{v}_{k-n}, \cdots, \mathbf{v}_{k+m}\}$, $\{\boldsymbol{\phi}_{k-n}, \cdots, \boldsymbol{\phi}_{k+m}\}$, $\{\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_n\}$ 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 $\hat{\mathbf{H}}$.

There are many variants of the traditional Lanczos method, so we compare the costs for Algorithm 3.2 with that estimated from the Lanczos Procedure Version 2. Of course, Version 2 is not a version that one would actually use for computation, but its costs and storage requirements approximate that of the MIMO method when no deflation or look-ahead occur.

4 A Transpose-free Version of the PVL ethod

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, [6, 7, 9, 10].

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:

$$\mathbf{Y}(z) = \mathbf{L}^H(z\mathbf{C} + \mathbf{G})^{-1}\mathbf{R} \cdot \mathbf{W}(z) \equiv \mathbf{F}(z) \cdot \mathbf{W}(z). \tag{12}$$

To compute $\mathbf{F}(z)$, write $z = z_0 + \theta$. Then

$$\begin{aligned}
\mathbf{F}(z) &= \mathbf{L}^H(z\mathbf{C}+\mathbf{G})^{-1}\mathbf{R} \\
&= \mathbf{L}^H(z_0\mathbf{C}+\mathbf{G}+\theta\mathbf{C})^{-1}\mathbf{R} \\
&= \mathbf{L}^H(\mathbf{I}+\theta(z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{C})^{-1}(z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{R} \\
&= \hat{\mathbf{U}}^H(\mathbf{I}-\theta\mathbf{A})^{-1}\hat{\mathbf{V}} \\
&= \sum_{k=0}^{\infty}\mathbf{M}_k\theta^k,
\end{aligned} \tag{13}$$

where

$$\mathbf{A} = -(z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{C}, \quad \hat{\mathbf{U}} = \mathbf{L}, \quad \hat{\mathbf{V}} = (z_0\mathbf{C}+\mathbf{G})^{-1}\mathbf{R}, \quad \mathbf{M}_k = \hat{\mathbf{U}}^H\mathbf{A}^k\hat{\mathbf{V}}, \tag{14}$$

and where we show the expansion of the transfer function $\mathbf{F}(z)$ in a power series about $z = z_0$. The $\mathbf{M}_k$'s are often called the *moments* or *Markov parameters*. Our goal is to seek a new lower order system

$$\frac{d\breve{\mathbf{x}}}{dt} = \breve{\mathbf{A}}\breve{\mathbf{x}} + \breve{\mathbf{V}}\mathbf{w}(t), \qquad \breve{\mathbf{y}}(t) = \breve{\mathbf{U}}^H\breve{\mathbf{x}}(t) \tag{15}$$

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 (12). Specifically, we seek a Padé approximant, which is a lower order system for which a number of the first moments $\breve{\mathbf{M}}_k$ agree with the original $\mathbf{M}_k$.

Definition 4.1 *An l-th Padé approximant $\mathbf{f}_l(\theta)$ of $\mathbf{F}(\theta+z_0)$ is defined to be a function of the form*

$$\mathbf{f}_l(\theta) = \breve{\mathbf{U}}^H(\mathbf{I}-\theta\breve{\mathbf{A}})^{-1}\breve{\mathbf{V}} \tag{16}$$

whose Taylor expansion about $\theta = 0$ matches as many leading terms of the Taylor expansion (13) 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, [9, 10].

With the given blocks $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ of (14) as the n left starting vectors and m right starting vectors respectively in the MIMO Lanczos procedure, we obtain data $\mathbf{U}_\nu, \mathbf{V}_\nu, \bar{\mathbf{H}}$ and $\tilde{\mathbf{H}}$. Because of (8), there exist matrices $\boldsymbol{\eta} \in \mathcal{C}^{n\times n}$ and $\boldsymbol{\rho} \in \mathcal{C}^{m\times m}$ such that

$$\hat{\mathbf{U}} = \mathbf{U}_n\boldsymbol{\eta} \qquad \text{and} \qquad \hat{\mathbf{V}} = \mathbf{V}_m\boldsymbol{\rho}. \tag{17}$$

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

$$\mathbf{D}_k = \mathbf{U}_k^H\mathbf{V}_k\,. \tag{18}$$

Then the following theorem [9, 10] provides us an l-th Padé approximant.

Theorem 4.2 *Let* $\max\{m,n\} \le l$. *Then,*

$$\mathbf{f}_l(\theta) = \begin{bmatrix} \mathbf{D}_n^H\boldsymbol{\eta} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H (\mathbf{I}-\theta\bar{\mathbf{H}}_l)^{-1}\begin{bmatrix} \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix} \tag{19}$$

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 [7, 9], the MPVL method was proposed to compute $\mathbf{f}_l(\theta)$ based on Theorem 4.2 using the two-sided MIMO Lanczos algorithm. The MPVL method consists of two steps: (a) the MIMO Lanczos procedure is run for the first l steps to obtain data $\bar{\mathbf{H}}_l, \mathbf{D}_n, \boldsymbol{\eta}$ and $\boldsymbol{\rho}$, then (b) the Lanczos-Padé connection (19) then yields coefficient matrices for the reduced order linear dynamical system (15),

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

whose transfer function is exactly the l-th Padé approximant $\mathbf{f}_l(\theta)$ defined by (19) to the transfer function $\mathbf{F}(\theta + z_0)$.

Observe that the MIMO Lanczos procedure generates not only the data $\bar{\mathbf{H}}_l, \mathbf{D}_n, \boldsymbol{\eta}, \boldsymbol{\rho}$ but $\mathbf{U}_l, \mathbf{V}_l, \tilde{\mathbf{H}}_l$ as well. However, the data $\mathbf{U}_l, \mathbf{V}_l$ and $\tilde{\mathbf{H}}_l$ do not contribute directly to compute $\mathbf{f}_l(\theta)$ in (19). 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 MIMO Lanczos procedure to bypass the computations of $\mathbf{U}_l, \mathbf{V}_l$ and $\tilde{\mathbf{H}}_l$ and still generate the quantities that are related to (19). 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 (19) in terms of the quantities computed by Algorithm 3.2. Since $\hat{\mathbf{U}} = \mathbf{U}_n \boldsymbol{\eta}$ and $\mathbf{D}_n = \mathbf{U}_n^H \mathbf{V}_n$ from (17) and (18), we have

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

If we let $\hat{\mathbf{H}}_l$ and $\boldsymbol{\Lambda}_l$ denote the $l \times l$ principal blocks of the matrices $\hat{\mathbf{H}}$ and $\boldsymbol{\Lambda}_\nu$ respectively, then we have $\hat{\mathbf{H}}_l = \boldsymbol{\Lambda}_l \bar{\mathbf{H}}_l \boldsymbol{\Lambda}_l^{-1}$ from (10) and therefore

$$\begin{aligned} \mathbf{f}_l(\theta) \;&=\; \begin{bmatrix} \mathbf{V}_n^H \hat{\mathbf{U}} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I} - \theta \boldsymbol{\Lambda}_l^{-1} \hat{\mathbf{H}}_l \boldsymbol{\Lambda}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix} \\ &=\; \begin{bmatrix} \boldsymbol{\Lambda}_n^{-1} \mathbf{V}_n^H \hat{\mathbf{U}} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I} - \theta \hat{\mathbf{H}}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\Lambda}_m \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix} \qquad (20) \\ &=\; \begin{bmatrix} \boldsymbol{\Omega}_n^H \hat{\mathbf{U}} \\ \mathbf{0}_{(l-n)\times n} \end{bmatrix}^H \left(\mathbf{I} - \theta \hat{\mathbf{H}}_l\right)^{-1} \begin{bmatrix} \boldsymbol{\Lambda}_m \boldsymbol{\rho} \\ \mathbf{0}_{(l-m)\times m} \end{bmatrix}, \end{aligned}$$

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 (9) and (11).

We are now ready to present a transpose-free implementation of the MPVL method in the following algorithm. Regarding the $m \times m$ matrix $\boldsymbol{\rho}$, it can be computed via a modified two-sided Gram-Schmidt-type process [7, 16].

Algorithm 4.3 Transpose-free MPVL (TFMPVL) *Given m right starting vectors $\{\hat{\mathbf{v}}_k\}_{k=1}^m$ and n left starting vectors $\{\hat{\mathbf{u}}_{k'}\}_{k'=1}^n$ 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 4.2.*

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 $\hat{\mathbf{H}}_l$.*
3. *Compute $\mathbf{f}_l(\theta)$ according to (20).*

Item	*Average count Algorithm 3.2*	*Item*	*Average count Algorithm 3.2*
Matrix vector product	$1+\frac{m}{n}$	Dot product	$m+n+1$
Saxpy	$m+n$	Scalar-scalar product	$m+n$
Scalar-vector product	1	Storage beyond $\mathbf{A}$, $\hat{\mathbf{H}}$	$(m+2n+1)N+m+n$

Table 2: Average cost per k-loop of Algorithm 3.2 without the computation of the $\tilde{\mathbf{v}}$-vectors and its corresponding total storage requirement.

Recall that Algorithm 3.2 also computes the vectors $\tilde{\mathbf{v}}$ in Lines 6 and 21. These vectors, however, actually contribute nothing to the computation of the quantities stated in Line 2 of Algorithm 4.3. So, we can skip Lines 6 and 21 when we implement the second step of Algorithm 4.3. The corresponding cost of this "$\hat{\mathbf{H}}$-only" version of Algorithm 3.2 is listed in Table 2. As a result, Algorithm 4.3 requires about $(1+m/n)l$ matrix-vector products with $\mathbf{A}$ to get the l-th Padé approximant $\mathbf{f}_l(\theta)$. When $m < n$, it is cheaper in terms of matrix-vector products than MPVL which needs $2l$ matrix-vector products (one with $\mathbf{A}$ and the other with $\mathbf{A}^H$) to get $\mathbf{f}_l(\theta)$.

5 An augmented version of TF PVL

The derivation of Algorithm 4.3 is based on Theorem 2.3. The theorem guarantees that breakdown does not occur and deflation (see, for instance, [1]) is not 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, it is possible that the parameter ν could be so small that the approximant $\mathbf{f}_\nu(\theta)$ were not accurate enough. To avoid the situation, we suggest the following use of Algorithm 4.3.

We can first augment the input/output data $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ before using Algorithm 4.3 by introducing some random vectors, say $\mathbf{r}_{\hat{U}} \in \mathcal{C}^{N\times n_0}$ and $\mathbf{r}_{\hat{V}} \in \mathcal{C}^{N\times m_0}$, as follows

$$\hat{\mathbf{U}}_{aug} = [\mathbf{r}_{\hat{U}}, \hat{\mathbf{U}}], \qquad \hat{\mathbf{V}}_{aug} = [\mathbf{r}_{\hat{V}}, \hat{\mathbf{V}}]\,. \tag{21}$$

Then, the matrix-valued function $\mathbf{F}(\theta+z_0)$ of (13) which we want to estimate is just the diagonal block at the lower-right corner of the matrix-valued function

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

Let

$$\hat{\mathbf{U}}_{aug} = \mathbf{Q}_{aug\hat{U}}\mathbf{R}_{aug\hat{U}}, \qquad \hat{\mathbf{V}}_{aug} = \mathbf{Q}_{aug\hat{V}}\mathbf{R}_{aug\hat{V}}$$

be the QR factorizations of $\hat{\mathbf{U}}_{aug}$ and $\hat{\mathbf{V}}_{aug}$ respectively, where $\mathbf{Q}_{aug\hat{U}} \in \mathcal{C}^{N\times(n_0+n)}$, $\mathbf{Q}_{aug\hat{V}} \in \mathcal{C}^{N\times(m_0+m)}$, $\mathbf{R}_{aug\hat{U}} \in \mathcal{C}^{(n_0+n)\times(n_0+n)}$ and $\mathbf{R}_{aug\hat{V}} \in \mathcal{C}^{(m_0+m)\times(m_0+m)}$. Then the function $\mathbf{F}_{aug}(\theta+z_0)$ can be written as

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

We now apply Algorithm 4.3 to find a Padé approximant $\mathbf{f}_l(\theta)$ to the function

$$\mathbf{Q}_{aug\hat{U}}^H(\mathbf{I}-\theta\mathbf{A})^{-1}\mathbf{Q}_{aug\hat{V}}\,, \tag{22}$$

then compute the following function $\mathbf{f}_l^{aug}(\theta)$ as an approximant to $\mathbf{F}_{aug}(\theta + z_0)$

$$\mathbf{f}_l^{aug}(\theta) \;=\; \mathbf{R}_{aug\hat{U}}^H \mathbf{f}_l(\theta) \mathbf{R}_{aug\hat{V}} \,. \tag{23}$$

Recalling that the original $\mathbf{F}(\theta+z_0)$ is the lower-right block of the matrix $\mathbf{F}_{aug}(\theta+z_0)$, we therefore extract the lower-right block of $\mathbf{f}_l^{aug}(\theta)$ as an approximant to $\mathbf{F}(\theta + z_0)$.

Since the columns of $\mathbf{Q}_{aug\hat{U}}$ and $\mathbf{Q}_{aug\hat{V}}$ are well linearly independent and have random directions to some degree, the hypothesis of Theorem 2.3 can be expected to hold for large ν when we compute $\mathbf{f}_l(\theta)$. Thus, a high-order Padé approximant to the function (22) can be expected to obtain without encountering breakdown or deflation.

The function $\mathbf{f}_l^{aug}(\theta)$ is an l-th Padé approximant of the augmented function $\mathbf{F}_{aug}(\theta + z_0)$, as stated in the following theorem.

Theorem 5.1 *Let* $\max\{m + m_0, n + n_0\} \leq l$ *and suppose both* $\hat{\mathbf{U}}_{aug}$ *and* $\hat{\mathbf{V}}_{aug}$ *are nonsingular. Then the function* $\mathbf{f}_l^{aug}(\theta)$ *of (23) is an* l*-th Padé approximant of the function* $\mathbf{F}_{aug}(\theta + z_0)$ *and*

$$\mathbf{f}_l^{aug}(\theta) = \mathbf{F}_{aug}(\theta + z_0) + O(\theta^{\lfloor l/(m+m_0)\rfloor + \lfloor l/(n+n_0)\rfloor})$$

on the disc $\{\theta : |\theta| < 1/\delta\}$ *where* $\delta = \max\{\delta(\mathbf{A}), \delta(\hat{\mathbf{H}}_l)\}$ *and where* $\hat{\mathbf{H}}_l$ *is the matrix obtained from Algorithm 3.2 when applied to the function (22).*

Proof. In the definition (23) of $\mathbf{f}_l^{aug}(\theta)$, the function $\mathbf{f}_l(\theta)$ is an l-th Padé approximant of the function (22) found by Algorithm 4.3. Hence, by definition 4.1, $\mathbf{f}_l(\theta)$ can be expressed in the form of (16) and its Taylor expansion about $\theta = 0$ matches as many leading terms of the Taylor expansion of (22) as possible. It then follows that $\mathbf{f}_l^{aug}(\theta)$ has the form (16) and that the number of matched coefficients of the Taylor expansions of $\mathbf{f}_l^{aug}(\theta)$ and $\mathbf{F}_{aug}(\theta + z_0)$ about $\theta = 0$ is maximal providing that $\mathbf{R}_{aug\hat{U}}$ and $\mathbf{R}_{aug\hat{V}}$ are both nonsingular. Finally, the approximation has a truncation error of order $\lfloor l/(m + m_0)\rfloor + \lfloor l/(n + n_0)\rfloor$ follows from Theorem 4.2. □

6 Numerical Experiments

In this section, we present some examples to illustrate the effectiveness of Algorithm 4.3. In all the experiments, we defined the parameters λ_k and $\hat{h}_{k+m}^{(k)}$ in Algorithm 3.2 as follows,

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

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.

Example 1. This is the same example used in [2, 3, 6] from a three-dimensional electromagnetic problem model via PEEC (partial element equivalent circuit) [17]. 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 (13) is a scalar-valued function.

In [2, 3, 6], the magnitude of $\mathbf{F}(z)$ with $z = 2\pi w\sqrt{-1}$ was approximated over the frequency interval $1 \leq w \leq 5 \times 10^9$ with a 60-th Padé approximant $\mathbf{f}_{60}(\theta)$ in Theorem 4.2 obtained by the PVL method, where $\theta = 2\pi w\sqrt{-1} - z_0$. The expansion point z_0 used was $z_0 = 2\pi 10^9$ and the numerical results therein illustrated that the approximation produced was indistinguishable from

the true $|\mathbf{F}(2\pi w\sqrt{-1})|$.

Figure 1(a): We repeated the experiment performed in [2, 3, 6], but Algorithm 4.3 was employed to compute $\mathbf{f}_{60}(\theta)$ according to (20). The figure plots the graphs of $|\mathbf{f}_{60}(\theta)|$ and $|\mathbf{F}(2\pi w\sqrt{-1})|$. We observe that the approximation is not accurate in the high frequency w-region. Theoretically, both PVL and TFMPVL produce the same $\mathbf{f}_{60}(\theta)$ in this case. Numerically, however, the TFMPVL method is less stable. It took 120 matrix-vector products with $\mathbf{A}$ to get $\mathbf{f}_{60}(\theta)$ whereas only 60 matrix-vector products with $\mathbf{A}$ required by the PVL method.

Figure 1(b): We continued the experiment of Figure 1(a), but this time, we used the method of §5 to compute $\mathbf{f}_{60}(\theta)$. We first augmented the vector $\hat{\mathbf{U}}$ into a 306×6 matrix $\hat{\mathbf{U}}_{aug}$ by adding 5 random vectors to it, that is, we set $n_0 = 5$ in (21). Then, we applied Algorithm 4.3 to (22) and then compute $\mathbf{f}_{60}(\theta)$ by (23). Again, Figure 1(b) plots the graphs of $|\mathbf{f}_{60}(\theta)|$ and $|\mathbf{F}(2\pi w\sqrt{-1})|$. We can see that the approximation has been improved. The reason to get the improvement is that the number of matrix-vector products with $\mathbf{A}$ to obtain $\mathbf{f}_{60}(\theta)$ is now reduced from 120 to 70.

Figure 2(a): Although the approximation in Figure 1(b) received a significant improvement, it is still unacceptable. Recall that the $\mathbf{f}_{60}(\theta)$ generated by the PVL method has a truncation error of order 120. According to Theorem 4.2, however, the truncation error of the $\mathbf{f}_{60}(\theta)$ in Figure 1(b) is only of order 70. This explains in part why the $\mathbf{f}_{60}(\theta)$ of Figure 1(b) does not perform so well as the $\mathbf{f}_{60}(\theta)$ produced by PVL does. Here, we repeated the experiment of Figure 1(b). Instead of $\mathbf{f}_{60}(\theta)$, however, we computed $\mathbf{f}_{103}(\theta)$ which has a truncation error of order about 120. The number of matrix-vector multiplications with $\mathbf{A}$ in the computation was about 120. Hence, TFMPVL is not more expensive than PVL in terms of matrix-vector multiplication in this experiment. The graphs of $\mathbf{f}_{103}(\theta)$ and $|\mathbf{F}(2\pi w\sqrt{-1})|$ are almost indistinguishable.

Figure 2(b): In the above experiments, we chose the expansion point $z_0 = 2\pi 10^9$. Here we redid the experiment of Figure 1(a), but with a new expansion at $z_0 = 2\pi\sqrt{-1} \times 2.5 \times 10^9$. Again, we observe that the $\mathbf{f}_{60}(\theta)$ performs bad in the w-region far away from the expansion point z_0.

Figure 3(a): We repeated the experiment of Figure 1(b) with the new expansion point $z_0 = 2\pi\sqrt{-1} \times 2.5 \times 10^9$. We observe no difference between the graphs of $|\mathbf{f}_{60}(\theta)|$ and $|\mathbf{F}(2\pi w\sqrt{-1})|$. 70 matrix-vector multiplications with $\mathbf{A}$ were needed to compute $\mathbf{f}_{60}(\theta)$. The corresponding truncation error is of order 70.

Figure 3(b): We plotted the relative error

$$\frac{|\,|F(2\pi w\sqrt{-1})| - |\mathbf{f}_{60}(\theta)|\,|}{|F(2\pi w\sqrt{-1})|}$$

where $\mathbf{f}_{60}(\theta)$ is from the experiment of Figure 3(a) and where $\theta = 2\pi w\sqrt{-1} - z_0$. From the figure, we see that $\mathbf{f}_{60}(\theta)$ is a very good estimator of $\mathbf{F}(z)$ over the interval we considered.

Example 2. The second example is the 120×120 system from [21] which describes the effects of a magnetic actuator on the radial tracking arm of a portable compact disc player. In this example, the matrix $\mathbf{C}$ is an identity matrix and the $\mathbf{G}$ is sparse with nonzero entries only on its diagonal and anti-diagonal. The (input) matrix $\mathbf{R}$ contains two columns which respectively correspond to

the voltages applied to the radial arm and lens actuators. The (output) matrix $\mathbf{L}$ contains two columns which respectively correspond to the positions of the radial arm and the focusing lens [21]. This is a difficult example due to the data which are too ill-conditioning.

We simulate the 2×2 matrix-valued function $\mathbf{F}(z)$ of (13) with the expansion point $z_0 = 5 \times 10^4\sqrt{-1}$. The corresponding matrix $\mathbf{A}$ defined in (14) then has a spectral radius $\delta(\mathbf{A})$ of about 1.49×10^{-4}. By Theorem 4.2, we can have a l-th Padé approximant $\mathbf{f}_l(\theta)$ defined by (19) to the function $\mathbf{F}(z)$ on a disc contained in $\{z = z_0 + \theta : |\theta| < 1/\delta(\mathbf{A}) = 0.67 \times 10^4\}$. We computed $\mathbf{F}(z)$ and its 30-th Padé approximant $\mathbf{f}_{30}(\theta)$ by Algorithm 4.3. The numerical results of the magnitudes of the four entries of $\mathbf{f}_{30}(\theta)$ and $\mathbf{F}(z)$ with $z = w\sqrt{-1}$ and $\theta = z - z_0$ are plotted over the w-interval $(10^4, 10^5)$ in Figures 4 and 5, respectively. Since $z = w\sqrt{-1}$, z lies in the disc stated above only when w is in $\Delta \equiv (4.33 \times 10^4, 5.67 \times 10^4)$. From the figures, we can see that the approximations are also good over a large region containing the interval Δ. This experiment illustrates that TFMPVL can behave well over a larger region than that stated by Theorem 4.2.

We also tried the method introduced in §5 by setting $m_0 = 0, n_0 = 5, 10$. There were some improvements observed in accuracy, but not so significant as in Example 1.

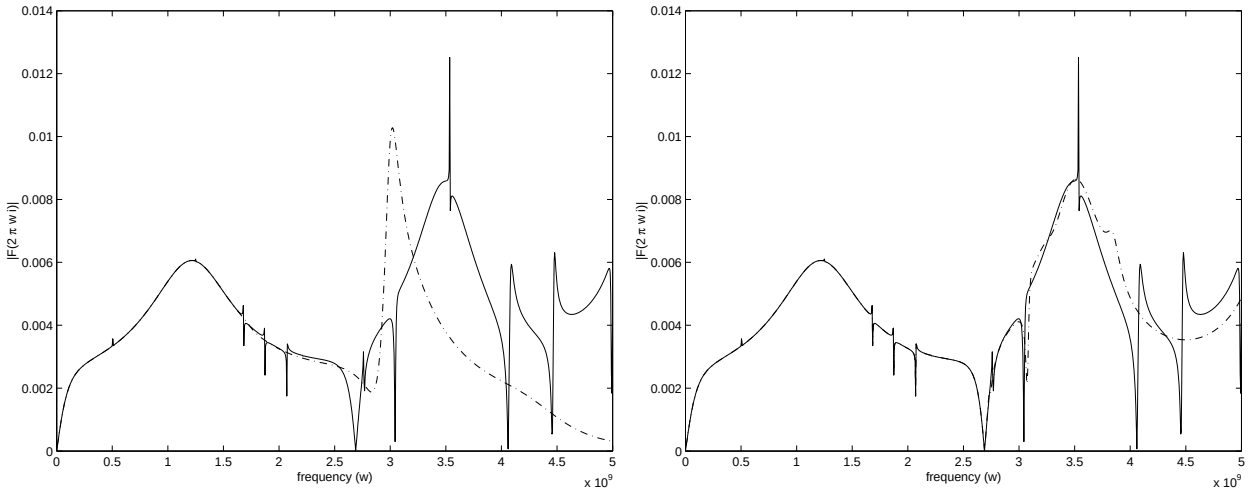

Figure 1: (a) Result computed in ordinary Padé, expansion point $z_0 = 2\pi 10^9$ and 60 TFMPVL iterations; (b) Result computed in augmented Padé, expansion point $z_0 = 2\pi 10^9$, 5 random vectors added to the left and 60 TFMPVL iterations. Solid: exact; Dashdot: approximant.

7 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. The method has been applied to 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 carrying the transpose of the matrix $\mathbf{A}$, TFMPVL may also reduce the average number of matrix-vector products per iteration from 2 (which is required by the two-side MPVL method) to $1 + m/n$, where m, n are the number of input, output vectors respectively. Numerical experiments indicate that, although the TFMPVL method is less stable than the original two-sided MPVL method in general, its numerical properties tend to be as favorable as those for MPVL when we increase the number n of the left starting vectors. However, increasing n will decrease the accuracy of the Padé approximation (Theorem 4.2), and therefore

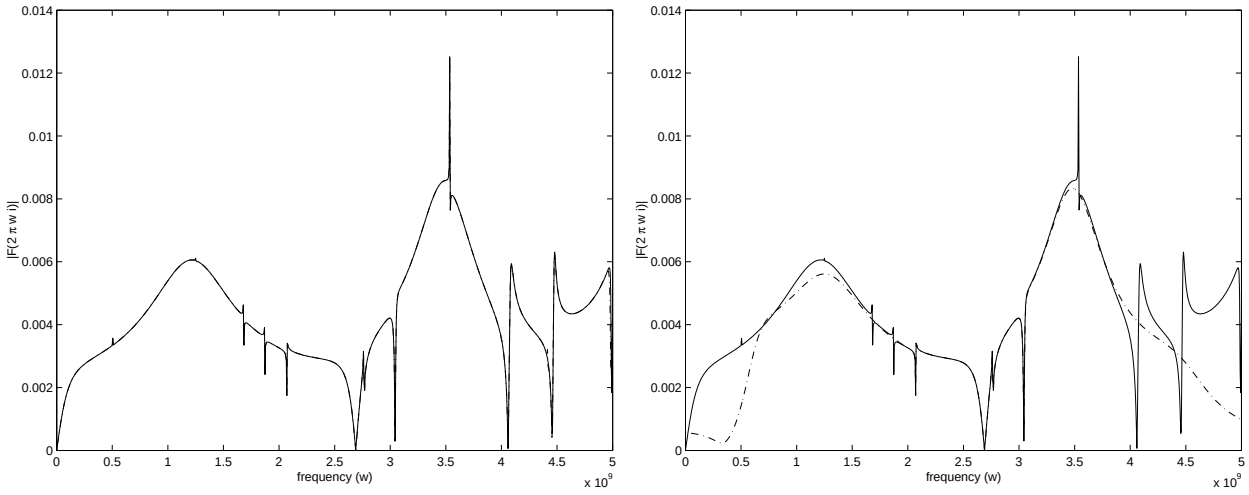

Figure 2: (a) Result computed in augmented Padé, expansion point $z_0 = 2\pi 10^9$, 5 random vectors added to the left and 103 TFMPVL iterations; (b) Result computed in ordinary Padé, expansion $z_0 = 2\pi\sqrt{-1} \times 2.5 \times 10^9$ and 60 TFMPVL iterations. Solid: exact; Dashdot: approximant.

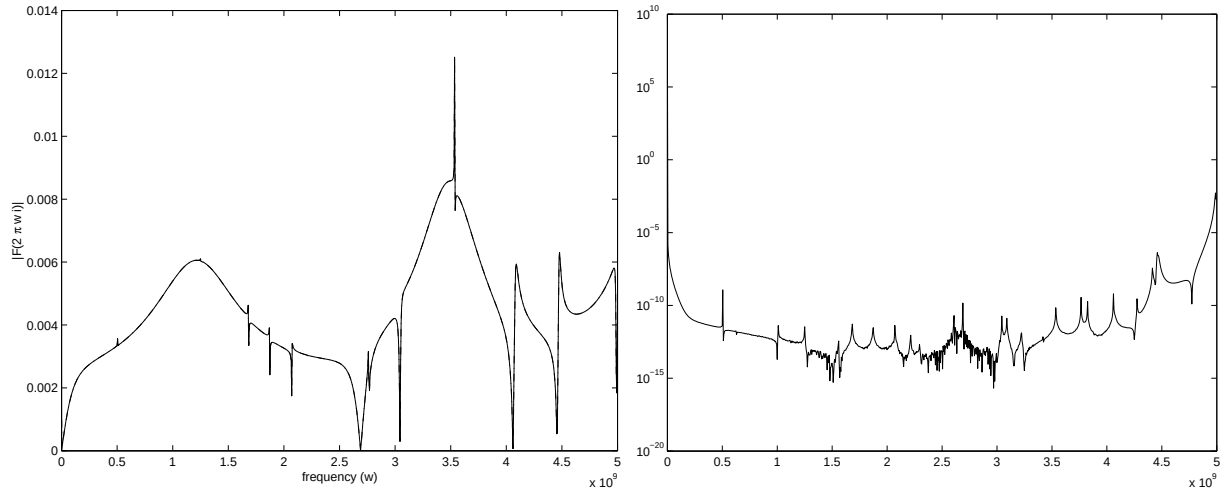

Figure 3: (a) Result computed in augmented Padé, expansion $z_0 = 2\pi\sqrt{-1} \times 2.5 \times 10^9$, 5 random vectors added to the left and 60 TFMPVL iterations. Solid: exact; Dashdot: approximant. (b) Relative error of the approximation in Figure 3(a).

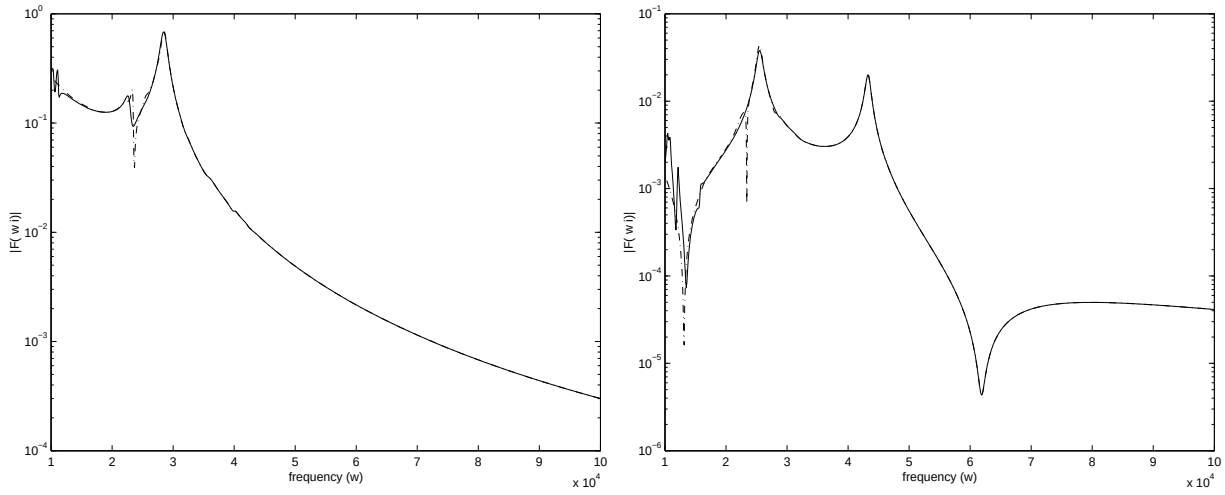

Figure 4: Frequency responses for Example 2. (a) Radial arm input to radial arm position output (the $(1,1)$-th entry of $\mathbf{F}(z)$); (b) Lens actuator input to radial arm position output (the $(1,2)$-th entry of $\mathbf{F}(z)$). Solid: exact; Dashdot: 30-th TFMPVL Padé approximant.

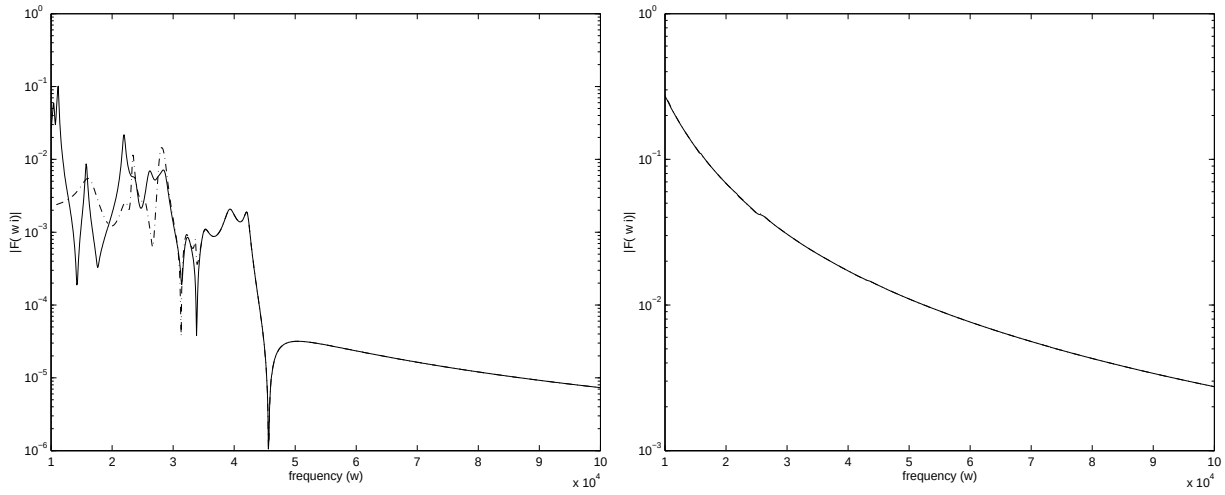

Figure 5: Frequency responses for Example 2. (a) Radial arm input to lens position output (the $(2,1)$-th entry of $\mathbf{F}(z)$); (b) Lens actuator input to lens position output (the $(2,2)$-th entry of $\mathbf{F}(z)$). Solid: exact; Dashdot: 30-th TFMPVL Padé approximant.

more steps are required to achieve a specified accuracy. Moreover, by adding some random vectors to the starting vectors, we may avoid the possible early occurrence of breakdown and deflation. We can understand the transpose-free Lanczos method to some extent through the behavior of the TFMPVL method.

In general, a practical multiple-vector Lanczos algorithm will have to include some deflation procedure. For the transpose-free algorithm, this may change the length of the required recurrences, requiring the storage of a few more vectors from step to step. This variant of this algorithm will be addressed in a future paper. However, the basic concept behind the transpose free Lanczos with deflation is identical to the concept behind the algorithm developed in this paper. In other words, the algorithm of this paper suffices to show the existence and feasibility of a transpose-free MIMO Lanczos procedure capable of accepting multiple vectors on both the left and the right, suitable for the computation of Padé approximants.

Acknowledgement. The authors wish to thank Prof. Zhao J. Bai for his help in our experiments. We would also like to thank the anonymous referees for their valuable comments on an earlier version of this paper.

eferences

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

[5] T. Chan, L. de Pillis and H. van der Vorst, *A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems,* Tech. Report CAM 91-17, UCLA, Dept. of Math., Los Angeles, CA 90024-1555, 1991.

[6] 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.

[7] ——, *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.

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

[9] ——, *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.

[10] ——, *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.

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

[12] K. Gallivan, E. Grimme, D. Sorensen and P. Van Dooren, *On some modifications of the Lanczos algorithm and the relation with Padé approximations,* in Proceedings of 1995 International Congress on Industrial and Applied Mathematics.

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

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

[15] ——, *Lanczos-type solvers for nonsymmetric linear systems of equations,* Acta Numerica, 271-397, 1997.

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

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

[18] Y. Saad, *Iterative methods for sparse linear systems,* PWS Publishing Company, 1996.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.