

Transpose-free Matrix Padé Via Lanczos Method

D. Boley* and M. Yeung*

June 29, 1998

Abstract

A transpose-free two-sided nonsymmetric Lanczos method is developed for multiple starting vectors on both the left and right. 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. The method is mathematically equivalent to the two-sided methods, but avoids the use of the transpose of the system matrix, and under certain circumstances will actually reduce the total number of matrix-vector products needed. The method is illustrated with some numerical examples.

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

AMS subject classifications: Primary 65F15; Secondary 65G05.

1 Introduction

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

$$\frac{dx}{dt} = Ax(t) + \hat{V}w(t), \quad y(t) = \hat{U}^T x(t).$$

where $A \in \mathcal{C}^{N \times N}$, $\hat{V} \in \mathcal{C}^{N \times m}$, $\hat{U} \in \mathcal{C}^{N \times n}$, and $w(t), y(t)$ and $x(t)$ are vector-valued functions of length m, n and N , respectively. See, for instance, [4, 5, 7, 8].

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

$$Y(z) = \hat{U}^T (zI - A)^{-1} \hat{V} \cdot W(z) \equiv F(z) \cdot W(z) = \sum_{i=1}^{\infty} \frac{M_i}{z^i} \cdot W(z),$$

where we show the expansion of the transfer function in a power series about $z = \infty$. The techniques of this paper can be applied just as easily to expansions about other points, but for simplicity we develop the theory for $z = \infty$. The M_i are called the *moments* or *Markov parameters*. We seek a new lower order system

$$\frac{d\tilde{x}}{dt} = \tilde{A}\tilde{x} + \tilde{V}w(t), \quad \tilde{y}(t) = \tilde{U}^T \tilde{x}(t)$$

with frequency domain description

$$\tilde{Y}(z) = \tilde{U}^T (zI - \tilde{A})^{-1} \tilde{V} \cdot W(z) = \sum_{i=1}^{\infty} \frac{\tilde{M}_i}{z^i} \cdot W(z),$$

that approximates the original. Specifically, we seek the matrix Padé approximant, which is a lower order system for which the first l moments agree with the original: $\tilde{M}_i = M_i$ for $i = 1, \dots, l$, for a given l . In

*Computer Science and Engineering Department, University of Minnesota, Minneapolis, MN 55455. E-mail: {boley,yeung}@cs.umn.edu. This research was supported in part by NSF grants CCR-9405380 and CCR-9628786.

[5, 7, 9] the MPVL method was proposed independently to compute this decomposition in a stable manner using the two-sided Lanczos algorithm with multiple starting vectors [1]. For convenience, we will refer to this Lanczos method as a MIMO Lanczos algorithm. This Lanczos algorithm is applied to a matrix A , and right and left starting vectors \hat{V} and \hat{U} , respectively. It employs matrix-vector products involving both A and A^T , and generates a block tridiagonal matrix which plays the role of \hat{A} in the reduced order system above. 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. [2, 9, 8] and references therein. We give some specifics of this connection relevant to this paper in §2 after we have defined some more notation.

It is well known that the Lanczos process is intimately related to bi-conjugate gradient (BiCG) method for solving nonsymmetric systems of linear equations [10, 17]. Transpose-free versions, e.g., 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 A^T , and in the process can actually reduce the total number of matrix-vector products necessary. These also enjoy improved stability properties [23] over the BiCG or CGS [20]. The goal of this paper is to apply the transpose free methods to the computation of the matrix Padé approximants for linear dynamical systems. In this way, the advantages of the transpose-free methods that were found in the context of systems of linear equations will be enjoyed also in the context of Padé approximants. To accomplish this, we extend the ML(k)BiCGSTAB method to handle multiple starting vectors on both the left and right, and in the process re-develop the method in the context of the general two-sided Lanczos algorithm.

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 we show how to use this Lanczos procedure to compute the Padé approximant, and in §5 we illustrate the methods with some numerical experiments.

2 Lanczos Procedure for Multiple Starting Vectors and the MPVL Method

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

$$(a) \quad p_{jn+i'} = (A^T)^j \hat{u}_{i'}, \quad (b) \quad q_{jm+i} = A^j \hat{v}_i \quad (1)$$

for $i' = 1, 2, \dots, n$, $i = 1, 2, \dots, m$ and $j = 0, 1, 2, \dots$. The ABFH Lanczos procedure generates two sequences $\{u_{\nu'}\}_{\nu'=1,2,\dots}$ and $\{v_l\}_{l=1,2,\dots}$ of vectors such that

$$\begin{aligned} u_{\nu'} &\in \mathcal{K}_{\nu'}(A^T, \hat{U}) \quad \text{and} \quad u_{\nu'} - \mathcal{K}_{\nu'-1}(A, \hat{V}), \\ v_l &\in \mathcal{K}_l(A, \hat{V}) \quad \text{and} \quad v_l - \mathcal{K}_{l-1}(A^T, \hat{U}), \end{aligned} \quad (2)$$

where $\mathcal{K}_{\nu'}(A^T, \hat{U}) \stackrel{\text{def}}{=} \text{span}\{p_1, p_2, \dots, p_{\nu'}\}$ and $\mathcal{K}_l(A, \hat{V}) \stackrel{\text{def}}{=} \text{span}\{q_1, q_2, \dots, q_l\}$. The subspaces $\mathcal{K}_{\nu'}(A^T, \hat{U})$ and $\mathcal{K}_l(A, \hat{V})$ are referred to as the left block Krylov subspace and the right block Krylov subspace, respectively.

The existence of such two sequences of vectors can be guaranteed if the matrices,

$$W_s = \begin{bmatrix} p_1^T q_1 & p_1^T q_2 & \cdots & p_1^T q_s \\ p_2^T q_1 & p_2^T q_2 & \cdots & p_2^T q_s \\ \vdots & \vdots & \ddots & \vdots \\ p_s^T q_1 & p_s^T q_2 & \cdots & p_s^T q_s \end{bmatrix}, \quad \text{for all } s = 1, 2, \dots, \nu$$

are nonsingular for some ν .

Lemma 2.1 *If all the leading principal submatrices of W_ν are nonsingular, then there exist two sets $\{u_\nu\}_{\nu=1}^\nu$ and $\{v_l\}_{l=1}^\nu$ of linearly independent vectors which satisfy properties (2). Moreover,*

$$\text{span}\{u_1, u_2, \dots, u_{l'}\} = \mathcal{K}_{l'}(A^T, \hat{U}), \quad \text{span}\{v_1, v_2, \dots, v_l\} = \mathcal{K}_l(A, \hat{V}),$$

where $l', l = 1, 2, \dots, \nu$.

Proof. In fact, if we express v_l as

$$v_l = \gamma_1^{(l)} q_1 + \gamma_2^{(l)} q_2 + \dots + \gamma_{l-1}^{(l)} q_{l-1} + q_l, \quad (3)$$

then (2) is equivalent to $W_{l-1} \gamma^{(l)} = -f$ where $\gamma^{(l)} = [\gamma_1^{(l)}, \gamma_2^{(l)}, \dots, \gamma_{l-1}^{(l)}]^T$ and $f = [p_1^T q_l, p_2^T q_l, \dots, p_{l-1}^T q_l]^T$. Furthermore, v_l defined by (3) satisfies $v_l \neq p_l$ for $l \leq \nu$ and hence v_1, v_2, \dots, v_ν are linearly independent. The same arguments can be applied to the vectors $u_{l'}$. \square

From the definition of q_l , we note that $q_l = Aq_{l-m}$ for $l > m$. Applying (3) to itself recursively, we can write v_l ($l > m$) in terms of the previous v_1, v_2, \dots, v_{l-1} as follows,

$$v_l = Av_{l-m} - \bar{h}_{l-1}^{(l-m)} v_{l-1} - \bar{h}_{l-2}^{(l-m)} v_{l-2} - \dots - \bar{h}_1^{(l-m)} v_1. \quad (4)$$

A similar equation for vectors $u_{l'}$ ($l' > n$) is also available.

Lemma 2.2 *The vectors v_l and $u_{l'}$ in Lemma 2.1 with $l > m$ and $l' > n$ can be expressed with $m + n + 1$ term recursion relationships of the forms*

$$v_l = Av_{l-m} - \bar{h}_{l-1}^{(l-m)} v_{l-1} - \bar{h}_{l-2}^{(l-m)} v_{l-2} - \dots - \bar{h}_{\bar{m}_{l-m}}^{(l-m)} v_{\bar{m}_{l-m}} \quad (5)$$

and

$$u_{l'} = A^T u_{l'-n} - \tilde{h}_{l'-1}^{(l'-n)} u_{l'-1} - \tilde{h}_{l'-2}^{(l'-n)} u_{l'-2} - \dots - \tilde{h}_{\bar{m}_{l'-n}}^{(l'-n)} u_{\bar{m}_{l'-n}},$$

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

Proof. Noting that $v_s \in \text{span}\{p_1, p_2, \dots, p_{s-1}\}$, $v_s \neq p_s$ and $p_s^T Av_{l-m} = p_{s+n}^T v_{l-m} = 0$ for $s \leq l - m - n - 1$, and examining in turn

$$p_s^T v_l = p_s^T Av_{l-m} - \bar{h}_{l-1}^{(l-m)} p_s^T v_{l-1} - \bar{h}_{l-2}^{(l-m)} p_s^T v_{l-2} - \dots - \bar{h}_1^{(l-m)} p_s^T v_1$$

for $s = 1, 2, \dots, l - m - n - 1$, we find all the coefficients in (4) zero except $\bar{h}_{l-1}^{(l-m)}, \bar{h}_{l-2}^{(l-m)}, \dots, \bar{h}_{\bar{m}_{l-m}}^{(l-m)}$. \square

Set $V_s = [v_1, v_2, \dots, v_s]$ and set $\bar{H}_\nu = (\bar{h}_{ij})_{i=1, \dots, \nu; j=1, \dots, \nu-m}$, the $\nu \times (\nu - m)$ band matrix with $\bar{h}_{j+m, j} = 1$; $\bar{h}_{ij} = \bar{h}_i^{(j)}$ for $\bar{m}_j \leq i \leq j + m - 1$; with $\bar{h}_{ij} = 0$ otherwise. Then the recurrence relations (5) can be written in matrix form as

$$AV_{\nu-m} = V_\nu \bar{H}_\nu.$$

Similar results can be drawn for the vectors $u_{l'}$. We collect the above results into the following.

Theorem 2.3 *Let vectors $\hat{u}_1, \hat{u}_2, \dots, \hat{u}_n, \hat{v}_1, \hat{v}_2, \dots, \hat{v}_m$ be given starting vectors and let p_s 's, q_s 's, $\mathcal{K}_{l'}(A^T, \hat{U})$, $\mathcal{K}_l(A, \hat{V})$ and W_ν be as defined above. If all the leading principal submatrices of W_ν are nonsingular, then there exist two sets $\{u_{l'}\}_{l'=1}^\nu$ and $\{v_l\}_{l=1}^\nu$ of linearly independent vectors, an $\nu \times (\nu - m)$ band matrix \bar{H}_ν , 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{H}_ν , 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*

$$\text{span}\{u_1, u_2, \dots, u_{l'}\} = \mathcal{K}_{l'}(A^T, \hat{U}), \quad \text{span}\{v_1, v_2, \dots, v_l\} = \mathcal{K}_l(A, \hat{V}) \quad (6)$$

and

$$u_{l'}^T v_l \begin{cases} \neq 0 & \text{if } l' = l \\ = 0 & \text{if } l' \neq l \end{cases}$$

for all $l', l = 1, 2, \dots, \nu$. Moreover,

$$A^T U_{\nu-n} = U_{\nu} \tilde{H}_{\nu} \quad \text{and} \quad AV_{\nu-m} = V_{\nu} \tilde{H}_{\nu},$$

where $U_{l'} = [u_1, u_2, \dots, u_{l'}]$ and $V_l = [v_1, v_2, \dots, v_l]$.

The MPVL method [5, 7, 8] is an approach, theoretically based on the ABFH Lanczos procedure, to compute matrix Padé approximations of matrix-valued functions of the form

$$F(z) = \hat{U}^T (zI - A)^{-1} \hat{V}, \quad (7)$$

where I is the $N \times N$ identity matrix.

An l -th Padé approximant $F_l(z)$ of $F(z)$ is defined to be a function such that

$$F_l(z) = F(z) + O(z^{-\lfloor l/m \rfloor - \lfloor l/n \rfloor - 1}).$$

See, for instance, [5, 7]. With the given blocks \hat{U} and \hat{V} as the n left starting vectors and m right starting vectors respectively in the ABFH Lanczos procedure, we obtain data $U_{\nu}, V_{\nu}, \tilde{H}_{\nu}$ and \tilde{H}_{ν} . Because of (6), there exist matrices $\eta \in \mathcal{C}^{n \times n}$ and $\rho \in \mathcal{C}^{m \times m}$ such that $\hat{U} = U_{\nu} \eta$ and $\hat{V} = V_{\nu} \rho$. Let H_l be the $l \times l$ principal block of the matrix \tilde{H}_{ν} , $0_{l' \times l}$ denote the $l' \times l$ zero matrix and set $D_l = U_l^T V_l$. Then the following theorem [7, 8] provides us an l -th Padé approximant.

Theorem 2.4 *Let $\max\{m, n\} \leq l$. Then,*

$$F_l(z) = \begin{bmatrix} D_n^T \eta \\ 0_{(l-n) \times n} \end{bmatrix}^T (zI - H_l)^{-1} \begin{bmatrix} \rho \\ 0_{(l-m) \times m} \end{bmatrix} \quad (8)$$

is an l -th Padé approximant of the function $F(z)$.

The MPVL method consists of two steps: (a) the ABFH Lanczos procedure is run for the first l steps to obtain data H_l, D_n, η and ρ , then (b) the Lanczos-Padé connection (8) then yields coefficient matrices for the reduced order linear dynamical system

$$\frac{d\tilde{x}}{dt} = H_l \tilde{x}(t) + \begin{bmatrix} \rho \\ 0_{(l-m) \times m} \end{bmatrix} w(t), \quad \tilde{y}(t) = \begin{bmatrix} D_n^T \eta \\ 0_{(l-n) \times n} \end{bmatrix}^T \tilde{x}(t).$$

whose transfer function is exactly the l -th Padé approximant $F_l(z)$ to the transfer function $F(z)$. Observe that the ABFH Lanczos procedure generates not only the data H_l, D_n, η, ρ but U_l, V_l, \tilde{H}_l as well. However, the data U_l, V_l and \tilde{H}_l do not contribute directly to compute $F_l(z)$ in (8). Instead, they are used only to obtain the matrix H_l in the Lanczos procedure itself. The question then arises as to whether or not it is possible in the ABFH Lanczos procedure to bypass the computations of U_l, V_l and \tilde{H}_l and still generate the quantities that are related to (8). In the next section, we will give a partial answer to this question by deriving a transpose-free version of a limited version of the ABFH Lanczos procedure which does not involve A^T in its implementation.

3 Transpose-free Lanczos Procedure for Multiple Starting Vectors

As mentioned in the Introduction, a number of articles in the literature have discussed Lanczos implementations without accessing A^T , see, for instance, [3, 6, 10, 13, 14, 17, 19, 20, 22, 23], mostly in the context of solving systems of linear equations. Techniques of avoiding matrix-vector multiplies with A^T in the classical Lanczos procedure can be generalized to the current case. In this section, we will give a new variant of

a limited ABFH Lanczos procedure which computes the quantities needed in (8) without using A^T . To simplify the derivation, we will assume that the deflation or look-ahead features in the full ABFH algorithm are not needed.

We continue to use the notations introduced in §2 and suppose the assumption of Theorem 2.3 holds. Also, we assume $m \leq n$. Now, we consider the equation (5),

$$v_{l+m} = Av_l - \bar{h}_{\bar{m}_l}^{(l)} v_{\bar{m}_l} - \bar{h}_{\bar{m}_l+1}^{(l)} v_{\bar{m}_l+1} - \cdots - \bar{h}_{l+m-1}^{(l)} v_{l+m-1},$$

where $l = 1, 2, \dots, \nu - m$. Because of the property (2), the coefficients $\bar{h}_s^{(l)}$ are determined by

$$\bar{h}_s^{(l)} = \frac{p_s^T Av_l - p_s^T \sum_{t=\bar{m}_l}^{s-1} \bar{h}_t^{(l)} v_t}{p_s^T v_s}, \quad s = \bar{m}_l, \bar{m}_l + 1, \dots, l + m - 1.$$

Thus, we have the following procedure to compute the matrix \bar{H}_ν .

Lanczos Procedure Version 1.

1. Compute vectors $\{v_l\}_{l=1}^m$ such that $v_l \perp \text{span}\{\hat{u}_1, \dots, \hat{u}_{l-1}\}$ and $v_l \in \text{span}\{\hat{v}_1, \dots, \hat{v}_l\}$, and compute p_s , $s = 1, \dots, m$, according to (1a).
2. For $l = 1, 2, \dots, \nu - m$
3. $\bar{m}_l = \max\{l - n, 1\}$;
4. For $s = \bar{m}_l, \bar{m}_l + 1, \dots, l + m - 1$
5. $\bar{h}_s^{(l)} = \frac{p_s^T Av_l - p_s^T \sum_{t=\bar{m}_l}^{s-1} \bar{h}_t^{(l)} v_t}{p_s^T v_s}$;
6. End
7. $v_{l+m} = Av_l - \sum_{s=\bar{m}_l}^{l+m-1} \bar{h}_s^{(l)} v_s$;
8. Compute (according to (1a)) $p_{l+m} = \begin{cases} \hat{u}_{l+m} & \text{if } l + m \leq n \\ A^T p_{l+m-n} & \text{if } l + m > n \end{cases}$
9. End

The above procedure requires the use of A^T to calculate p_{l+m} in line 8. In order to remove the A^T , we rearrange the outer *for* loop of the procedure into a form more convenient for our development. Let $l = jn + i$ and let the index i vary from 1 to n and j start with 0. Moreover, we regard any quantity with non-positive subscript as zero. Then we convert the loop in l (omitting lines 3 and 8) into doubly nested loops in j and i respectively as follows.

Lanczos Procedure Version 2.

1. For $j = 0, 1, 2, \dots$
2. For $i = 1, 2, \dots, n$
3. For $s = (j-1)n + i, \dots, jn + i + m - 1$
4. $\bar{h}_s^{(jn+i)} = \frac{p_s^T Av_{jn+i} - p_s^T \sum_{t=(j-1)n+i}^{s-1} \bar{h}_t^{(jn+i)} v_t}{p_s^T v_s}$;
5. End
6. $v_{jn+i+m} = Av_{jn+i} - \sum_{s=(j-1)n+i}^{jn+i+m-1} \bar{h}_s^{(jn+i)} v_s$;
7. End
8. End

We split the range of the s loop and split the summations appearing in lines 4 and 6 to obtain

Lanczos Procedure Version 3.

1. For $j = 0, 1, 2, \dots$
2. For $i = 1, 2, \dots, n$
3. For $s = (j-1)n + i, \dots, (j-1)n + n$
4. $\bar{h}_s^{(jn+i)} = \frac{p_s^T Av_{jn+i} - p_s^T \sum_{t=(j-1)n+i}^{s-1} \bar{h}_t^{(jn+i)} v_t}{p_s^T v_s}$;

5. End
6. If $i \leq n - m + 1$
7. For $s = jn + 1, \dots, jn + i + m - 1$
8.
$$\bar{h}_s^{(jn+i)} = \frac{p_s^T A v_{jn+i} - p_s^T \left(\sum_{t=(j-1)n+i}^{(j-1)n+n} \bar{h}_t^{(jn+i)} v_t + \sum_{t=jn+1}^{s-1} \bar{h}_t^{(jn+i)} v_t \right)}{p_s^T v_s};$$

9. End
10. Else
11. For $s = jn + 1, \dots, jn + n$
12.
$$\bar{h}_s^{(jn+i)} = \frac{p_s^T A v_{jn+i} - p_s^T \left(\sum_{t=(j-1)n+i}^{(j-1)n+n} \bar{h}_t^{(jn+i)} v_t + \sum_{t=jn+1}^{s-1} \bar{h}_t^{(jn+i)} v_t \right)}{p_s^T v_s};$$

13. End
14. For $s = (j+1)n + 1, \dots, (j+1)n + i + m - n - 1$
15.
$$\bar{h}_s^{(jn+i)} = \left(p_s^T A v_{jn+i} - p_s^T \left(\sum_{t=(j-1)n+i}^{(j-1)n+n} \bar{h}_t^{(jn+i)} v_t + \sum_{t=jn+1}^{jn+n} \bar{h}_t^{(jn+i)} v_t + \sum_{t=(j+1)n+1}^{s-1} \bar{h}_t^{(jn+i)} v_t \right) \right) / p_s^T v_s;$$

16. End
17. End
18. If $i \leq n - m$
19.
$$v_{jn+i+m} = A v_{jn+i} - \left(\sum_{s=(j-1)n+i}^{(j-1)n+n} \bar{h}_s^{(jn+i)} v_s + \sum_{s=jn+1}^{jn+i+m-1} \bar{h}_s^{(jn+i)} v_s \right);$$

20. Else
21.
$$v_{jn+i+m} = A v_{jn+i} - \left(\sum_{s=(j-1)n+i}^{(j-1)n+n} \bar{h}_s^{(jn+i)} v_s + \sum_{s=jn+1}^{jn+n} \bar{h}_s^{(jn+i)} v_s + \sum_{s=(j+1)n+1}^{jn+i+m-1} \bar{h}_s^{(jn+i)} v_s \right);$$

22. End
23. End
24. End

Since we have assumed that any quantity with non-positive subscript is zero, we can add a conditional control "if $j \geq 1$ " to lines 3 - 5 without changing their meanings. Moreover, by shifting the s -loops properly and replacing p_s with the right-hand side of (1a), we have

Lanczos Procedure Version 4.

1. For $j = 0, 1, 2, \dots$
2. For $i = 1, 2, \dots, n$
3. For $s = i, \dots, n$ and $j \geq 1$
4.
$$\bar{h}_{(j-1)n+s}^{(jn+i)} = \frac{\hat{u}_s^T A^j v_{jn+i} - \hat{u}_s^T A^{j-1} \sum_{t=i}^{s-1} \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t}}{\hat{u}_s^T A^{j-1} v_{(j-1)n+s}};$$

5. End
6. If $i \leq n - m + 1$
7. For $s = 1, \dots, i + m - 1$
8.
$$\bar{h}_{jn+s}^{(jn+i)} = \frac{\hat{u}_s^T A^{j+1} v_{jn+i} - \hat{u}_s^T A^j \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} v_{jn+t} \right)}{\hat{u}_s^T A^j v_{jn+s}};$$

9. End
10. Else
11. For $s = 1, \dots, n$
12.
$$\bar{h}_{jn+s}^{(jn+i)} = \frac{\hat{u}_s^T A^{j+1} v_{jn+i} - \hat{u}_s^T A^j \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} v_{jn+t} \right)}{\hat{u}_s^T A^j v_{jn+s}};$$

13. End
14. For $s = 1, \dots, i - n + m - 1$

15.
$$\bar{h}_{(j+1)n+s}^{(jn+i)} = \left(\hat{u}_s^T A^{j+2} v_{jn+i} - \hat{u}_s^T A^{j+1} \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t} + \sum_{t=1}^n \bar{h}_{jn+t}^{(jn+i)} v_{jn+t} + \sum_{t=1}^{s-1} \bar{h}_{(j+1)n+t}^{(jn+i)} v_{(j+1)n+t} \right) \right) / \hat{u}_s^T A^{j+1} v_{(j+1)n+s};$$

16. End

17. End

18. If $i \leq n - m$

19.
$$v_{jn+i+m} = Av_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} v_{(j-1)n+s} + \sum_{s=1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} v_{jn+s} \right);$$

20. Else

21.
$$v_{jn+i+m} = Av_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} v_{(j-1)n+s} + \sum_{s=1}^n \bar{h}_{jn+s}^{(jn+i)} v_{jn+s} + \sum_{s=1}^{i+m-n-1} \bar{h}_{(j+1)n+s}^{(jn+i)} v_{(j+1)n+s} \right);$$

22. End

23. End

24. End

Based on the formulas listed in the above procedure, we are now ready to give a transpose-free procedure to compute \bar{H}_ν . For the purpose, we introduce auxiliary vectors ϕ_{jn+i} and ψ_{jn+i} defined by

$$\phi_{jn+i} = A^{j+1} v_{jn+i}, \quad \psi_{jn+i} = A^j v_{jn+i}, \quad (9)$$

for $i = 1, 2, \dots, n$ and $j = 0, 1, 2, \dots$. Set

$$\phi_l = \psi_l = 0$$

for $l \leq 0$ for convenience. We will find that the coefficients $\bar{h}_l^{(jn+i)}$ can be expressed in terms of these vectors which in turn can be updated recursively. In fact, we have

$$\begin{aligned} \bar{h}_{(j-1)n+s}^{(jn+i)} &= \frac{\hat{u}_s^T A^j v_{jn+i} - \hat{u}_s^T A^{j-1} \sum_{t=i}^{s-1} \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t}}{\hat{u}_s^T A^{j-1} v_{(j-1)n+s}} \\ &= \frac{\hat{u}_s^T \psi_{jn+i} - \hat{u}_s^T \sum_{t=i}^{s-1} \bar{h}_{(j-1)n+t}^{(jn+i)} \psi_{(j-1)n+t}}{\hat{u}_s^T \psi_{(j-1)n+s}}, \end{aligned}$$

for $1 \leq i \leq n, 1 \leq j$ and $i \leq s \leq n$ (line 4 of version 4);

$$\begin{aligned} \bar{h}_{jn+s}^{(jn+i)} &= \frac{\hat{u}_s^T A^{j+1} v_{jn+i} - \hat{u}_s^T A^j \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} v_{jn+t} \right)}{\hat{u}_s^T A^j v_{jn+s}} \\ &= \frac{\hat{u}_s^T \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} \phi_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} \psi_{jn+t} \right)}{\hat{u}_s^T \psi_{jn+s}}, \end{aligned}$$

for $1 \leq i \leq n - m + 1, 0 \leq j$ and $1 \leq s \leq i + m - 1$ (line 8 of version 4);

$$\begin{aligned} \bar{h}_{jn+s}^{(jn+i)} &= \frac{\hat{u}_s^T A^{j+1} v_{jn+i} - \hat{u}_s^T A^j \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} v_{jn+t} \right)}{\hat{u}_s^T A^j v_{jn+s}} \\ &= \frac{\hat{u}_s^T \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} \phi_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} \psi_{jn+t} \right)}{\hat{u}_s^T \psi_{jn+s}} \end{aligned}$$

for $n - m + 2 \leq i \leq n, 0 \leq j$ and $1 \leq s \leq n$ (line 12 of version 4);

$$\begin{aligned}
\bar{h}_{(j+1)n+s}^{(jn+i)} &= \left(\hat{u}_s^T A^{j+2} v_{jn+i} - \hat{u}_s^T A^{j+1} \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} v_{(j-1)n+t} + \sum_{t=1}^n \bar{h}_{jn+t}^{(jn+i)} v_{jn+t} + \right. \right. \\
&\quad \left. \left. \sum_{t=1}^{s-1} \bar{h}_{(j+1)n+t}^{(jn+i)} v_{(j+1)n+t} \right) \right) / \hat{u}_s^T A^{j+1} v_{(j+1)n+s} \\
&= \left(\hat{u}_s^T A \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} A \phi_{(j-1)n+t} + \sum_{t=1}^n \bar{h}_{jn+t}^{(jn+i)} \phi_{jn+t} + \right. \right. \\
&\quad \left. \left. \sum_{t=1}^{s-1} \bar{h}_{(j+1)n+t}^{(jn+i)} \psi_{(j+1)n+t} \right) \right) / \hat{u}_s^T \psi_{(j+1)n+s}
\end{aligned}$$

for $n - m + 2 \leq i \leq n, 0 \leq j$ and $1 \leq s \leq i + m - n - 1$ (line 15 of version 4).

The vectors ϕ_l and ψ_l themselves can be updated as follows, by adaptation of line 19 of version 4,

$$\begin{aligned}
\psi_{jn+i+m} &= A^j v_{jn+i+m} \\
&= A^{j+1} v_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} A^j v_{(j-1)n+s} + \sum_{s=1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} A^j v_{jn+s} \right) \\
&= \phi_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} \phi_{(j-1)n+s} + \sum_{s=1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} \psi_{jn+s} \right), \\
\phi_{jn+i+m} &= A^{j+1} v_{jn+i+m} \\
&= A \phi_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} A \phi_{(j-1)n+s} + \sum_{s=1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} \phi_{jn+s} \right),
\end{aligned}$$

for $1 \leq i \leq n - m$ and $0 \leq j$; and by adaptation of line 21 of version 4:

$$\begin{aligned}
\psi_{jn+i+m} &= A^{j+1} v_{jn+i+m} \\
&= A \phi_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} A \phi_{(j-1)n+s} + \sum_{s=1}^n \bar{h}_{jn+s}^{(jn+i)} \phi_{jn+s} + \right. \\
&\quad \left. \sum_{s=n+1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} \psi_{jn+s} \right), \\
\phi_{jn+i+m} &= A^{j+2} v_{jn+i+m} = A \psi_{jn+i+m},
\end{aligned}$$

for $n - m + 1 \leq i \leq n$ and $0 \leq j$.

We use the above formulas to eliminate the vectors v_l from version 4 above to obtain the following.

Lanczos Procedure Version 5.

1. For $j = 0, 1, 2, \dots$
2. For $i = 1, 2, \dots, n$
3. For $s = i, \dots, n$ and $j \geq 1$
4.
$$\bar{h}_{(j-1)n+s}^{(jn+i)} = \frac{\hat{u}_s^T \psi_{jn+i} - \hat{u}_s^T \sum_{t=i}^{s-1} \bar{h}_{(j-1)n+t}^{(jn+i)} \psi_{(j-1)n+t}}{\hat{u}_s^T \psi_{(j-1)n+s}};$$
5. End
6. If $i \leq n - m + 1$
7. For $s = 1, \dots, i + m - 1$
8.
$$\bar{h}_{jn+s}^{(jn+i)} = \frac{\hat{u}_s^T \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} \phi_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} \psi_{jn+t} \right)}{\hat{u}_s^T \psi_{jn+s}};$$
9. End

10. Else
11. For $s = 1, \dots, n$
12.
$$\bar{h}_{jn+s}^{(jn+i)} = \frac{\hat{u}_s^T \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} \phi_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} \psi_{jn+t} \right)}{\hat{u}_s^T \psi_{jn+s}};$$

13. End
14. For $s = 1, \dots, i + m - n - 1$
15.
$$\bar{h}_{(j+1)n+s}^{(jn+i)} = \left(\hat{u}_s^T A \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} A \phi_{(j-1)n+t} + \sum_{t=1}^n \bar{h}_{jn+t}^{(jn+i)} \phi_{jn+t} + \sum_{t=1}^{s-1} \bar{h}_{(j+1)n+t}^{(jn+i)} \psi_{(j+1)n+t} \right) \right) / \hat{u}_s^T \psi_{(j+1)n+s};$$

16. End
17. End
18. If $i \leq n - m$
19.
$$\psi_{jn+i+m} = \phi_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} \phi_{(j-1)n+s} + \sum_{s=1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} \psi_{jn+s} \right);$$

20.
$$\phi_{jn+i+m} = A \phi_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} A \phi_{(j-1)n+s} + \sum_{s=1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} \phi_{jn+s} \right);$$

21. Else
22.
$$\psi_{jn+i+m} = A \phi_{jn+i} - \left(\sum_{s=i}^n \bar{h}_{(j-1)n+s}^{(jn+i)} A \phi_{(j-1)n+s} + \sum_{s=1}^n \bar{h}_{jn+s}^{(jn+i)} \phi_{jn+s} + \sum_{s=n+1}^{i+m-1} \bar{h}_{jn+s}^{(jn+i)} \psi_{jn+s} \right);$$

23.
$$\phi_{jn+i+m} = A \psi_{jn+i+m};$$

24. End
25. End
26. End

in which lines 6 - 17 are equivalent to

Fragment 6.

1. $l_{min} = \min\{n, i + m - 1\};$
2. For $s = 1, \dots, l_{min}$
3.
$$\bar{h}_{jn+s}^{(jn+i)} = \frac{\hat{u}_s^T \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} \phi_{(j-1)n+t} + \sum_{t=1}^{s-1} \bar{h}_{jn+t}^{(jn+i)} \psi_{jn+t} \right)}{\hat{u}_s^T \psi_{jn+s}};$$

4. End
5. For $s = 1, \dots, i + m - n - 1$
6.
$$\bar{h}_{(j+1)n+s}^{(jn+i)} = \left(\hat{u}_s^T A \phi_{jn+i} - \hat{u}_s^T \left(\sum_{t=i}^n \bar{h}_{(j-1)n+t}^{(jn+i)} A \phi_{(j-1)n+t} + \sum_{t=1}^n \bar{h}_{jn+t}^{(jn+i)} \phi_{jn+t} + \sum_{t=1}^{s-1} \bar{h}_{(j+1)n+t}^{(jn+i)} \psi_{(j+1)n+t} \right) \right) / \hat{u}_s^T \psi_{(j+1)n+s};$$

7. End

The procedure version 5 can be further simplified by noting that the summations in the numerators of lines 4, 8, 12, 15 of version 5 are exactly the leading terms in the summations in lines 19, 20, 22. So we can accumulate the summations one term at a time into temporary vectors e_ψ , e_ϕ , $e_{A\psi}$, and use the partial sums directly in lines 4, 8, 12, 15 as they are generated. This avoids effectively having to accumulate the summations multiple times. The result is the following.

Lanczos Procedure Version 7.

1. For $j = 0, 1, 2, \dots$
2. For $i = 1, 2, \dots, n$
3. $e_\psi = \psi_{jn+i}; \quad e_\phi = \phi_{jn+i};$
4. For $s = i, \dots, n$ and $j \geq 1$
5.
$$\bar{h}_{(j-1)n+s}^{(jn+i)} = \frac{\hat{u}_s^T e_\psi}{\hat{u}_s^T \psi_{(j-1)n+s}};$$

```

6.            $e_\psi = e_\psi - \bar{h}_{(j-1)n+s}^{(jn+i)} \psi_{(j-1)n+s};$ 
7.            $e_\phi = e_\phi - \bar{h}_{(j-1)n+s}^{(jn+i)} \phi_{(j-1)n+s};$ 
8.         End
9.          $l_{min} = \min\{n, i + m - 1\};$ 
10.         $e_\psi = e_\psi; \quad e_{A\phi} = Ae_\phi;$ 
11.        For  $s = 1, \dots, l_{min}$ 
12.           $\bar{h}_{jn+s}^{(jn+i)} = \frac{\hat{u}_s^T e_\psi}{\hat{u}_s^T \psi_{jn+s}};$ 
13.           $e_\psi = e_\psi - \bar{h}_{jn+s}^{(jn+i)} \psi_{jn+s};$ 
14.           $e_{A\phi} = e_{A\phi} - \bar{h}_{jn+s}^{(jn+i)} \phi_{jn+s};$ 
15.        End
16.        For  $s = 1, \dots, i + m - n - 1$ 
17.           $\bar{h}_{(j+1)n+s}^{(jn+i)} = \frac{\hat{u}_s^T e_{A\phi}}{\hat{u}_s^T \psi_{(j+1)n+s}};$ 
18.           $e_{A\phi} = e_{A\phi} - \bar{h}_{(j+1)n+s}^{(jn+i)} \psi_{(j+1)n+s};$ 
19.        End
20.        If  $i \leq n - m$ 
21.           $\psi_{jn+i+m} = e_\psi;$ 
22.           $\phi_{jn+i+m} = e_{A\phi};$ 
23.        Else
24.           $\psi_{jn+i+m} = e_{A\phi};$ 
25.           $\phi_{jn+i+m} = A\psi_{jn+i+m};$ 
26.        End
27.      End
28. End

```

Since the value of e_ϕ will no longer be used after line 10 in each double (i, j) -loop, we can store the value of $e_{A\phi}$ in e_ϕ to save memory. Furthermore, since the new vectors ϕ_{jn+i+m} and ψ_{jn+i+m} are computed at the end of each (i, j) -loop, the values of e_ϕ and e_ψ again can be stored in ϕ_{jn+i+m} and ψ_{jn+i+m} respectively. Moreover, we introduce a variable c_{jn+i} defined by $c_{jn+i} = \hat{u}_i^T \psi_{jn+i}$ to save some computational cost, where $i = 1, 2, \dots, n; j = 0, 1, \dots$. Thus, we arrive at the following algorithm which produces \bar{H}_ν without using A^T .

Algorithm 3.1 Transpose-free Multiple Lanczos Procedure *Given m right starting vectors $\{\hat{v}_l\}_{l=1}^m$ and n left starting vectors $\{\hat{u}_l\}_{l=1}^n$ with $m \leq n$. Suppose the assumption of Theorem 2.3 holds. The following algorithm computes the matrix \bar{H}_ν in Theorem 2.3.*

```

1. Compute vectors  $\{v_l\}_{l=1}^m$  such that  $v_l - \text{span}\{\hat{u}_1, \dots, \hat{u}_{l-1}\}$  and  $v_l \in \text{span}\{\hat{v}_1, \dots, \hat{v}_l\}$ .
2. Set  $\psi_l = v_l$  for  $l = 1, \dots, m$  and compute  $\phi_l = A\psi_l, c_l = \hat{u}_l^T \psi_l$ .
3. For  $j = 0, 1, 2, \dots$ 
4.   For  $i = 1, 2, \dots, n$ 
5.      $\psi_{jn+i+m} = \psi_{jn+i}; \quad \phi_{jn+i+m} = \phi_{jn+i};$ 
6.     For  $s = i, \dots, n$  and  $j \geq 1$ 
7.        $\bar{h}_{(j-1)n+s}^{(jn+i)} = \hat{u}_s^T \psi_{jn+i+m} / c_{(j-1)n+s};$ 
8.        $\psi_{jn+i+m} = \psi_{jn+i+m} - \bar{h}_{(j-1)n+s}^{(jn+i)} \psi_{(j-1)n+s};$ 
9.        $\phi_{jn+i+m} = \phi_{jn+i+m} - \bar{h}_{(j-1)n+s}^{(jn+i)} \phi_{(j-1)n+s};$ 
10.    End
11.     $l_{min} = \min\{n, i + m - 1\};$ 
12.     $\psi_{jn+i+m} = \phi_{jn+i+m}; \quad \phi_{jn+i+m} = A\psi_{jn+i+m};$ 
13.    For  $s = 1, \dots, l_{min}$ 
14.       $\bar{h}_{jn+s}^{(jn+i)} = \hat{u}_s^T \psi_{jn+i+m} / c_{jn+s};$ 
15.       $\psi_{jn+i+m} = \psi_{jn+i+m} - \bar{h}_{jn+s}^{(jn+i)} \psi_{jn+s};$ 
16.       $\phi_{jn+i+m} = \phi_{jn+i+m} - \bar{h}_{jn+s}^{(jn+i)} \phi_{jn+s};$ 

```

17. *End*
 18. *For* $s = 1, \dots, i + m - n - 1$
 19. $\bar{h}_{(j+1)n+s}^{(jn+i)} = \hat{u}_s^T \phi_{jn+i+m} / c_{(j+1)n+s};$
 20. $\phi_{jn+i+m} = \phi_{jn+i+m} - \bar{h}_{(j+1)n+s}^{(jn+i)} \psi_{(j+1)n+s};$
 21. *End*
 22. *If* $i \geq n - m + 1$
 23. $\psi_{jn+i+m} = \phi_{jn+i+m};$
 24. $\phi_{jn+i+m} = A\psi_{jn+i+m};$
 25. *End*
 26. $c_{jn+i+m} = \hat{u}_{i+m-n[(i+m-1)/n]}^T \psi_{jn+i+m};$
 27. *End*
 28. *End*

It is often the case in practice that the norms $\|\phi_l\|_2$ and $\|\psi_l\|_2$ become very large or very small as the algorithm progresses and instability can occur in this situation. So, it is necessary to normalize the vectors ϕ_l and ψ_l in order to make the algorithm more practicable. One way to do so is that we redefine the variables $c_l, \bar{h}_l^{(jn+i)}, \psi_l$ and ϕ_l in Algorithm 3.1 as follows

$$\begin{aligned}
 \bar{\mathbf{h}}_l^{(jn+i)} &\stackrel{\text{def}}{=} \bar{h}_l^{(jn+i)} \|\psi_l\|_2 / \|\psi_{jn+i}\|_2, & \omega_l &\stackrel{\text{def}}{=} \psi_l / \|\psi_l\|_2, \\
 \pi_l &\stackrel{\text{def}}{=} \phi_l / \|\psi_l\|_2, & b_l &\stackrel{\text{def}}{=} c_l / \|\psi_l\|_2.
 \end{aligned} \tag{10}$$

With these new definitions, Algorithm 3.1 becomes

Algorithm 3.2 Scaled Transpose-free Multiple Lanczos Procedure *Given* m *right starting vectors* $\{\hat{v}_l\}_{l=1}^m$ *and* n *left starting vectors* $\{\hat{u}_l\}_{l=1}^n$ *with* $m \leq n$. *Suppose the assumption of Theorem 2.3 holds. The following algorithm computes an* $\nu \times (\nu - m)$ *band matrix* $\bar{\mathbf{H}}_\nu = (\bar{\mathbf{h}}_{ij})_{i=1, \dots, \nu; j=1, \dots, \nu-m}$ *with* $\bar{\mathbf{h}}_{ij} = \bar{\mathbf{h}}_i^{(j)}$ *for* $\bar{m}_j \leq i \leq j + m; \bar{\mathbf{h}}_{ij} = 0$ *otherwise. $\bar{\mathbf{H}}_\nu$ is similar to* H_ν *computed in Algorithm 3.1.*

1. *Compute vectors* $\{v_l\}_{l=1}^m$ *such that* $v_l = \text{span}\{\hat{u}_1, \dots, \hat{u}_{l-1}\}$ *and* $v_l \in \text{span}\{\hat{v}_1, \dots, \hat{v}_l\}$.
 2. *Set* $\omega_l = v_l / \|v_l\|_2$ *for* $l = 1, 2, \dots, m$ *and compute* $\pi_l = A\omega_l, b_l = \hat{u}_l^T \omega_l$.
 3. *For* $j = 0, 1, 2, \dots$
 4. *For* $i = 1, 2, \dots, n$
 5. $\omega_{jn+i+m} = \omega_{jn+i}; \pi_{jn+i+m} = \pi_{jn+i};$
 6. *For* $s = i, \dots, n$ *and* $j \geq 1$
 7. $\bar{\mathbf{h}}_{(j-1)n+s}^{(jn+i)} = \hat{u}_s^T \omega_{jn+i+m} / b_{(j-1)n+s};$
 8. $\omega_{jn+i+m} = \omega_{jn+i+m} - \bar{\mathbf{h}}_{(j-1)n+s}^{(jn+i)} \omega_{(j-1)n+s};$
 9. $\pi_{jn+i+m} = \pi_{jn+i+m} - \bar{\mathbf{h}}_{(j-1)n+s}^{(jn+i)} \pi_{(j-1)n+s};$
 10. *End*
 11. $l_{\min} = \min\{n, i + m - 1\};$
 12. $\omega_{jn+i+m} = \pi_{jn+i+m}; \pi_{jn+i+m} = A\pi_{jn+i+m};$
 13. *For* $s = 1, \dots, l_{\min}$
 14. $\bar{\mathbf{h}}_{jn+s}^{(jn+i)} = \hat{u}_s^T \omega_{jn+i+m} / b_{jn+s};$
 15. $\omega_{jn+i+m} = \omega_{jn+i+m} - \bar{\mathbf{h}}_{jn+s}^{(jn+i)} \omega_{jn+s};$
 16. $\pi_{jn+i+m} = \pi_{jn+i+m} - \bar{\mathbf{h}}_{jn+s}^{(jn+i)} \pi_{jn+s};$
 17. *End*
 18. *For* $s = 1, \dots, i + m - n - 1$
 19. $\bar{\mathbf{h}}_{(j+1)n+s}^{(jn+i)} = \hat{u}_s^T \pi_{jn+i+m} / b_{(j+1)n+s};$
 20. $\pi_{jn+i+m} = \pi_{jn+i+m} - \bar{\mathbf{h}}_{(j+1)n+s}^{(jn+i)} \omega_{(j+1)n+s};$
 21. *End*
 22. *If* $i \geq n - m + 1$
 23. $\omega_{jn+i+m} = \pi_{jn+i+m};$

| <i>item</i> | <i>average count Algorithm 3.2</i> | <i>average count Lanczos Version 1</i> |
|--|--|---|
| Matrix vector products | $1 + \frac{m}{n}$ | 2 |
| Saxpy's | $2(m+n) - \frac{m(m-1)}{2n}$ | $m+n+1$ |
| Dot products | $m+n+2$ | $m+n+3$ |
| Total Storage beyond $A, \bar{\mathbf{H}}_\nu$ | $(2m+3n+2)N + m+n$ | $\frac{2(m+n+2)N}{2} + \frac{(m+n)^2}{2} + m+n$ |

Table 1: Average cost per column computation of Algorithm 3.2 and its storage requirement, compared to the Lanczos Procedure Version 1, where in the latter we assume the results of matrix-vector products Av_l and dot products $p_s^T v_t$ are saved for re-use.

24. $\pi_{jn+i+m} = A\omega_{jn+i+m};$
25. *End*
26. $\bar{\mathbf{h}}_{jn+i+m}^{(jn+i)} = \|\omega_{jn+i+m}\|_2;$
27. $\omega_{jn+i+m} = \omega_{jn+i+m} / \bar{\mathbf{h}}_{jn+i+m}^{(jn+i)};$
28. $\pi_{jn+i+m} = \pi_{jn+i+m} / \bar{\mathbf{h}}_{jn+i+m}^{(jn+i)};$
29. $b_{jn+i+m} = \hat{u}_{i+m-n[(i+m-1)/n]}^T \omega_{jn+i+m};$
30. *End*
31. *End*

We remark that we did not compute $\bar{h}_{jn+i+m}^{(jn+i)}$ in Algorithm 3.1 since it is equal to 1 there. However, it has been scaled in Algorithm 3.2 and therefore we can not ignore it. Moreover, by (10) and the definitions of the matrices \bar{H}_ν and $\bar{\mathbf{H}}_\nu$, they are related by

$$\bar{\mathbf{H}}_\nu = \Lambda_\nu \bar{H}_\nu \Lambda_{\nu-m}^{-1}, \quad (11)$$

where $\Lambda_l = \text{diag}\{\|\psi_1\|_2, \|\psi_2\|_2, \dots, \|\psi_l\|_2\}$.

Each pass through the j loop in Algorithm 3.2 calculates n columns of $\bar{\mathbf{H}}_\nu$ and involves $m+n$ matrix-vector multiplications, $n(m+n+2)$ dot products, $2n(m+n) - \frac{1}{2}m(m-1)$ saxpy's, and $2n$ scalar-vector multiplications, ignoring lower order effects. We list the average cost to compute each new column in Table 3. Moreover, in addition to the matrices A and $\bar{\mathbf{H}}_\nu$, the data $\{\pi_{(j-1)n+i}, \dots, \pi_{jn+i+m}\}$, $\{\omega_{(j-1)n+i}, \dots, \omega_{jn+i+m}\}$, $\{\hat{u}_1, \dots, \hat{u}_n\}$ and $\{b_{(j-1)n+i}, \dots, b_{jn+i+m-1}\}$ of storage are required in the process of each (i, j) -loop. 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 1. Of course Version 1 is not a version that one would actually use for computation, but its costs and storage requirements approximate that of the ABFH method when no deflation or look-ahead occur.

4 A Transpose-free Version of the MPVL Method

Recall from Theorem 2.4 that

$$F_l(z) = \begin{bmatrix} D_n^T \eta \\ 0_{(l-n) \times n} \end{bmatrix}^T (zI - H_l)^{-1} \begin{bmatrix} \rho \\ 0_{(l-m) \times m} \end{bmatrix}$$

is an l -th Padé approximant of the transfer function $F(z)$, where H_l is the $l \times l$ principal block of \bar{H}_ν . Since $D_n = U_n^T V_n$ and $\hat{U} = U_n \eta$ from §2, we have

$$F_l(z) = \begin{bmatrix} V_n^T \hat{U} \\ 0_{(l-n) \times n} \end{bmatrix}^T (zI - H_l)^{-1} \begin{bmatrix} \rho \\ 0_{(l-m) \times m} \end{bmatrix}.$$

Let \mathbf{H}_l be the $l \times l$ principal block of $\bar{\mathbf{H}}_\nu$. Because of (11), (9) and (10),

$$\begin{aligned}
F_l(z) &= \begin{bmatrix} V_n^T \hat{U} \\ 0_{(l-n) \times n} \end{bmatrix}^T (zI - \Lambda_l^{-1} \mathbf{H}_l \Lambda_l)^{-1} \begin{bmatrix} \rho \\ 0_{(l-m) \times m} \end{bmatrix} \\
&= \begin{bmatrix} (V_n \Lambda_n^{-1})^T \hat{U} \\ 0_{(l-n) \times n} \end{bmatrix}^T (zI - \mathbf{H}_l)^{-1} \begin{bmatrix} \Lambda_m \rho \\ 0_{(l-m) \times m} \end{bmatrix} \\
&= \begin{bmatrix} (\Psi_n \Lambda_n^{-1})^T \hat{U} \\ 0_{(l-n) \times n} \end{bmatrix}^T (zI - \mathbf{H}_l)^{-1} \begin{bmatrix} \Lambda_m \rho \\ 0_{(l-m) \times m} \end{bmatrix} \\
&= \begin{bmatrix} \Omega_n^T \hat{U} \\ 0_{(l-n) \times n} \end{bmatrix}^T (zI - \mathbf{H}_l)^{-1} \begin{bmatrix} \Lambda_m \rho \\ 0_{(l-m) \times m} \end{bmatrix}.
\end{aligned} \tag{12}$$

Here $\Psi_n = [\psi_1, \psi_2, \dots, \psi_n]$, $\Omega_n = [\omega_1, \omega_2, \dots, \omega_n]$ and $\Lambda_m = \text{diag}\{\|\psi_1\|_2, \dots, \|\psi_m\|_2\} \equiv \text{diag}\{\|v_1\|_2, \dots, \|v_m\|_2\}$ by the definition of Λ_m and (9). Thus, the function $F_l(z)$ is expressed in terms of the quantities in Algorithm 3.2. Regarding the $m \times m$ matrix ρ , it can be computed in Step 1 in Algorithm 3.2 via a modified two-sided Gram-Schmidt-type process [16, 5]. We can now present a transpose-free implementation of the MPVL method in the following algorithm.

Algorithm 4.1 Transpose-free MPVL (TFMPVL) *Given m right starting vectors $\{\hat{v}_l\}_{l=1}^m$ and n left starting vectors $\{\hat{u}_l\}_{l=1}^n$ with $m \leq n$. Suppose the assumption of Theorem 2.3 holds. The following algorithm computes an l -th Padé approximant $F_l(z)$ of the transfer function $F(z)$.*

1. Compute ρ via a modified two-sided Gram-Schmidt-type process
2. Run $l \equiv jn + i$ steps of the transpose-free Lanczos process with multiple starting vectors (Algorithm 3.2) to obtain Λ_m, Ω_n and \mathbf{H}_l .
3. Compute $F_l(z)$ according to (12).

This algorithm requires about $(1 + m/n)l$ matrix vector products to get an l -th Padé approximant $F_l(z)$. When $m < n$, it is cheaper in terms of matrix vector products than MPVL which needs $2l$ matrix vector products to get $F_l(z)$.

5 Numerical Experiments

The first example is the DC operating point problem from [15, 21] and the test matrix was obtained from [18]. It is a representative of full-scale commercial design at Bell Laboratories. The problem is typically formulated as a system of equations which represents the application of Kirchoff's current and voltage laws at various points in the circuit. The coefficient matrix, denoted WATSON3 in [18], is of order 124×125 with 780 nonzero entries. In this example, we artificially generate the matrix-valued function $F(z)$ defined in (7). The matrix A is the matrix obtained from the WATSON3 matrix by deleting the last column. \hat{U} and \hat{V} are matrices of order 124×2 with independent and identically distributed (iid) entries from a normal distribution with mean 0 and variance 1 ($N(0, 1)$). Figure 1 plots the frequency responses $|F^{(1,1)}(i\omega)|$ of the system and its 4-th, 8-th and 16-th Padé approximants computed by Algorithm 3.2, where $F^{(1,1)}(z)$ denotes the (1,1)-th scalar component of the matrix-valued function $F(z)$ and where $i = \sqrt{-1}$. Note that the 16-th Padé approximant has been very close to the response of the original system.

Our second example is the 120×120 system from [11, 12] which describes the effects of a magnetic actuator on the radial tracking arm of a portable compact disc player. The matrix A in this example is sparse with nonzero entries only on its diagonal and anti-diagonal. The (input) matrix \hat{V} contains two columns which respectively correspond to the voltages applied to the radial arm and lens actuators. The (output) matrix \hat{U} contains two columns which respectively correspond to the positions of the radial arm and the focusing lens [11]. In this example, we compare our TFMPVL method to an MPVL method using a two-side Lanczos procedure. Since $\hat{U}^T \hat{V} \approx 0$, the system is unstable when approximated with methods based on the

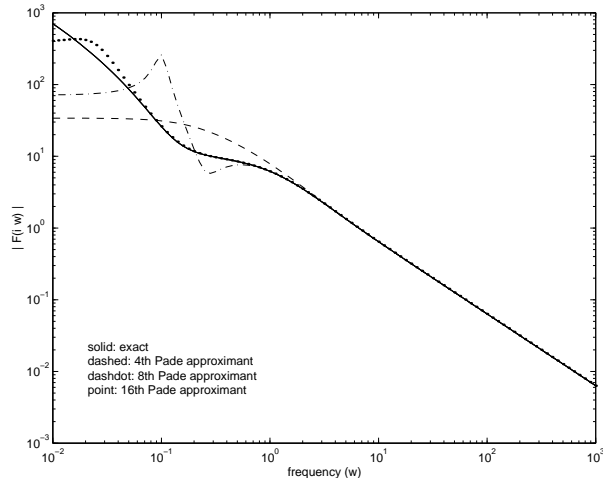


Figure 1: Frequency responses for the first example.

standard Lanczos procedure, see [9, 11, 12]. The difficulty applies to TFMPVL and MPVL. They almost suffer breakdown in their first two iterations due to, say, in the case of TFMPVL, b_1 and b_2 near zero. Recall that b_1 and b_2 are the dot products of \hat{V} -columns with \hat{U} -columns. To avoid the near-zero denominators b_1 and b_2 , we expanded the column-size of \hat{U} by adding some random vectors to \hat{U} . More precisely, we used the following \hat{U} matrix in our experiment for both TFMPVL and MPVL

$$\hat{U}^{exp} = [R_1, \hat{u}_1, R_2, \hat{u}_2],$$

where R_1 and R_2 are $N \times (n/2 - 1)$ matrices with iid entries from $N(0,1)$ and \hat{u}_1 and \hat{u}_2 are the first and second columns of the original \hat{U} matrix respectively. Thus, b_1 and b_2 are now the dot products of \hat{V} -columns with random vectors. Moreover, by injecting random vectors into this CD player system in this way, both algorithms seem to be able to generate stable partial realizations for the system. Another remedy of employing the implicit restart technique to stabilize the Lanczos generated model has already been suggested in [9, 11, 12] and our idea here seems to be a good alternative.

We computed the 30-th and 34-th Padé approximants $F_{30}(z)$ and $F_{34}(z)$ with $n = 20$ left starting vectors by the TFMPVL and MPVL algorithms for the matrix-valued function

$$F(z) = \left(\hat{U}^{exp} \right)^T (zI - A)^{-1} \hat{V}.$$

The magnitudes of the $(n/2, 1)$ -th, $(n/2, 2)$ -th, $(n, 1)$ -th and $(n, 2)$ -th scalar components of $F_{30}(i\omega)$, $F_{34}(i\omega)$ and $F(i\omega)$ are showed in Figures 2 and 3. According to Theorem 2.4, $F_{30}(z)$ and $F_{34}(z)$ satisfy that $F_{30}(z) = F(z) + O(z^{-17})$ and $F_{34}(z) = F(z) + O(z^{-19})$. In this experiment, TFMPVL used 34 and 38 matrix-vector products with A to get $F_{30}(z)$ and $F_{34}(z)$ respectively, and MPVL used 38 to get $F_{30}(z)$.

From both experiments, we can see that the computed Padé approximants via TFMPVL significantly capture the original systems at high frequency, but not at low frequency. It is because that Algorithm 3.2 was derived based on the Taylor expansion of $F(z)$ about $z = \infty$. Versions of the algorithm based on Taylor expansion of $F(z)$ about other z -values can be obtained similarly.

6 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. Besides avoiding the need for carrying the transpose of the matrix A , this method may also reduce the average number of matrix-vector products per vector generated from 2 to $1 + m/n$, where m, n are the number of right, left starting vectors (respectively). The

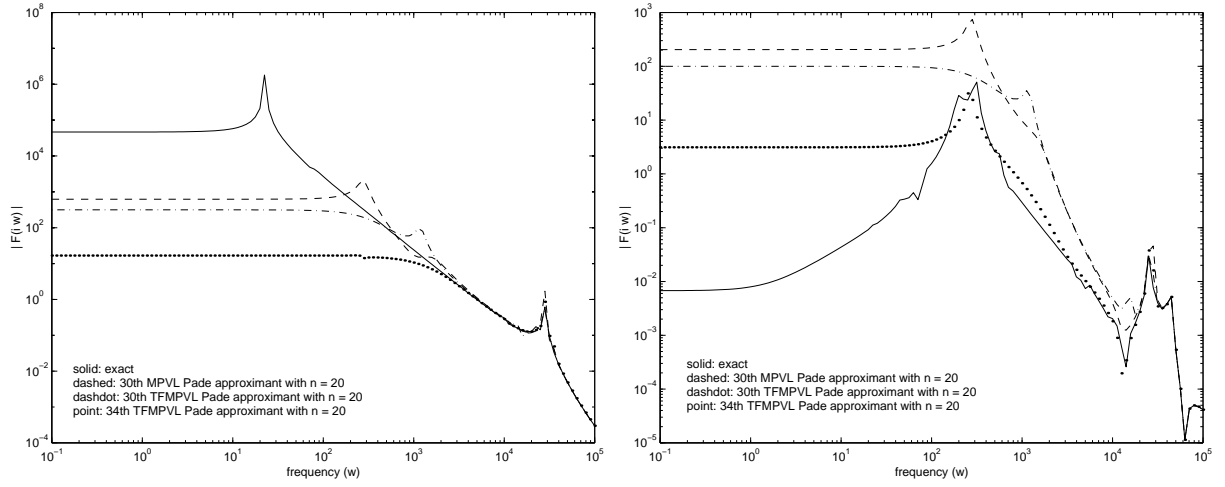


Figure 2: Frequency responses for the second example. (a) Radial arm input to radial arm position output (the $(n/2, 1)$ -th entry of $F(z)$); (b) Lens actuator input to radial arm position output (the $(n/2, 2)$ -th entry of $F(z)$)

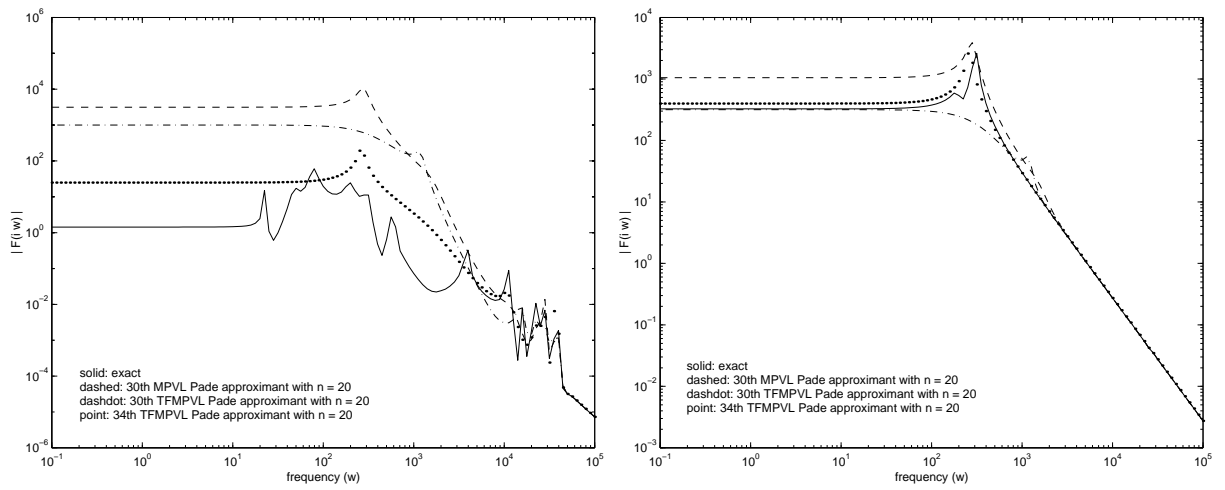


Figure 3: Frequency responses for the second example. (a) Radial arm input to lens position output (the $(n, 1)$ -th entry of $F(z)$); (b) Lens actuator input to lens position output (the $(n, 2)$ -th entry of $F(z)$)

method has been applied to the problem of computing a Padé approximation to a given transfer function. Since for the Padé approximation, the Lanczos vectors V_l is not needed, the method we have proposed generated only the banded coefficient matrix H_l , but it is an easy matter to generate V_l as well. Numerical experiments indicate that the method's numerical properties appear to be as favorable as those for the original two-sided Lanczos 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 Dr. Roland W. Freund for his help in our experiments.

References

- [1] J. Aliaga, D. Boley, R. Freund and V. Hernández, *A Lanczos-type method for multiple starting vectors*, Numerical Analysis Manuscript No. 96-18, Bell Laboratories, Murray Hill, NJ, 1996. (Available on WWW at <http://cm.bell-labs.com/cs/doc/96>)
- [2] D. L. Boley, *Krylov Space Methods on State-Space Control Models*, Circ. Syst. & Signal Proc. 13 #6, pp 733-758, 1994.
- [3] 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.
- [4] 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.
- [5] —, *Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm*, Numerical Analysis Manuscript 94-11, Bell Laboratories, Murray Hill, NJ, 1995.
- [6] R. Freund, *A transpose-free quasi-minimum residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470-482.
- [7] —, *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.
- [8] —, *Circuit simulation techniques based on Lanczos-type algorithms*, Numerical Analysis Manuscript 96-19, Bell Laboratories, Murray Hill, NJ, Oct. 1996.
- [9] K. Gallivan, E. Grimme, D. Sorensen and P. Van Dooren, *On some modifications of the Lanczos algorithm and the relation with Padé approximations*, appear in Proceedings of 1995 International Congress on Industrial and Applied Mathematics.
- [10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, second edition, The Johns Hopkins University Press (Baltimore), 1989.
- [11] E. Grimme, *An implicitly restarted Lanczos method for the model reduction of stable, large-scale systems*, M.S. thesis in electrical engineering, the Graduate College of the University of Illinois at Urbana-Champaign, 1994.

- [12] E. Grimme, D. Sorensen and P. Van Dooren, *Model reduction of state space systems via an implicitly restarted Lanczos method*, Numerical Algorithms, 1995.
- [13] M. H. Gutknecht, *Variants of BiCGStab for matrices with complex spectrum*, SIAM J. Sci. Comput. 14, 1020-1033, 1993.
- [14] ———, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numerica, 271-397, 1997.
- [15] W. D. McQuain, R. C. Melville, C. J. Ribbens and L. T. Watson, *Preconditioned iterative methods for sparse linear algebra problems arising in circuit simulation*, Comput. Math. Appl., 27, 25-45, 1994.
- [16] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matr. Anal., 13 (1992), pp. 567-593.
- [17] Y. Saad, *Iterative methods for sparse linear systems*, PWS Publishing Company, 1996.
- [18] ———, *Sparskit*, <http://math.nist.gov/MatrixMarket/data/SPARSKIT/circuits/circuits.html>.
- [19] G. L. G. Sleijpen and D. R. Fokkema, *BiCGSTAB(k) for linear equations involving insymmetric 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. Sosonkina, L. T. Watson, R. K. Kapania and H. F. Walker, *A new adaptive GMRES algorithm for achieving high accuracy*, Numer. Linear Algebra Appl., to appear.
- [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*, UCLA CAM Reports 97-15, 1997. (Available on WWW at <http://www.math.ucla.edu/applied/cam/index.html>)