SIAM J. MATRIX ANAL. APPL.
Vol. 42, No. 2, pp. 635–658

PARALLEL PRONY'S METHOD WITH MULTIVARIATE MATRIX PENCIL APPROACH AND ITS NUMERICAL ASPECTS*

NELA BOSNER†

Abstract. Prony's method is a standard tool exploited for solving many imaging and data analysis problems that result in parameter identification in sparse exponential sums $f(k) = \sum_{j=1}^{M} c_j e^{-2\pi i\langle t_j,k\rangle}$, $k \in \mathbb{Z}^d$, where the parameters are pairwise different $\{t_j\}_{j=1}^M \subset [0,1)^d$, and $\{c_j\}_{j=1}^M \subset \mathbb{C} \setminus \{0\}$ are nonzero. The focus of our investigation is on a Prony's method variant based on a multivariate matrix pencil approach. The method constructs matrices $S_1, \dots, S_d$ from the sampling values, and their simultaneous diagonalization yields the parameters $\{t_j\}_{j=1}^M$. The parameters $\{c_j\}_{j=1}^M$ are computed as the solution of an linear least squares problem, where the matrix of the problem is determined by $\{t_j\}_{j=1}^M$. Since the method involves independent generation and manipulation of a certain number of matrices, there is an intrinsic capacity for parallelization of the whole computational process on several levels. Hence, we propose a parallel version of the Prony's method in order to increase its efficiency. The tasks concerning the generation of matrices are divided among the block of threads of the graphics processing unit (GPU) and the central processing unit (CPU), where heavier load is put on the GPU. From the algorithmic point of view, the CPU is dedicated to the more complex tasks of computing the singular value decomposition, the eigendecomposition, and the solution of the least squares problem, while the GPU is performing matrix–matrix multiplications and summations. With careful choice of algorithms solving the subtasks, the load between CPU and GPU is balanced. Besides the parallelization techniques, we are also concerned with some numerical issues, and we provide detailed numerical analysis of the method in case of noisy input data. Finally, we performed a set of numerical tests which confirm superior efficiency of the parallel algorithm and consistency of the forward error with the results of numerical analysis.

Key words. Prony's method, parallel algorithm, efficient GPU-CPU implementation, numerical analysis

AMS subject classifications. 15A06, 15A18, 15A20, 15A60, 65F99, 65G99, 65Y05, 65Y20, 65Z05

DOI. 10.1137/20M1343658

1. Introduction.

Prony's method is a standard tool for parameter identification in sparse exponential sums

$$f(k) = \sum_{j=1}^{m} c_j e^{-2\pi\imath\langle t_j,k\rangle}, \quad k \in \mathbb{Z}^d, \tag{1.1}$$

for small m, occurring in many imaging and data analysis problems. The task is to determine pairwise different parameters $\{t_j\}_{j=1}^m \subset [0,1)^d$ and nonzero coefficients $\{c_j\}_{j=1}^m \subset \mathbb{C} \setminus \{0\}$ from relatively few sampling values. There are several variants of Prony's method; see, e.g., [18], [19], [11], as well as similar methods like matrix pencil method [8], MUSIC [23], and ESPRIT [21], [1]. The focus of our investigation is on a Prony's method variant based on a multivariate matrix pencil approach presented in [3], since its implementation is more elegant from the numerical point of view. Our

*Received by the editors June 5, 2020; accepted for publication (in revised form) by J. M. Bardsley February 18, 2021; published electronically April 26, 2021.
https://doi.org/10.1137/20M1343658
Funding: The work of the author was supported by the Croatian Science Foundation grant IP-2019-04-6268.
†Department of Mathematics, Faculty of Science, University of Zagreb, Zagreb, 10000, Croatia (nela.bosner@math.hr).

intention is to discuss the numerical aspects of this method, as well as to propose a more efficient way of its implementation.

The details of the algorithm follow. For $n \in \mathbb{N}$, let $I_n = \{0, \ldots, n\}^d$ with fixed ordering of the elements, let us build the $N \times N$ matrices

$$T = [f(k-h)]_{k,h \in I_n}, \quad T_\ell = [f(k-h+e_\ell)]_{k,h \in I_n}, \ \ell = 1, \ldots, d$$

from sampling values, where $N = \# I_n = (n+1)^d$, and e_ℓ is the ℓth unit vector. If T has rank m, then one should compute reduced singular value decomposition (SVD)

$$T = U \Sigma V^*, \tag{1.2}$$

where $\Sigma \in \mathbb{R}^{m \times m}$ is diagonal positive definite, $U, V \in \mathbb{C}^{N \times m}$ and $U^*U = V^*V = I \in \mathbb{C}^{m \times m}$. Let us define the set of $m \times m$ matrices

$$S_\ell = U^* T_\ell V \Sigma^{-1}, \quad \ell = 1, \ldots, d. \tag{1.3}$$

Theorem 2.1 from [3] states that if $n \geq \frac{K_d}{\min_{i \neq j} \|z_i - z_j\|}$, where K_d is an absolute constant that only depends on d and is specified in [11], then T has rank equal to the number of terms m in the exponential sum (1.1), and $S_1, \ldots, S_d$ are simultaneously diagonalizable. Furthermore, let us define, as in [3]

$$z_j = e^{-2\pi\iota t_j} = [e^{-2\pi\iota t_j(1)} \ \ldots \ e^{-2\pi\iota t_j(d)}]^T, \quad j = 1, \ldots, m, \tag{1.4}$$

where $t_j(\ell)$ denotes the ℓth component of the vector t_j, and

$$z_j^k = e^{-2\pi\iota \langle t_j, k \rangle}, \quad j = 1, \ldots, m, \ k \in I_n;$$

then any regular matrix W that simultaneously diagonalizes $S_1, \ldots, S_d$ yields a permutation τ on $\{1, \ldots, m\}$ such that

$$W^{-1} S_\ell W = \operatorname{diag}(\langle z_{\tau(1)}, e_\ell \rangle, \ldots, \langle z_{\tau(m)}, e_\ell \rangle), \quad \ell = 1, \ldots, d. \tag{1.5}$$

From the proof of this theorem it is also easy to see that for

$$A = [z_j^k]_{j=1,\ldots,m, \ k \in I_n} \in \mathbb{C}^{m \times N}, \quad f = [f(k)]_{k \in I_n} \in \mathbb{C}^N, \quad c = [c_j]_{j=1,\ldots,m} \in \mathbb{C}^m, \tag{1.6}$$

equation $f(k) = \sum_{j=1}^m c_j z_j^k$, $k \in I_n$, can be written in the matrix form as

$$f = A^T c. \tag{1.7}$$

Theorem 2.1 in [3] represents a core of Prony's algorithm. Simultaneous diagonalization of $S_1, \ldots, S_d$ is obtained by finding W that diagonalizes a random linear combination

$$C_\mu = \sum_{\ell=1}^d \mu_\ell S_\ell, \quad \mu \in \mathbb{C}^d,$$

for random μ such that $\|\mu\|_2 = 1$. The system of equations (1.7) is solved as a linear least squares problem. Putting all this together results in Algorithm 1.1, as proposed in [3].

From the numerical point of view this algorithm involves several nontrivial issues, such as computing the SVD and the spectral decomposition (diagonalization), as well

Algorithm 1.1. Prony's method with the multivariate matrix pencil approach.

```
Input: $f(k)$, $k \in I$
Output: $t_{\tau(1)}, \dots, t_{\tau(m)}$ and $c_{\tau(1)}, \dots, c_{\tau(m)}$

1: Build the matrices $T$, $T_\ell$, $\ell = 1, \dots, d$, and the vector $f$;
2: Compute the reduced SVD of $T$;
3: Compute the matrices $S_1, \dots, S_d$;
4: Choose random $\mu \in \mathbb{S}_{\mathbb{C}}^{d-1}$ and compute a matrix $W$ that diagonalizes $C_\mu$;
5: Use $W$ to simultaneously diagonalize $S_1, \dots, S_d$ and reconstruct $z_{\tau(1)}, \dots, z_{\tau(m)}$;
6: Compute $t_{\tau(j)}$ as the principal value of $\log(z_{\tau(j)})$, $j = 1, \dots, m$;
7: Solve $\operatorname{argmin}_c \|A^T c - f\|_2$ to recover $c_{\tau(1)}, \dots, c_{\tau(m)}$;
```

as establishing the numerical rank of a matrix. Both decompositions are based on iterative methods and require a respectable number of operations, which depends cubically on the matrix dimension. On the other hand, in Prony's algorithm they are not in the same position, since dimensions of the matrices are different. The spectral decomposition is performed on a matrix with small dimension m and does not pose a problem. The most time consuming task in the algorithm is the full SVD of the matrix T, whose dimension is N. In the case when the parameters t_j are close to each other, and particularly when $d > 1$, N can be quite large. Closely related with the SVD is rank determination, which in the floating point arithmetics is not a straightforward task. Especially, when we introduce noise into the sampling values, we cannot expect that the matrix T has rank m any more, but there will be a significant drop in the singular values. It seems that in many applications of Prony's method, the full SVD is computed with all N singular values, and then the singular values are examined. This is a wasteful procedure, since it spends a large number of operations for a small number of required singular values and vectors.

For computing the reduced SVD, more suitable are Lanczos bidiagonalization method [4], [17], and the block power method for SVD [2]. The Lanczos method computes partial bidiagonalization of a matrix, and if it completes it produces left and right singular subspaces and determines the rank. The power method computes left and right singular subspaces for a fixed number of dominant singular values. In cases when we know an overestimation of the rank, and if we combine the method with a rank revealing factorization on the projected matrix, we can obtain the reduced SVD.

On the other hand, as described earlier, Prony's method involves independent generation and manipulation of a certain number of matrices; hence there is intrinsic capacity for parallelization of the whole computational process on several levels, using CPU and GPU.

The goal of this paper is to give a numerical aspect of Prony's algorithm, its numerical analysis in the case of the noisy input data, and to propose algorithmic enhancements as well as its parallel implementation. The parallel algorithm is going to exploit a hybrid CPU-GPU environment in order to produce the most efficient implementation.

2. Choice of efficient SVD method.

Let us first make a short analysis of execution times for all steps in Algorithm 1.1. Table 1 displays times expressed in seconds, for specific parameters d, m, n, and N.

TABLE 1
Execution time of the algorithm subtasks.

d	m	n	N	t_{T,T_ℓ}	t_{SVD}	t_{S_ℓ}	t_{C_μ}	t_{eig}	$t_{z,t}$	t_A	t_{LS}	t_{total}
3	5	20	9261	48.12	166.55	0.1100	0.000040	0.000071	0.000038	0.01207	0.000968	214.80
3	10	20	9261	47.30	232.13	0.1199	0.000046	0.000124	0.000089	0.02406	0.002209	279.57
3	15	20	9261	47.34	232.52	0.1290	0.000046	0.000221	0.000064	0.03597	0.003891	280.04
3	20	20	9261	47.93	211.00	0.1495	0.000046	0.000359	0.000135	0.04800	0.006585	259.13

Notation of the time fractions is as follows:
- t_{T,T_ℓ} — time spent on generating matrices T and T_ℓ, $\ell = 1, \dots, d$, and vector f
- t_{SVD} — time spent on computing full SVD of T
- t_{S_ℓ} — time spent on generating matrices S_ℓ
- t_{C_μ} — time spent on computing matrix C_μ
- t_{eig} — time spent on diagonalizing C_μ
- $t_{z,t}$ — time spent on computing vectors z_j and t_j
- t_A — time spent on computing matrix A
- t_{LS} — time spent on solving least squares problem $\operatorname{argmin}_c \|A^T c - f\|_2$
- t_{total} — total execution time

As we can see from the table, the dominant task is full SVD of the matrix T which consumes about 80% of the total execution time in the presented example for $d = 3$ and $n = 20$.

To avoid computation of the full SVD of a large matrix and to reduce time spent on finding the reduced SVD, the first step in producing an efficient Prony's algorithm is the right choice of the SVD algorithm. Instead of computing the full SVD, the algorithm should be able to determine the rank of the matrix and to compute the singular subspaces directly.

2.1. Lanczos bidiagonalization method.

The Lanczos bidiagonalization method is suitable for rank determination in the case when we have no information on rank. The idea of using Lanczos bidiagonalization in Prony-like methods is not new. It was first used in [25] and [26] for ESPRIT method and was also exploited in [20] and [22] with observation that some sort of reorthogonalization is required in order to obtain numerical stability of the whole process. Hence, this method has some numerical issues that we will refer later. As described in [5, subsection 9.3.3], suppose $U^*AV = B$ represents the full bidiagonalization of $A \in \mathbb{C}^{m\times n}$ $(m \geq n)$, with $U = [u_1, \dots, u_n] \in \mathbb{C}^{m\times n}$ such that $U^*U = I_n$, $V = [v_1, \dots, v_n] \in \mathbb{C}^{n\times n}$ such that $V^*V = I_n$, and

$$B = \begin{bmatrix} \alpha_1 & \beta_2 & & \cdots & 0 \\ 0 & \alpha_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_n \\ 0 & \cdots & & 0 & \alpha_n \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

By comparing columns in the equations $AV = UB$ and $A^*U = VB^T$ we obtain the method displayed in Algorithm 2.1.

In the matrix form the algorithm can be rewritten as

$$AV_i = U_i B_i \tag{2.1}$$

$$A^*U_i = V_{i+1} B_{i,i+1}^T, \qquad i = 1, 2, \dots, \tag{2.2}$$

Algorithm 2.1. Lanczos bidiagonalization method.

```
Input: $A \in \mathbb{C}^{m\times n}$, $p_1 \in \mathbb{C}^n$
Output: rank $r$, $U_r \in \mathbb{C}^{m\times r}$, $V_{r+1} \in \mathbb{C}^{n\times(r+1)}$, $B_{r,r+1} \in \mathbb{R}^{r\times(r+1)}$
 1: $p_1$ = given vector;
 2: $\beta_1 = \|p_1\|_2$; $v_1 = p_1/\beta_1$; $u_0 = 0$; $i = 1$;
 3: while $\beta_i \neq 0$ do
 4:     $r_i = Av_i - \beta_i u_{i-1}$; $\alpha_i = \|r_i\|_2$;
 5:     if $\alpha_i = 0$ then
 6:         exit the loop;
 7:     end if
 8:     $u_i = r_i/\alpha_i$; $i = i+1$;
 9:     $p_i = A^* u_{i-1} - \alpha_{i-1} v_{i-1}$; $\beta_i = \|p_i\|_2$;
10:     if $\beta_i \neq 0$ then
11:         $v_i = p_i/\beta_i$;
12:     end if
13: end while
14: $r = i - 1$;
```

where $B_i = B(1:i, 1:i)$ is top $i \times i$ submatrix of B, $B_{i,i+1} = B(1:i, 1:i+1)$ is top $i \times (i+1)$ submatrix of B, $U_i = [u_1 \ \dots \ u_i]$, and $V_i = [v_1 \ \dots \ v_i]$. If $\alpha_{i+1} = 0$, then $\operatorname{span}\{Av_1, \dots, Av_{i+1}\} \subset \{u_1, \dots, u_i\}$ which implies rank deficiency. In that case B_{i+1} is singular with the last row equal to zero; hence (2.1) transforms into

$$AV_{i+1} = U_i B_{i,i+1},$$

which together with (2.2) shows that, like in [17],

$$A^*AV_{i+1} = V_{i+1} B_{i,i+1}^T B_{i,i+1}, \qquad AA^*U_i = U_i B_{i,i+1} B_{i,i+1}^T,$$

and $\sigma(B_{i,i+1}^T B_{i,i+1}) \subset \sigma(A^*A)$ and $\sigma(B_{i,i+1} B_{i,i+1}^T) \subset \sigma(AA^*)$, where $\sigma(X)$ denotes spectrum of the matrix X. So, $\alpha_{i+1} = 0$ implies that i nontrivial singular values of $B_{i,i+1}$ are also singular values of A. Suppose that

$$B_{i,i+1} = \bar{U}_B[\bar{\Sigma}_B, 0]\bar{V}_B^T, \quad \bar{U}_B \in \mathbb{R}^{i\times i},\ \bar{U}_B^T \bar{U}_B = I,\ \bar{V}_B \in \mathbb{R}^{(i+1)\times(i+1)},\ \bar{V}_B^T \bar{V}_B = I$$

is SVD of $B_{i,i+1}$ where $\bar{\Sigma}_B \in \mathbb{R}^{i\times i}$ is diagonal, with positive diagonal elements. Then, diagonal elements of $\bar{\Sigma}_B$ are also singular values of A, the columns of $U_i\bar{U}_B$ are left singular vectors of A, and the columns of $V_{i+1}\bar{V}_B(:, 1:i)$ are right singular vectors of A. The vector $V_{i+1}\bar{V}_B(:, i+1)$ is in the nullspace of A.

There is another stopping criteria for the Lanczos algorithm. When $\beta_{i+1} = 0$, (2.2) reduces to

$$A^*U_i = V_i B_i^T, \tag{2.3}$$

which together with (2.1) implies that singular values of B_i are also singular values of A. If

$$B_i = U_B \Sigma_B V_B^T, \quad U_B \in \mathbb{R}^{i\times i},\ U_B^T U_B = I,\ V_B \in \mathbb{R}^{i\times i},\ V_B^T V_B = I$$

is SVD of B_i, then columns of $U_i U_B$ are left singular vectors of A, and columns of $V_i V_B$ are right singular vectors of A.

It is interesting to investigate when and how the bidiagonalization is going to stop. Similar to the conclusion of Paige in [17], if $v_1 \in \mathcal{R}(A^*)$, then by line 9 in Algorithm 2.1 it follows that $v_i \in \mathcal{R}(A^*)$ for all i, meaning that all v_i are orthogonal to $\mathcal{N}(A)$. If the method stopped with $\alpha_{i+1} = 0$, then $AV_{i+1}\bar{V}_B(:, i+1) = 0$ implying that there exist a linear combination of $v_1,\ldots,v_{i+1}$ which is in $\mathcal{N}(A)$. This is a contradiction; hence in this case the algorithm has to stop with $\beta_{i+1} = 0$ for $i \le r$ where $r = \operatorname{rank}(A)$. Early stopping with $i < r$ can only occur if v_1 is a linear combination of i right singular vectors of A, belonging to nontrivial singular values. If $v_1 \notin \mathcal{R}(A^*)$ and the method stopped with $\beta_{i+1} = 0$, then by (2.3) and nonsingularity of B_i it follows that $V_i = A^*U_iB_i^{-T}$ and all $v_j \in \mathcal{R}(A^*)$ which is a contradiction. Hence in this case the algorithm has to stop with $\alpha_{i+1} = 0$ for $i \le r$. Early stopping can occur if v_1 is a linear combination of $i < r$ right singular vectors of A and a vector from the null space of A.

If we start the bidiagonalization algorithm with random vector p_1, there is a great chance that it would not belong to a space spanned by only a few right singular vectors and vectors from the nullspace. But nevertheless, if early stopping occurs there is a way to continue the process. First of all, we have to detect early stopping. In the case of $\alpha_{i+1} = 0$ we have to check that $\operatorname{span}\{u_1, \ldots, u_i\} = \mathcal{R}(A)$, and, respectively, that it is the orthogonal complement of $\mathcal{N}(A^*)$. It suffices to take a random vector y, orthogonalize it against $u_1, \ldots, u_i$ by $y = y - U_iU_i^*y$, and check whether $y \in \mathcal{N}(A^*)$. If the condition on y is satisfied we can conclude that we are finished, otherwise we stopped too early, and we can restart the algorithm with $u_{i+1} = y/\|y\|_2$. In case of $\beta_{i+1} = 0$ we have to check that $\operatorname{span}\{v_1, \ldots, v_i\} = \mathcal{R}(A^*)$, and, respectively, that it is the orthogonal complement of $\mathcal{N}(A)$. Hence, we take a random vector w, orthogonalize it against $v_1, \ldots, v_i$ by $w = w - V_iV_i^*w$, and check whether $w \in \mathcal{N}(A)$. If the condition on w is not satisfied we stopped too early, and we can proceed the algorithm with $v_{i+1} = w/\|w\|_2$. This way we can be sure that the algorithm stops for $i = r$, and we can determine the rank of A from dimensions of its bidiagonal factor.

When the Lanczos bidiagonalization method is implemented in the finite precision arithmetic on a computer, then there is several numerical issues that have to be taken under consideration. Let u be the unit roundoff, and let U_i, V_{i+1} and $B_{i,i+1}$ be matrices computed in the finite precision arithmetic by Algorithm 2.1, which stopped as described in the previous paragraph. Orthogonality among the columns of U_i and V_{i+1} is gradually lost, and as a consequence the rank of $B_{i,i+1}$ can be larger than r, since set of its singular values will contain false multiple copies of singular values even if they are single, or ghost singular values that appear between singular values of $B_{i,i+1}$. So, implementation of the Algorithm 2.1 should include full reorthogonalization, where each vector u_i and v_i should be explicitly orthogonalized against all previous vectors u_j and v_j, respectively, for $j = 1, \ldots, i-1$. This adds much to the algorithms operation count, but in the case when we do not expect the rank to be large it is not too expensive. In this case, by Theorem 5 in [12], we can expect that

$$\hat{U}_iA\hat{V}_{i+1}^* = B_{i,i+1} + E_{i,i+1},$$

where the columns of $\hat{U}_i$ form an orthonormal basis for $\operatorname{span}\{u_1, \ldots, u_i\}$, the columns of $\hat{V}_{i+1}$ form an orthonormal basis for $\operatorname{span}\{v_1, \ldots, v_{i+1}\}$, and the elements of $E_{i,i+1}$ are of order $\mathcal{O}(u\|A\|_2)$. This means that the stopping criteria has to be rephrased, instead for checking $\alpha_{i+1} = 0$ and $\beta_{i+1} = 0$ we will test for $\alpha_{i+1} \le \mathrm{tol}$ and $\beta_{i+1} \le \mathrm{tol}$. The tolerance is $\mathrm{tol} = u\|A\|_2$, or, in the case when we use the Lanczos method as a step of Prony's method for noisy input data, we can increase the tolerance to the perturbation magnitude in order to catch the drop of the singular values.

2.2. Block power method for SVD. In the case when we can obtain a slight overestimation of the rank, we can use a faster approach such as the block power method for SVD (see [2]). For a matrix $A \in \mathbb{C}^{m\times n}$ with rank r the goal is to find orthonormal $\bar{U} \in \mathbb{C}^{m\times r}$ and $\bar{V} \in \mathbb{C}^{n\times r}$ and a nonsingular matrix $Q \in \mathbb{C}^{r\times r}$ such that

$$A = \bar{U}Q\bar{V}^*.$$

The reduced SVD of A is then easily obtained from SVD of $Q = U_Q\Sigma V_Q^*$, so that columns of $U = \bar{U}U_Q$ represent left singular vectors, and columns of $V = \bar{V}V_Q$ represent right singular vectors of A. Our approach is to use the power method with rank determination, displayed in Algorithm 2.2.

Algorithm 2.2. Block power method for SVD.

```
Input: $A \in \mathbb{C}^{m\times n}$, overestimation of rank $r_0$, orthonormal $U_0 \in \mathbb{C}^{m\times r_0}$, $V_0 \in \mathbb{C}^{n\times r_0}$
Output: rank $r$, $U \in \mathbb{C}^{m\times r}$, $V \in \mathbb{C}^{n\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$
 1: $Q_0 = U_0^* A V_0$;
 2: $R_0 = AV_0 - U_0Q_0$;
 3: $k = 0$;
 4: while $\|R_k\|_F > \text{tol}\|A\|_F$ do
 5:     $k = k + 1$;
 6:     $\bar{U}_k = AV_{k-1}$;
 7:     Compute reduced QR factorization $\bar{U}_k = U_k R_{U_k}$;
 8:     $\bar{V}_k = A^* U_k$;
 9:     Compute reduced QR factorization with column pivoting $\bar{V}_k P_{V_k} = V_k R_{V_k}$;
10:     Determine new rank $r_k$ from $R_{V_k}$;
11:     $V_k = V_k(:, 1 : r_k)$, $Q_k^* = (R_{V_k} P_{V_k}^T)(1 : r_k, 1 : r_k)$;
12:     $R_k = AV_k - U_k Q_k$;
13: end while
14: Compute SVD $Q_k = U_{Q_k}\Sigma V_{Q_k}^*$;
15: $r = r_k$, $U = U_k U_{Q_k}$, $V = V_k V_{Q_k}$;
```

There is a slight difference between Algorithm 2.2 and the power method described in [2]. In lines 9 and 10 of Algorithm 2.2 there is QR factorization with column pivoting required for the rank determination instead of ordinary QR, and it has to be performed only once in the first step of the loop. Rank determination finds index i such that $R_{V_1}(i : r_0, i : r_0) = 0$, or in case of finite precision arithmetics $\|R_{V_1}(i : r_0, i : r_0)\|_F \leq tol\|R_{V_1}\|_F$. Then we take $r_1 = i - 1$. Since we are searching for a sharp drop in singular values, meaning that $\sigma_{r_1} \gg \sigma_{r_1+1}$, we expect that the power method will converge in only a few steps.

Even if all columns of V_0 are linear combination of only $k < r$ right singular vectors and vectors from the null space of A, then the QR factorization in line 7 of Algorithm 2.2 will produce U_1 whose columns consist of k left singular vectors and $r_0 - k$ vectors which almost always have components in the rest $r - k$ left singular vectors. Then, we will detect that $r_1 = r$, and the power method will produce the expected result. In finite precision arithmetics there is no problem with orthogonality like in Lanczos method since QR factorization implemented with Householder reflectors is numerically stable, and we are going to use the same tolerance $tol = u$, or higher in case of noisy data, for rank determination by QR factorization with pivoting.

Hence, the power method is a very convenient method for obtaining the reduced SVD in the case when we have an upper bound on the rank. Moreover, it employs mostly matrix-matrix multiplications which are the most efficient matrix operations, as well as QR factorizations which are implemented as efficient blocked algorithms, so we can expect that the SVD power method is the fastest SVD method especially for larger dimensions, and numerical tests are going to confirm that.

3. Parallel algorithm.

Prony's method involves independent generation and manipulation of a certain number of matrices, so, as mentioned earlier, we can do that simultaneously on several levels, employing both CPU and GPU. On the first level of parallelization, the tasks concerning generation of matrices is divided among the GPU's block of threads and the CPU, where heavier load is put on the GPU. On the second level, the individual threads are dealing with individual matrix elements.

From the algorithmic point of view, the CPU is dedicated to the more complex tasks of computing the SVD, eigendecompositions, and solving the least squares problem, while the GPU is performing matrix-matrix multiplications and summations of matrices. This is so, since multicore CPU, capable of executing several threads simultaneously, excels at computations which require frequent synchronizations, and at the algorithms that require complicated control flows. GPU is optimally exploited for large, massively parallel, and regular computations, where work is equally balanced among GPU cores and where all cores are constantly occupied in full capacity. With careful choice of the algorithms solving the subtasks, the load between CPU and GPU can be balanced. The basic scheme of the hybrid CPU-GPU algorithm is presented in Figure 1.

The main idea is to minimize communication between the CPU and the GPU. Therefore, CPU is going to store T, f, C_μ, and c, as well as auxiliary matrices U, V,

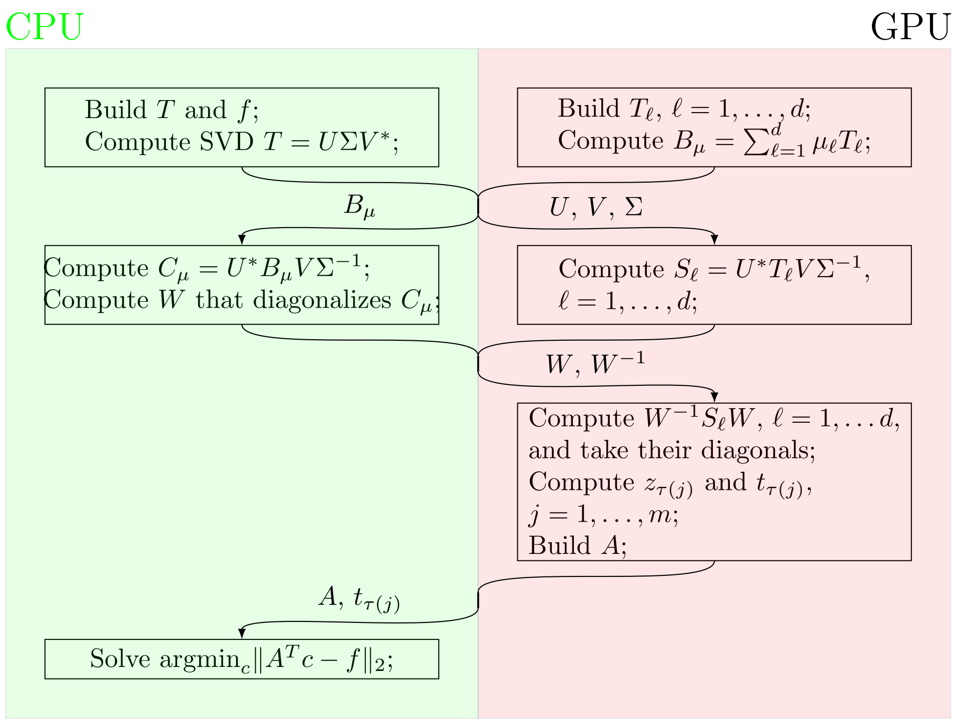
CPU
GPU
Build T and f;
Compute SVD $T = U\Sigma V^*$;
Build T_ℓ, $\ell = 1,\dots,d$;
Compute $B_\mu = \sum_{\ell=1}^{d} \mu_\ell T_\ell$;
B_μ
U, V, Σ
Compute $C_\mu = U^* B_\mu V \Sigma^{-1}$;
Compute W that diagonalizes C_μ;
Compute $S_\ell = U^* T_\ell V \Sigma^{-1}$, $\ell = 1,\dots,d$;
W, W^{-1}
Compute $W^{-1} S_\ell W$, $\ell = 1,\dots d$, and take their diagonals;
Compute $z_{\tau(j)}$ and $t_{\tau(j)}$, $j = 1,\dots,m$;
Build A;
A, $t_{\tau(j)}$
Solve $\operatorname{argmin}_c \|A^T c - f\|_2$;

FIG. 1. *The hybrid CPU-GPU parallel Prony's algorithm.*

and W, while B_μ, A, and $t_{\tau(j)}$ are going to be copied from the GPU. The GPU is going to store T_ℓ, B_μ, S_ℓ, A, and $t_{\tau(j)}$, while U, V, and W are going to be copied from the CPU. Since T_ℓ for $\ell = 1, \dots, d$ are stored on the GPU, S_ℓ has to be generated on the GPU as well. On the other side, the CPU is diagonalizing C_μ, so only this matrix is required, not all S_ℓ. To speed up the computation, we will perform preprocessing step of computing $B_\mu = \sum_{\ell=1}^{d} \mu_\ell T_\ell$ in parallel on the GPU which is going to be very fast and simultaneous with SVD computation on the CPU. This way only the matrix B_μ will be copied from the GPU to the CPU, instead of all S_ℓ matrices. C_μ is obtained from B_μ in the same way as S_ℓ from T_ℓ, after U, Σ, and V are obtained.

As we mentioned before, the CPU is performing reduced SVD computation of T and diagonalization of C_μ, and it is solving the problem of least squares $\operatorname{argmin}_c \|A^T c - f\|_2$. The GPU only computes the matrix B_μ, performs matrix-matrix multiplications with matrices T_ℓ and S_ℓ, and generates $z_{\tau(j)}$, $t_{\tau(j)}$ and the matrix A.

3.1. Parallelization on the CPU. Matrix-matrix multiplications are implemented by multithreaded BLAS library [14]. Only generation of the matrix T is explicitly parallelized, where T is partitioned in uniform block columns, and each CPU thread generates elements in one block column.

3.2. Parallelization on the GPU. On the GPU much more tasks are explicitly parallelized. First of all, each matrix T_ℓ is built by one block of threads on the GPU, where generations of elements are distributed among threads. B_μ is partitioned in uniform block columns, where each block of threads is building one block column, and generation of elements is distributed among threads. Diagonal extraction of each S_ℓ is performed by one block of threads, where access to the elements of diagonals, $z_{\tau(j)}$ and $t_{\tau(j)}$ is distributed among threads. Generation of A is distributed among threads of different blocks of threads, where each block processes one column. Matrix-matrix multiplications are implemented by cuBLAS [15], a library of parallel BLAS routines implemented on NVIDIA GPUs by CUDA [16]. There is a possibility to perform matrix multiplications that generate matrices S_ℓ in (1.3) and that diagonalize them in (1.5) in parallel, by exploiting CUDA streams or batched matrix-matrix multiplication kernels. Nevertheless, since the dimensions of involved matrices are rather small the benefit of such approach is negligible.

4. Numerical analysis.

One of the important topics of this paper is numerical analysis of Prony's method in the presence of noisy data. In real applications of the method input data $f(k)$, $k \in I$ may contain some perturbations, for example, due to measurement errors or rounding errors when placing data in computer memory. So, instead of $f(k)$ our input data are

$$\tilde{f}(k) = f(k) + \Delta f_k, \quad \text{such that } |\Delta f_k| \le \varepsilon |f(k)|, \tag{4.1}$$

for some relative error ε. In [18] sensitivity analysis of the approximate Prony method was presented but only for the coefficient c_j obtained as solution of a weighted linear least squares problem. In [22] first-order perturbations of the computed quantities in the ESPRIT method are derived, and in [8] the same is done for the matrix-pencil method. Our interest is focused on how size of ε influences the final result of the Algorithm 1.1 executed in exact arithmetic. Direct consequence of such choice of input data is that instead of matrices T and T_ℓ, $\ell = 1, \dots, d$, we are going to operate with matrices

$$\tilde{T} = T + \Delta T, \quad \tilde{T}_\ell = T_\ell + \Delta T_\ell, \tag{4.2}$$

such that

$$|\Delta T| \leq \varepsilon |T|, \quad |\Delta T_\ell| \leq \varepsilon |T_\ell|,$$

where these inequalities are componentwise, and consequently

$$\|\Delta T\|_F \leq \varepsilon \|T\|_F, \quad \|\Delta T_\ell\|_F \leq \varepsilon \|T_\ell\|_F, \tag{4.3}$$

where $\|\cdot\|_F$ is Frobenius matrix norm. Since we are computing the SVD of $\tilde{T}$, by Hoffman–Wielandt-type bound [9, subsection 3.2] we have bound on errors in singular values

$$\sqrt{\sum_{i=1}^{N} |\tilde{\sigma}_i - \sigma_i|^2} \leq \|\Delta T\|_F \leq \varepsilon \|T\|_F, \tag{4.4}$$

where σ_i are singular values of T, and $\tilde{\sigma}_i$ are singular values of $\tilde{T}$, both in nonincreasing order. Let,

$$\tilde{T} = \begin{bmatrix} \tilde{U} & \tilde{U}_\perp \end{bmatrix} \begin{bmatrix} \tilde{\Sigma} & \\ & \tilde{\Sigma}_\perp \end{bmatrix} \begin{bmatrix} \tilde{V}^* \\ \tilde{V}_\perp^* \end{bmatrix} = \tilde{U}\tilde{\Sigma}\tilde{V}^* + \tilde{U}_\perp \tilde{\Sigma}_\perp \tilde{V}_\perp^* = \tilde{U}\tilde{\Sigma}\tilde{V}^* + T_\perp \tag{4.5}$$

be the SVD of $\tilde{T}$, with $\tilde{U}, \tilde{V} \in \mathbb{C}^{N\times m}$, and $\tilde{\Sigma} \in \mathbb{R}^{m\times m}$ consisting of the m largest singular values $\tilde{\sigma}_i$. Further, since the last $N-m$ singular values of T are 0, from (4.4) it follows

$$\|T_\perp\|_F = \sqrt{\sum_{i=m+1}^{N} \tilde{\sigma}_i^2} \leq \varepsilon \|T\|_F. \tag{4.6}$$

This bound gives us criteria for numerical rank determination, i.e., detection of significant drop in singular values of the matrix $\tilde{T}$.

LEMMA 4.1. *Let $T \in \mathbb{C}^{N\times N}$ have rank m, and let $\tilde{T}$ be such that $\tilde{T} = T + \Delta T$ and $\|\Delta T\|_F \leq \varepsilon \|T\|_F$. Then*

$$\tilde{\sigma}_j \leq \tilde{\sigma}_{m+1} \leq \sqrt{\sum_{i=m+1}^{N} \tilde{\sigma}_i^2} \leq \varepsilon \|T\|_F \leq (\varepsilon + \mathcal{O}(\varepsilon^2))\|\tilde{T}\|_F \quad \textit{for } j \in \{m+1, \ldots, N\} \tag{4.7}$$

represents a bound on the singular values of $\tilde{T}$ which can be reduced to zero, and we can determine the numerical rank of $\tilde{T}$ being m, when m is the smallest integer satisfying (4.7).

LEMMA 4.2. *For T and $\tilde{T}$ satisfying conditions of Lemma* 4.1 *and* (4.5), *the following holds*

$$\tilde{U}\tilde{\Sigma}\tilde{V}^* = T + \Delta T_{SVD}, \tag{4.8}$$

where

$$\|\Delta T_{SVD}\|_F \leq 2\varepsilon \|T\|_F. \tag{4.9}$$

Proof. Combining (4.2) and (4.5) we obtain

$$\tilde{U}\tilde{\Sigma}\tilde{V}^* = T + \Delta T - T_\perp, \quad \Delta T_{SVD} = \Delta T - T_\perp,$$

where by (4.3) and (4.6) it is

$$\|\Delta T_{SVD}\|_F \le \|\Delta T\|_F + \|T_\perp\|_F \le 2\varepsilon\|T\|_F. \qquad \square$$

This is the backward error for SVD, and $\tilde{\tilde{T}} = \tilde{U}\tilde{\Sigma}\tilde{V}^*$ is a rank m matrix close to T whose reduced SVD we actually use in the algorithm.

The next step is generation of matrices S_ℓ, $\ell = 1, \dots, d$.

LEMMA 4.3. *Instead of* $S_\ell = U^*T_\ell V\Sigma^{-1}$ *we compute*

$$\tilde{S}_\ell = \tilde{U}^*\tilde{T}_\ell\tilde{V}\tilde{\Sigma}^{-1} = \tilde{U}^*T_\ell\tilde{V}\tilde{\Sigma}^{-1} + \Delta S_{\ell,1} \tag{4.10}$$

such that

$$\|\Delta S_{\ell,1}\|_F \le \frac{\varepsilon}{\tilde{\sigma}_m}\|T_\ell\|_F. \tag{4.11}$$

Proof. From (4.2) it follows

$$\tilde{U}^*\tilde{T}_\ell\tilde{V}\tilde{\Sigma}^{-1} = \tilde{U}^*T_\ell\tilde{V}\tilde{\Sigma}^{-1} + \tilde{U}^*\Delta T_\ell\tilde{V}\tilde{\Sigma}^{-1}, \quad \Delta S_{\ell,1} = \tilde{U}^*\Delta T_\ell\tilde{V}\tilde{\Sigma}^{-1},$$

and from (4.3) we have that

$$\|\Delta S_{\ell,1}\|_F \le \frac{1}{\tilde{\sigma}_m}\|\Delta T_\ell\|_F \le \frac{\varepsilon}{\tilde{\sigma}_m}\|T_\ell\|_F. \qquad \square$$

Further we need to determine a relationship between exact and computed singular vectors. Let $U_\perp$ and $V_\perp$ be orthonormal matrices whose columns span orthogonal complements of span$\{U\}$ and span$\{V\}$, respectively, where U and V are defined by (1.2). Then we can write

$$\tilde{U} = \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} U^* \\ U_\perp^* \end{bmatrix} \tilde{U} = U(U^*\tilde{U}) + U_\perp(U_\perp^*\tilde{U}), \tag{4.12}$$

$$\tilde{V} = \begin{bmatrix} V & V_\perp \end{bmatrix} \begin{bmatrix} V^* \\ V_\perp^* \end{bmatrix} \tilde{V} = V(V^*\tilde{V}) + V_\perp(V_\perp^*\tilde{V}). \tag{4.13}$$

Let us assume that all k columns of X are some choice of left singular vectors from U, that all k columns of $\tilde{X}$ are some choice of left singular vectors from $\tilde{U}$, and that we have the same assumption for Y and $\tilde{Y}$ regarding right singular vectors. By $X_\perp$ we denote the matrix whose columns are remaining $N-k$ columns of $\begin{bmatrix} U & U_\perp \end{bmatrix}$, and we define $Y_\perp$ in a similar way from $\begin{bmatrix} V & V_\perp \end{bmatrix}$. And finally, let τ_i $i = 1, \dots, k$ be the singular values of T that correspond to X and Y, and τ_i $i = k+1, \dots, N$ are all the rest. $\tilde{\tau}_i$ are similarly defined for singular values of $\tilde{\tilde{T}}$.

COROLLARY 4.4. *Let* X, $\tilde{X}$, Y, *and* $\tilde{Y}$ *be as defined in the previous paragraph; then*

$$\sqrt{\|X_\perp^*\tilde{X}\|_F^2 + \|Y_\perp^*\tilde{Y}\|_F^2} \le \frac{2\sqrt{2}\varepsilon\|T\|_F}{\delta}, \tag{4.14}$$

where

$$\delta = \min\left\{ \min_{\substack{i=1,\dots,k \\ j=1,\dots,N-k}} |\tau_i - \tilde{\tau}_{k+j}|, \min_{i=1,\dots,k} \tau_i \right\}. \tag{4.15}$$

Further, $\|X^*\tilde{X}\|_F = \|\cos\Theta(X,\tilde{X})\|_F$ *and* $\|X_\perp^*\tilde{X}\|_F = \|\sin\Theta(X,\tilde{X})\|_F$*, where* $\Theta(X,\tilde{X})$ *is diagonal matrix with canonical angles between subspaces span*$\{X\}$ *and span*$\{\tilde{X}\}$*. Further,* $\cos\Theta(X,\tilde{X}) \geq 0$ *and* $\sin\Theta(X,\tilde{X}) \geq 0$*. The same is valid for* Y *and* $\tilde{Y}$*.*

Proof. By Wedin's theorem [13, Theorem 3.3], [24] we have

$$\sqrt{\|X_\perp^*\tilde{X}\|_F^2 + \|Y_\perp^*\tilde{Y}\|_F^2} \leq \frac{\sqrt{\|(\tilde{\tilde{T}} - T)Y\|_F^2 + \|(\tilde{\tilde{T}}^* - T^*)X\|_F^2}}{\delta} \leq \frac{\sqrt{2}\|\Delta T_{SVD}\|_F}{\delta}$$
$$\leq \frac{2\sqrt{2}\varepsilon\|T\|_F}{\delta},$$

and the rest follows from the Wedin's result. ∎

Next, we are going to apply Corrolary 4.4 to the special choices of X, Y, and their perturbed versions, to determine relationship between $\tilde{S}_\ell$ and S_ℓ for every $\ell = 1,\ldots,d$.

LEMMA 4.5. *Let* U *and* V *be defined by* (1.2)*, and let* $\tilde{U}$ *and* $\tilde{V}$ *be defined by* (4.5)*. Let* $|U(:,i)^*\tilde{U}(:,i)| = \cos\theta_i$ *for* $\cos\theta_i \geq 0$*,* $i = 1,\ldots,m$ *so that there exist* α_i *such that* $e^{\iota\alpha_i}U(:,i)^*\tilde{U}(:,i) = \cos\theta_i$*, let* $|e^{\iota\alpha_i}V(:,i)^*\tilde{V}(:,i)| = \cos\psi_i$ *for* $\cos\psi_i \geq 0$*, so that* $e^{\iota\alpha_i}V(:,i)^*\tilde{V}(:,i) = \cos\psi_i e^{\iota\beta_i}$*,* $i = 1,\ldots,m$*. Then we have the following bounds*

$$U^*\tilde{U} = I + \Delta U, \quad \|\Delta U\|_F \leq \frac{2\sqrt{2m}\varepsilon\|T\|_F}{\delta_{\min}} + \mathcal{O}(\varepsilon^3), \tag{4.16}$$

where

$$\delta_{\min} = \min_{i=1,\ldots,m} \delta_i, \quad \delta_i = \min\left\{\min_{\substack{j=1,\ldots,N\\ j\neq i}} |\sigma_i - \tilde{\sigma}_j|, \sigma_i\right\} \tag{4.17}$$

and

$$V^*\tilde{V} = J + \Delta V_0, \quad \|\Delta V_0\|_F \leq \frac{2\sqrt{2m}\varepsilon\|T\|_F}{\delta_{\min}} + \mathcal{O}(\varepsilon^3), \tag{4.18}$$

for $J = diag(e^{\iota\beta_1},\ldots,e^{\iota\beta_m})$*.*

Proof. In the case when $X = U(:,i)$, $\tilde{X} = \tilde{U}(:,i)$, $Y = V(:,i)$ and $\tilde{Y} = \tilde{V}(:,i)$ we can conclude the following. Let $\theta_i = \Theta(U(:,i),\tilde{U}(:,i))$; then

$$|(U^*\tilde{U})(i,i)| = |U(:,i)^*\tilde{U}(:,i)| = \cos\theta_i = \sqrt{1-(\sin\theta_i)^2} = 1 - \frac{1}{2}(\sin\theta_i)^2 + \mathcal{O}((\sin\theta_i)^4),$$

and by Corollary 4.4 we have

$$\sin\theta_i = \|X_\perp^*\tilde{X}\|_F \leq \frac{2\sqrt{2}\varepsilon\|T\|_F}{\delta_i}, \quad \delta_i = \min\left\{\min_{\substack{j=1,\ldots,N\\ j\neq i}} |\sigma_i - \tilde{\sigma}_j|, \sigma_i\right\}.$$

We can scale columns of U by scalars $e^{\iota\alpha_i}$, with absolute value equal to 1, so that $U(:,i)^*\tilde{U}(:,i) = \cos\theta_i$ for all i. The same scaling applies to the columns of V in order that the columns of U and V remain singular vectors. Hence

$$(U^*\tilde{U})(i,i) = 1 + \Delta U(i,i), \quad |\Delta U(i,i)| \leq \frac{4\varepsilon^2\|T\|_F^2}{\delta_i^2} + \mathcal{O}(\varepsilon^4), \tag{4.19}$$

$$(V^*\tilde{V})(i,i) = e^{\iota\beta_i} + \Delta V_0(i,i), \quad |\Delta V_0(i,i)| \leq \frac{4\varepsilon^2\|T\|_F^2}{\delta_i^2} + \mathcal{O}(\varepsilon^4). \tag{4.20}$$

On the other hand,

$$\text{(4.21)}\qquad \sqrt{\sum_{\substack{j=1\\ j\neq i}}^{m}|(U^*\tilde{U})(j,i)|^2} \le \sqrt{\sum_{\substack{j=1\\ j\neq i}}^{m}|(U^*\tilde{U})(j,i)|^2 + \sum_{j=1}^{N-m}|(U_\perp^*\tilde{U})(j,i)|^2}$$
$$= \|X_\perp^*\tilde{X}\|_F \le \frac{2\sqrt{2}\varepsilon\|T\|_F}{\delta_i}.$$

So, from (4.19) and (4.21) we can conclude that

$$(U^*\tilde{U})(:,i) = e_i + \Delta U(:,i), \quad \|\Delta U(:,i)\|_2^2 \le \frac{8\varepsilon^2\|T\|_F^2}{\delta_i^2} + \frac{16\varepsilon^4\|T\|_F^4}{\delta_i^4} + \mathcal{O}(\varepsilon^6).$$

In the same way we can prove that

$$(V^*\tilde{V})(:,i) = e^{\iota\beta_i}\cdot e_i + \Delta V_0(:,i), \quad \|\Delta V_0(:,i)\|_2^2 \le \frac{8\varepsilon^2\|T\|_F^2}{\delta_i^2} + \frac{16\varepsilon^4\|T\|_F^4}{\delta_i^4} + \mathcal{O}(\varepsilon^6).$$

Putting all this together we obtain the result of the lemma. ☐

LEMMA 4.6. *Under assumptions of Lemma* 4.5 *the following bound holds:*

$$\text{(4.22)}\quad V^*\tilde{V} = I + \Delta V, \quad \|\Delta V\|_F \le \frac{g\sqrt{2m}\varepsilon\|T\|_F}{\delta_{\min}} + \mathcal{O}(\varepsilon^2), \quad \textit{for } g = \frac{3}{\sqrt{2m}} + 2\sigma_1.$$

Proof. Since from (4.4), (4.8), (4.12), and (4.13) we have

$$\begin{aligned}\tilde{\tilde{T}} &= \tilde{U}\tilde{\Sigma}\tilde{V}^* = (U(U^*\tilde{U}) + U_\perp(U_\perp^*\tilde{U}))(\Sigma+\Delta\Sigma)(V(V^*\tilde{V}) + V_\perp(V_\perp^*\tilde{V}))^*\\ &= (U(I+\Delta U) + U_\perp(U_\perp^*\tilde{U}))(\Sigma+\Delta\Sigma)(V(I+(J-I)+\Delta V_0) + V_\perp(V_\perp^*\tilde{V}))^*\\ &= U\Sigma V^* + U\Delta U\Sigma V^* + U_\perp(U_\perp^*\tilde{U})\Sigma V^* + U\Sigma(\bar{J}-I)V^* + U\Sigma\Delta V_0^*V^* +\\ &\quad U\Sigma(V_\perp^*\tilde{V})^*V_\perp^* + U\Delta\Sigma V^* + \mathcal{O}(\varepsilon^2)\\ &= T + \Delta T_{SVD},\end{aligned}$$

we can conclude that

$$\Delta V^* = \bar{J} - I + \Delta V_0^* = \Sigma^{-1}U^*\Delta T_{SVD}V - \Sigma^{-1}\Delta U\Sigma - \Sigma^{-1}\Delta\Sigma + \mathcal{O}(\varepsilon^2).$$

From Lemma 4.2, Lemma 4.5, (4.4), and (4.17) it follows

$$\begin{aligned}\|\Delta V\|_F &\le \frac{\|\Delta T_{SVD}\|_F}{\sigma_m} + \frac{\sigma_1\|\Delta U\|_F}{\sigma_m} + \frac{\|\Delta\Sigma\|_F}{\sigma_m} + \mathcal{O}(\varepsilon^2)\\ &\le \frac{2\varepsilon\|T\|_F}{\delta_{min}} + \frac{2\sqrt{2m}\sigma_1\varepsilon\|T\|_F}{\delta_{min}} + \frac{\varepsilon\|T\|_F}{\delta_{min}} + \mathcal{O}(\varepsilon^2)\\ &= \frac{(3+2\sqrt{2m}\sigma_1)\varepsilon\|T\|_F}{\delta_{min}} + \mathcal{O}(\varepsilon^2).\end{aligned}$$

The bound of the lemma is obtained directly from this result. ☐

LEMMA 4.7. *Let $U_\perp$ and $V_\perp$ be defined by* (4.12) *and* (4.13), *and let $\tilde{U}$ and $\tilde{V}$ be defined by* (4.5). *Then*

$$\|U_\perp^* \tilde{U}\|_F \le \frac{2\sqrt{2}\varepsilon\|T\|_F}{\delta_{T,\tilde{\tilde{T}}}}, \tag{4.23}$$

$$\|V_\perp^* \tilde{V}\|_F \le \frac{2\sqrt{2}\varepsilon\|T\|_F}{\delta_{T,\tilde{\tilde{T}}}}, \tag{4.24}$$

$$\delta_{T,\tilde{\tilde{T}}} = \min\left\{ \min_{\substack{i=1,\ldots,m \\ j=1,\ldots,N-m}} |\sigma_i - \tilde{\sigma}_{m+j}|, \sigma_m \right\} \ge \delta_{\min}. \tag{4.25}$$

Proof. Directly from Corollary 4.4 for $X = U$, $\tilde{X} = \tilde{U}$, $Y = V$, and $\tilde{Y} = \tilde{V}$. ◻

Now, we switch back to matrices S_ℓ, where by (4.10)

$$\begin{aligned} S_\ell &= U^* T_\ell V \Sigma^{-1}, \quad \ell = 1, \ldots, d, \\ \tilde{S}_\ell &= \tilde{U}^* \tilde{T}_\ell \tilde{V} \tilde{\Sigma}^{-1} = \tilde{U}^* T_\ell \tilde{V} \tilde{\Sigma}^{-1} + \Delta S_{\ell,1}, \quad \ell = 1, \ldots, d. \end{aligned}$$

Theorem 4.8. *The relationship between $\tilde{S}_\ell$ and S_ℓ for $\ell = 1, \ldots, d$ is characterized by the following expression:*

$$\tilde{S}_\ell = S_\ell + \Delta S_\ell,$$

$$\|\Delta S_\ell\|_F \le \frac{\varepsilon\|T_\ell\|_F}{\tilde{\sigma}_m}\left(1 + \frac{(1 + 4\sqrt{2} + (2+g)\sqrt{2m})\|T\|_F}{\delta_{\min}}\right) + \mathcal{O}(\varepsilon^2). \tag{4.26}$$

Proof. Again, from (4.12) and (4.13) it follows

$$\begin{aligned} \tilde{S}_\ell &= (U(U^*\tilde{U}) + U_\perp(U_\perp^*\tilde{U}))^* T_\ell (V(V^*\tilde{V}) + V_\perp(V_\perp^*\tilde{V}))\tilde{\Sigma}^{-1} + \Delta S_{\ell,1} \\ &= (\tilde{U}^*U)[U^*T_\ell V\Sigma^{-1}]\Sigma(V^*\tilde{V})\tilde{\Sigma}^{-1} + (\tilde{U}^*U_\perp)U_\perp^*T_\ell V(V^*\tilde{V})\tilde{\Sigma}^{-1} + \\ &\quad + (\tilde{U}^*U)U^*T_\ell V_\perp(V_\perp^*\tilde{V})\tilde{\Sigma}^{-1} + (\tilde{U}^*U_\perp)U_\perp^*T_\ell V_\perp(V_\perp^*\tilde{V})\tilde{\Sigma}^{-1} + \Delta S_{\ell,1}. \end{aligned}$$

If we define

$$\begin{aligned} \Delta S_{\ell,2} &= (\tilde{U}^*U_\perp)U_\perp^*T_\ell V(V^*\tilde{V})\tilde{\Sigma}^{-1}, \\ \Delta S_{\ell,3} &= (\tilde{U}^*U)U^*T_\ell V_\perp(V_\perp^*\tilde{V})\tilde{\Sigma}^{-1}, \\ \Delta S_{\ell,4} &= (\tilde{U}^*U_\perp)U_\perp^*T_\ell V_\perp(V_\perp^*\tilde{V})\tilde{\Sigma}^{-1}, \end{aligned}$$

and if we take into account the results of Lemma 4.5 and Lemma 4.6, then

$$\begin{aligned} \tilde{S}_\ell &= (I + \Delta U^*)S_\ell\Sigma(I + \Delta V)\tilde{\Sigma}^{-1} + \Delta S_{\ell,1} + \Delta S_{\ell,2} + \Delta S_{\ell,3} + \Delta S_{\ell,4} \\ &= S_\ell + S_\ell(\Sigma\tilde{\Sigma}^{-1} - I) + \Delta U^*S_\ell\Sigma\tilde{\Sigma}^{-1} + S_\ell\Sigma\Delta V\tilde{\Sigma}^{-1} + \Delta U^*S_\ell\Sigma\Delta V\tilde{\Sigma}^{-1} + \\ &\quad + \Delta S_{\ell,1} + \Delta S_{\ell,2} + \Delta S_{\ell,3} + \Delta S_{\ell,4}. \end{aligned}$$

Further, if we define

$$\begin{aligned} \Delta S_{\ell,5} &= \Delta U^*S_\ell\Sigma\tilde{\Sigma}^{-1} + S_\ell\Sigma\Delta V\tilde{\Sigma}^{-1} + \Delta U^*S_\ell\Sigma\Delta V\tilde{\Sigma}^{-1}, \\ \Delta S_{\ell,6} &= S_\ell(\Sigma\tilde{\Sigma}^{-1} - I), \\ \Delta S_\ell &= \Delta S_{\ell,1} + \Delta S_{\ell,2} + \Delta S_{\ell,3} + \Delta S_{\ell,4} + \Delta S_{\ell,5} + \Delta S_{\ell,6}, \end{aligned}$$

it only remains to find norm bounds on $\Delta S_{\ell,k}$ for $k = 2,\dots,6$. By (4.4) and Lemmas 4.5, 4.6, and 4.7 it follows

$$\begin{aligned}
\|\Delta S_{\ell,2}\|_2 &\le \|\tilde{U}^* U_\perp\|_F \|T_\ell\|_F \|I + \Delta V\|_2 \|\tilde{\Sigma}^{-1}\|_2 \\
&\le \frac{2\sqrt{2}\varepsilon \|T\|_F \|T_\ell\|_F}{\delta_{T,\tilde{T}}\tilde{\sigma}_m}\left(1 + \frac{g\sqrt{2m}\varepsilon\|T\|_F}{\delta_{\min}} + \mathcal{O}(\varepsilon^2)\right) \\
&\le \frac{2\sqrt{2}\varepsilon \|T\|_F \|T_\ell\|_F}{\delta_{min}\tilde{\sigma}_m} + \mathcal{O}(\varepsilon^2), \\
\|\Delta S_{\ell,3}\|_F &\le \|I + \Delta U\|_2 \|T_\ell\|_F \|V_\perp^* \tilde{V}\|_F \|\tilde{\Sigma}^{-1}\|_2 \le \frac{2\sqrt{2}\varepsilon \|T\|_F \|T_\ell\|_F}{\delta_{min}\tilde{\sigma}_m} + \mathcal{O}(\varepsilon^2), \\
\|\Delta S_{\ell,4}\|_F &\le \|\tilde{U}^* U_\perp\|_F \|T_\ell\|_F \|V_\perp^* \tilde{V}\|_F \|\tilde{\Sigma}^{-1}\|_2 \le \mathcal{O}(\varepsilon^2), \\
\|\Delta S_{\ell,5}\|_F &\le \|\Delta U\|_F \|S_\ell \Sigma\|_F \|\tilde{\Sigma}^{-1}\|_2 + \|S_\ell \Sigma\|_F \|\Delta V\|_F \|\tilde{\Sigma}^{-1}\|_2 \\
&\qquad + \|\Delta U^*\|_F \|S_\ell \Sigma\|_F \|\Delta V\|_F \|\tilde{\Sigma}^{-1}\|_2 \\
&\le \frac{(2+g)\sqrt{2m}\varepsilon \|T\|_F \|T_\ell\|_F}{\delta_{\min}\tilde{\sigma}_m} + \mathcal{O}(\varepsilon^2), \\
\|\Delta S_{\ell,6}\|_F &\le \|S_\ell\|_F \left\| \operatorname{diag}\left(\frac{\sigma_i - \tilde{\sigma}_i}{\tilde{\sigma}_i}\right)\right\|_F \le \frac{\varepsilon\|T\|_F}{\tilde{\sigma}_m}\|S_\ell\|_F \le \frac{\varepsilon \|T\|_F \|T_\ell\|_F}{\sigma_m \tilde{\sigma}_m}.
\end{aligned}$$

Finally, by taking results of Lemma 4.3 into account, we can conclude that

$$\begin{aligned}
\tilde{S}_\ell &= S_\ell + \Delta S_\ell, \\
\|\Delta S_\ell\|_F &\le \frac{\varepsilon \|T_\ell\|_F}{\tilde{\sigma}_m}\left(1 + \frac{4\sqrt{2}\|T\|_F}{\delta_{min}} + \frac{(2+g)\sqrt{2m}\|T\|_F}{\delta_{\min}} + \frac{\|T\|_F}{\sigma_m}\right) + \mathcal{O}(\varepsilon^2) \\
&\le \frac{\varepsilon \|T_\ell\|_F}{\tilde{\sigma}_m}\left(1 + \frac{(1 + 4\sqrt{2} + (2+g)\sqrt{2m})\|T\|_F}{\delta_{\min}}\right) + \mathcal{O}(\varepsilon^2). \qquad \square
\end{aligned}$$

Now we switch to C_μ, and we will operate with

$$\tilde{C}_\mu = \sum_{\ell=1}^{d} \mu_\ell \tilde{S}_\ell = \sum_{\ell=1}^{d} \mu_\ell S_\ell + \sum_{\ell=1}^{d} \mu_\ell \Delta S_\ell = C_\mu + \Delta C_\mu, \quad \|\mu\|_2 = 1 \tag{4.27}$$

instead, where by Theorem 4.8

$$\begin{aligned}
\|\Delta C_\mu\|_F &\le \max_{\ell=1,\dots,d} \|\Delta S_\ell\|_F \sum_{\ell=1}^{d} \mu_\ell \\
&\le \frac{\sqrt{d}\varepsilon \max_{\ell=1,\dots,d}\|T_\ell\|_F}{\tilde{\sigma}_m}\left(1 + \frac{(1 + 4\sqrt{2} + (2+g)\sqrt{2m})\|T\|_F}{\delta_{\min}}\right) + \mathcal{O}(\varepsilon^2). \qquad (4.28)
\end{aligned}$$

We will turn now to the perturbation theory for the eigenvalue problem, and we will investigate the relation between the spectral decompositions of C_μ and $\tilde{C}_\mu$. Let us define spectral decompositions

$$C_\mu = W \Lambda_\mu W^{-1}, \qquad \Lambda_\mu = \operatorname{diag}(\lambda_1,\dots,\lambda_m), \tag{4.29}$$

$$\tilde{C}_\mu = \tilde{W} \tilde{\Lambda}_\mu \tilde{W}^{-1}, \qquad \tilde{\Lambda}_\mu = \operatorname{diag}(\tilde{\lambda}_1,\dots,\tilde{\lambda}_m). \tag{4.30}$$

If we assume that all eigenvalues of C_μ and $\tilde{C}_\mu$ are distinct with no particular order, then columns of $W = [\ w_1, \ldots, w_m\ ]$ and $\tilde{W} = [\ \tilde{w}_1, \ldots, \tilde{w}_m\ ]$ are corresponding eigenvectors which are usually normalized to have norm 1. Further, we choose again to scale the vectors $w_1, \ldots, w_m$ by $e^{\iota\nu_i}$ so that $\tilde{w}_i^* w_i = \cos\phi_i$ and not only $|\tilde{w}_i^* w_i| = \cos\phi_i$, where ϕ_i is the angle between eigenvectors w_i and $\tilde{w}_i$ belonging to the eigenvalues λ_i and $\tilde{\lambda}_i$.

THEOREM 4.9. *If we define by $S_\ell = W\Lambda_\ell W^{-1}$ the spectral decomposition of S_ℓ, then*

$$\tilde{W}^{-1}\tilde{S}_\ell\tilde{W} = \Lambda_\ell + \Delta\Lambda_\ell,$$

where $\tilde{W}$ is connected to W via (4.29), (4.30), (4.27), *and* (4.28), *and*

$$\begin{aligned} \|\Delta\Lambda_\ell\|_F \le \varepsilon &\left(\frac{2\sqrt{md}\|\Lambda_\ell\|_2}{\gamma\tilde{\sigma}_m\sigma_{min}(W)} + \frac{1}{\tilde{\sigma}_m}\right) \\ &\times \left(1 + \frac{(1+4\sqrt{2}+(2+g)\sqrt{2m})\|T\|_F}{\delta_{\min}}\right)\max_{\ell=1,\ldots,d}\|T_\ell\|_F\kappa_2(W) + \mathcal{O}(\varepsilon^2), \end{aligned} \tag{4.31}$$

with $\sigma_{min}(W)$ being the smallest singular value of W, and

$$\gamma = \min_{\substack{i,j=1,\ldots,m\\ j\neq i}} |\lambda_j - \tilde{\lambda}_i|. \tag{4.32}$$

Since we are extracting the diagonal of $\tilde{W}^{-1}\tilde{S}_\ell\tilde{W}$, we can bound it from the exact eigenvalues of S_ℓ by

$$\|diag(\tilde{W}^{-1}\tilde{S}_\ell\tilde{W} - \Lambda_\ell)\|_2 \le \|\Delta\Lambda_\ell\|_F, \tag{4.33}$$

where $\|\Delta\Lambda_\ell\|_F$ is bounded by (4.31), *and $\Lambda_\ell = diag(z_1(\ell), \ldots, z_m(\ell))$ with $z_j(\ell)$ denoting ℓth component of z_j.*

Proof. By the assumption on $\tilde{w}_i^* w_i$ we have

$$\|\tilde{w}_i - w_i\|_2 = \sqrt{\|\tilde{w}_i\|_2^2 + \|w_i\|_2^2 - 2\cos\phi_i\|\tilde{w}_i\|_2\|w_i\|_2} = \sqrt{2}\sqrt{1-\cos\phi_i}.$$

By [10, Theorem 5.1] we can conclude that

$$|\sin\phi_i| \le \frac{\kappa_2(W^{-1}(j\neq i,:))\kappa_2(\tilde{w}_i)\|\Delta C_\mu\|_F}{\min_{\substack{j=1,\ldots,m\\ j\neq i}}|\lambda_j - \tilde{\lambda}_i|}.$$

By interlacing property for singular values [7, Corollary 3.1.3] it follows that

$$\kappa_2(W^{-1}(j\neq i,:)) \le \kappa_2(W^{-1}) = \kappa_2(W),$$

and $\kappa_2(\tilde{w}_i) = 1$, so finally

$$|\sin\phi_i| \le \frac{\kappa_2(W)\|\Delta C_\mu\|_F}{\min_{\substack{j=1,\ldots,m\\ j\neq i}}|\lambda_j - \tilde{\lambda}_i|}. \tag{4.34}$$

Now we can finish the bound on $\tilde{w}_i - w_i$

$$\|\tilde{w}_i - w_i\|_2 = \sqrt{2}\sqrt{1-\sqrt{1-(\sin\phi_i)^2}} = \sqrt{\frac{(\sin\phi_i)^2}{1+\sqrt{1-(\sin\phi_i)^2}}} \le |\sin\phi_i|. \tag{4.35}$$

Thus

$$\tilde{W} = W + \Delta W, \tag{4.36}$$

where

$$\begin{aligned}
\|\Delta W\|_F &= \sqrt{\sum_{i=1}^{m} \|\tilde{w}_i - w_i\|_2^2} \\
&\leq \frac{\sqrt{m}\kappa_2(W)}{\min_{\substack{i,j=1,\ldots,m\\ j\neq i}} |\lambda_j - \tilde{\lambda}_i|} \cdot \frac{\sqrt{d}\varepsilon \max_{\ell=1,\ldots,d} \|T_\ell\|_F}{\tilde{\sigma}_m} \\
&\quad \times \left(1 + \frac{(1 + 4\sqrt{2} + (2+g)\sqrt{2m})\|T\|_F}{\delta_{\min}}\right) + \mathcal{O}(\varepsilon^2) \\
&= \frac{\sqrt{md}\varepsilon \max_{\ell=1,\ldots,d} \|T_\ell\|_F \kappa_2(W)}{\gamma\tilde{\sigma}_m} \\
&\quad \times \left(1 + \frac{(1 + 4\sqrt{2} + (2+g)\sqrt{2m})\|T\|_F}{\delta_{\min}}\right) + \mathcal{O}(\varepsilon^2).
\end{aligned} \tag{4.37}$$

We also need a bound on difference $\tilde{W}^{-1} - W^{-1}$. Since,

$$\tilde{W} = W(I + W^{-1}\Delta W),$$

in case when

$$\|W^{-1}\Delta W\|_F \leq \frac{\|\Delta W\|_F}{\sigma_{min}(W)} < 1,$$

where $\sigma_{min}(W)$ is the smallest singular value of W, we have

$$\tilde{W}^{-1} = \left(I - W^{-1}\Delta W + (W^{-1}\Delta W)^2 - (W^{-1}\Delta W)^3 + \cdots\right) W^{-1}. \tag{4.38}$$

We further compute

$$\begin{aligned}
\tilde{W}^{-1}\tilde{S}_\ell\tilde{W} &= \left(I - W^{-1}\Delta W + (W^{-1}\Delta W)^2 - \cdots\right) W^{-1}(S_\ell + \Delta S_\ell)W(I + W^{-1}\Delta W) \\
&= \left(I - W^{-1}\Delta W + (W^{-1}\Delta W)^2 - \cdots\right) \Lambda_\ell(I + W^{-1}\Delta W) + W^{-1}\Delta S_\ell W \\
&\quad + \mathcal{O}(\varepsilon^2) \\
&= \Lambda_\ell - W^{-1}\Delta W\Lambda_\ell + \Lambda_\ell W^{-1}\Delta W + W^{-1}\Delta S_\ell W + \mathcal{O}(\varepsilon^2) \\
&= \Lambda_\ell + \Delta\Lambda_\ell,
\end{aligned}$$

where

$$\|\Delta\Lambda_\ell\|_F \leq 2\frac{\|\Delta W\|_F}{\sigma_{min}(W)}\|\Lambda_\ell\|_2 + \kappa_2(W)\|\Delta S_\ell\|_F + \mathcal{O}(\varepsilon^2),$$

and bound in (4.31) follows from (4.37) and Theorem 4.8. ◻

Furthermore, if we proceed with Algorithm 1.1 we obtain

$$\left(\tilde{W}^{-1}\tilde{S}_\ell\tilde{W}\right)(j,j) = \tilde{z}_j(\ell), \quad \tilde{z}_j = z_j + \Delta z_j, \tag{4.39}$$

and each component of Δz_j is bounded by (4.31). Further, $\tilde{t}_j$ are computed from $\tilde{z}_j$ as

$$\tilde{t}_j(i) = \mathrm{Re}\left(\frac{\log \tilde{z}_j(i)}{-2\pi\iota}\right) = \frac{\mathrm{Im}(\log \tilde{z}_j(i))}{2\pi} = \frac{1}{2\pi}\mathrm{arctg}\frac{\mathrm{Im}(\tilde{z}_j(i))}{\mathrm{Re}(\tilde{z}_j(i))}. \tag{4.40}$$

COROLLARY 4.10. *For $\tilde{t}_j(i)$ computed as in* (4.40) *and for t_j connected with z_j as in* (1.4)*, the following expression holds*

$$\tilde{t}_j(i) = t_j(i) + \Delta t_j(i), \quad |\Delta t_j(i)| \le \frac{|\Delta z_j(i)|}{\pi\eta} + \mathcal{O}(\varepsilon^2), \tag{4.41}$$

where $|\Delta z_j(i)|$ is bounded by (4.31) *from Theorem* 4.9*, and $\eta = \min_{\substack{j=1,\dots,m\\ i=1,\dots,d}} |Re(z_j(i)) Im(z_j(i))|$.*

Proof. If we analyze expression (4.40) we obtain

$$\operatorname{arctg}\frac{\operatorname{Im}(\tilde{z}_j(i))}{\operatorname{Re}(\tilde{z}_j(i))} = \operatorname{arctg}\frac{\operatorname{Im}(z_j(i)) + \operatorname{Im}(\Delta z_j(i))}{\operatorname{Re}(z_j(i)) + \operatorname{Re}(\Delta z_j(i))} = \operatorname{arctg}\frac{\operatorname{Im}(z_j(i))}{\operatorname{Re}(z_j(i))}\left(\frac{1 + \frac{\operatorname{Im}(\Delta z_j(i))}{\operatorname{Im}(z_j(i))}}{1 + \frac{\operatorname{Re}(\Delta z_j(i))}{\operatorname{Re}(z_j(i))}}\right).$$

If we denote by

$$q_j(i) = \frac{\operatorname{Im}(z_j(i))}{\operatorname{Re}(z_j(i))}, \quad \tilde{q}_j(i) = \frac{\operatorname{Im}(\tilde{z}_j(i))}{\operatorname{Re}(\tilde{z}_j(i))}, \quad \delta p_j(i) = \frac{\operatorname{Im}(\Delta z_j(i))}{\operatorname{Im}(z_j(i))}, \quad \delta r_j(i) = \frac{\operatorname{Re}(\Delta z_j(i))}{\operatorname{Re}(z_j(i))},$$

then we have

$$\operatorname{arctg}\,\tilde{q}_j(i) = \operatorname{arctg}\,\left(q_j(i)\frac{1 + \delta p_j(i)}{1 + \delta r_j(i)}\right),$$

with

$$\begin{aligned} \frac{1 + \delta p_j(i)}{1 + \delta r_j(i)} &= 1 + \frac{\delta p_j(i) - \delta r_j(i)}{1 + \delta r_j(i)} = 1 + \delta q_j(i), \\ |\delta q_j(i)| &\le \frac{2\sqrt{\delta r_j(i)^2 + \delta p_j(i)^2}}{1 - \sqrt{\delta r_j(i)^2 + \delta p_j(i)^2}}, \quad \text{where} \end{aligned}$$

$$\sqrt{\delta r_j(i)^2 + \delta p_j(i)^2} = \sqrt{\frac{\operatorname{Re}(\Delta z_j(i))^2 \operatorname{Im}(z_j(i))^2 + \operatorname{Im}(\Delta z_j(i))^2 \operatorname{Re}(z_j(i))^2}{\operatorname{Re}(z_j(i))^2 \operatorname{Im}(z_j(i))^2}} \le \frac{1}{\eta}|\Delta z_j(i)|,$$

taking into account $|z_j(i)| = 1$. Thus

$$|\delta q_j(i)| \le \frac{2|\Delta z_j(i)|}{\eta - |\Delta z_j(i)|} = \frac{2|\Delta z_j(i)|}{\eta} + \mathcal{O}(\varepsilon^2). \tag{4.42}$$

If we assume that for $|q_j(i)| < 1$ it is $|\tilde{q}_j(i)| < 1$, and $|\tilde{q}_j(i)| > 1$ for $|q_j(i)| > 1$, from Taylor expansion of function arctg we can conclude that

$$\operatorname{arctg}\,\tilde{q}_j(i) = \operatorname{arctg}\,q_j(i) + \xi_j(i), \quad |\xi_j(i)| \le |\delta q_j(i)| + \mathcal{O}(\varepsilon^2). \tag{4.43}$$ ∎

The only thing that remains is to analyze solution of the least squares problem

$$\min_{\tilde{c}} \|\tilde{A}^T\tilde{c} - \tilde{f}\|_2, \tag{4.44}$$

where $\tilde{A} = [\tilde{z}_j^k]_{j=1,\dots,m,\ k\in I_n} \in \mathbb{C}^{m\times N}$.

THEOREM 4.11. *For $\tilde{A} = [\tilde{z}_j^k]_{j=1,\dots,m,\ k\in I_n} \in \mathbb{C}^{m\times N}$, where $\tilde{z}_j$ are defined by* (4.39)*, the following holds:*

$$\tilde{A} = A + \Delta A, \ \textit{where}$$
$$\|\Delta A\|_F \leq \varepsilon\sqrt{mN}nd\left(\frac{2\sqrt{md}\|\Lambda_\ell\|_2}{\gamma\tilde{\sigma}_m\sigma_{min}(W)} + \frac{1}{\tilde{\sigma}_m}\right)\left(1 + \frac{(1+4\sqrt{2}+(2+g)\sqrt{2m})\|T\|_F}{\delta_{\min}}\right)$$
$$\times \max_{\ell=1,\dots,d}\|T_\ell\|_F\kappa_2(W)\|A\|_F + \mathcal{O}(\varepsilon^2) \tag{4.45, 4.46}$$

with A as in (1.6)*, or simpler we can write*

$$\|\Delta A\|_F = \zeta\|A\|_F,$$
$$\zeta \leq \mathcal{O}\left(\frac{d^{\frac{3}{2}}m^{\frac{3}{2}}nN^{\frac{1}{2}}}{\gamma\delta_{\min}\tilde{\sigma}_m\sigma_{min}(W)}\right)\|T\|_F \max_{\ell=1,\dots,d}\|T_\ell\|_F\kappa_2(W)\varepsilon + \mathcal{O}(\varepsilon^2). \tag{4.47}$$

Proof. We have

$$\begin{aligned}
\tilde{z}_j^k &= \tilde{z}_j(1)^{k(1)}\cdots\tilde{z}_j(d)^{k(d)} = (z_j(1)+\Delta z_j(1))^{k(1)}\cdots(z_j(d)+\Delta z_j(d))^{k(d)}\\
&= \left(z_j(1)^{k(1)} + k(1)z_j(1)^{k(1)-1}\Delta z_j(1) + \mathcal{O}(\varepsilon^2)\right)\cdots\\
&\quad \left(z_j(d)^{k(d)} + k(d)z_j(d)^{k(d)-1}\Delta z_j(1) + \mathcal{O}(\varepsilon^2)\right)\\
&= z_j(1)^{k(1)}\cdots z_j(d)^{k(d)} + z_j(1)^{k(1)}\cdots z_j(d)^{k(d)}\\
&\quad \times\left(\frac{k(1)\Delta z_j(1)}{z_j(1)} + \cdots + \frac{k(d)\Delta z_j(d)}{z_j(d)}\right) + \mathcal{O}(\varepsilon^2)\\
&= z_j^k(1+\Delta z_j^k)
\end{aligned} \tag{4.48}$$

with

$$|\Delta z_j^k| \leq nd\max_{\ell=1,\dots,d}\|\Delta\Lambda_\ell\|_F + \mathcal{O}(\varepsilon^2). \tag{4.49}$$

Hence, the following componentwise bound holds

$$\begin{aligned}\tilde{A} &= A + \Delta A,\\ |\Delta A| &\leq nd\max_{\ell=1,\dots,d}\|\Delta\Lambda_\ell\|_F|A| + \mathcal{O}(\varepsilon^2)\end{aligned} \tag{4.50}$$

and

$$\|\Delta A\|_F = \sqrt{\sum_{j=1}^m\sum_{k\in I_n}|\Delta z_j^k|^2|z_j^k|^2} \leq \sqrt{mN}nd\max_{\ell=1,\dots,d}\|\Delta\Lambda_\ell\|_F\|A\|_F + \mathcal{O}(\varepsilon^2) \tag{4.51}$$

(4.46) follows from Theorem 4.9. ∎

Finally, we employ perturbation theory for linear least squares problems, where for the final result we need the following

$$\begin{aligned}
\tilde{f} &= f + \Delta f, & \|\Delta f\|_2 &\leq \varepsilon\|f\|_2,\\
\tilde{A} &= A + \Delta A, & \|\Delta A\|_2 \leq \|\Delta A\|_F = \zeta\|A\|_F &\leq \sqrt{m}\zeta\|A\|_2,\\
r &= f - A^T c = 0;
\end{aligned}$$

then, by [6, Theorem 20.1] provided that $\kappa_2(A)\sqrt{m}\zeta < 1$, the statement of the following theorem holds.

THEOREM 4.12. *For the solution c of the linear least squares problem $\text{argmin}_c \|A^T c - f\|_2$, and for the solution $\tilde{c}$ of $\text{argmin}_{\tilde{c}} \|\tilde{A}^T \tilde{c} - \tilde{f}\|_2$, under assumption* (4.1) *and assumptions of Theorem* 4.11*, we have*

(4.52)
$$\begin{aligned}\frac{\|\tilde{c}-c\|_2}{\|c\|_2} &\leq \frac{2\kappa_2(A)\sqrt{m}\zeta}{1-\kappa_2(A)\sqrt{m}\zeta} \\ &\leq \mathcal{O}\left(\frac{d^{\frac{3}{2}} m^2 n N^{\frac{1}{2}}}{\gamma \delta_{\min} \tilde{\sigma}_m \sigma_{min}(W)}\right) \|T\|_F \max_{\ell=1,\ldots,d} \|T_\ell\|_F \kappa_2(W)\kappa_2(A)\varepsilon + \mathcal{O}(\varepsilon^2).\end{aligned}$$

As we can see from the derived analysis, the forward errors of computed parameters $\tilde{t}_j$ and coefficients $\tilde{c}_j$ are proportional to the relative error ε of the input data and depend on condition numbers of the eigenvalue matrix W and the least squares problem matrix A, as expected from the perturbation theory. The gap function on singular values of T and $\tilde{T}$ also plays an important role in the error bound. In the case when the algorithm is executed in finite precision arithmetic there would be additional terms in error bounds proportional to the unit roundoff.

5. Numerical tests.

The aim of this section is to prove efficiency of the parallel variant of Prony's method as well as to illustrate results of its numerical analysis.

In the efficiency tests the sampling values were generated from a predetermined exponential sum with no noise, where m was chosen from the set $\{5, 10, 15, 20\}$, d was set to 2 and 3. The sample size was fixed for $n = 20$, which produces matrix dimension of $N = 441$ for $d = 2$, and $N = 9261$ for $d = 3$. The parameters and coefficients are defined as follows:

$$t_j(i) = ((i-1)\cdot m + j - 1)\cdot 10^{-\lceil \log_{10}(d\cdot m)\rceil}, \quad c_j = j + \iota j, \quad i = 1,\ldots,d,\ j = 1,\ldots,m.$$

The criteria for the rank determination was the first i that satisfies $\sigma_i < N\varepsilon_M \sigma_1$, where ε_M is the machine precision. This tolerance is chosen in order to incorporate possible rounding error of the SVD algorithm. The following computational environment was used:

- 2x Intel(R) Xeon(R) E5-2690 v3 @ 2.60GHz (24 cores in total);
- 256 GB RAM, each processor is equipped with 30 MB of cache memory;
- Nvidia Tesla K40c (Kepler generation, 12 GB of GDDR5);
- Intel Parallel Studio XE 2016 + MKL 11.3;
- Nvidia CUDA 8.0.
- The CPUs reach the peak DGEMM performance of about 800 Gflops, while the GPU has the peak performance of about 1200 Gflops.

First, we are going to compare the SVD algorithms. SVD computation of the matrix T is performed on CPU for all algorithm variants; hence we measured CPU execution time for three SVD algorithms:

1. computation of full SVD implemented by LAPACK function `zgesvd()`,
2. Lanczos bidiagonalization method,
3. block power method for SVD, with starting column-dimensions of matrices U_0 and V_0 equal to $2m$.

The execution times are presented in Figure 2. It is interesting to see that all three algorithms determined the same rank, and in cases when $d = 2$ and $m = 15, 20$ it seems that condition on n from [3, Theorem 2.1] was not satisfied and the sample was too small. The determined rank of T was smaller than the number of terms in the exponential sum of sample. In these cases the accuracy of Prony's solution was

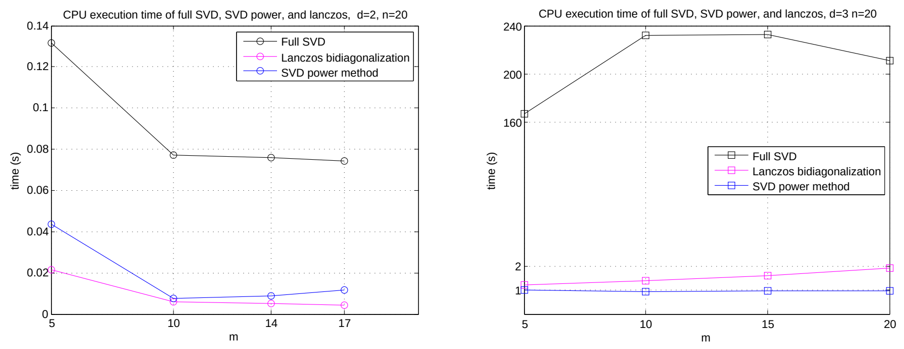

FIG. 2. *CPU execution time of three SVD algorithms, for $d = 2$ and $d = 3$.*

reduced. Further, we can also conclude that for smaller matrix dimensions Lanczos bidiagonalization method is the fastest method, but for larger dimensions the SVD power method is the most efficient as expected, specially for larger ranks. Computation of full SVD is by far the slowest method, even 200 times slower than the power method for $N = 9261$. Thus, we can assert that the right choice of the SVD algorithm is crucial for the efficiency of the Prony's method.

Next, we are going to compare four variants of the Prony's method:

1. sequential variant with full SVD,
2. sequential variant with block power method for SVD,
3. parallel variant described in Figure 1 with full SVD,
4. parallel variant described in Figure 1 with block power method for SVD.

Sequential algorithms mentioned above execute their subtasks in sequential order, but not all operations are fully sequential. Matrix operations are implicitly parallelized by calls to multithreaded BLAS routines on our machine. The execution times are presented in Figure 3, and speed-up factors of parallel over sequential versions can be found in Figure 4. The first obvious conclusion is that there isn't much benefit from the parallel Prony's method with full SVD; its execution time compared to the execution time of sequential Prony's algorithm was not reduced much (the largest speed-up factor is about 1.6). The reason for this is that the SVD algorithm occupies the largest fraction of the total execution time, for both sequential and parallel algorithm. On the other hand, replacing the full SVD with the SVD power method in the sequential algorithm produced speed-up factors up to 5.8 for $d = 3$, and the parallel variant with SVD power method is in some cases over 120 times faster than the original method. This parallel algorithm is much more efficient even than its sequential version, and for $d = 3$ it is about 22 times faster. In the sequential Prony's algorithm with SVD power method, the most demanding task is generation of matrices T and T_ℓ, and the SVD algorithm is very fast. In the parallel version generation of matrices elements is performed in parallel, and thus the larger speed-up factor is obtained.

In the next test round we generate the sampling values from a predetermined exponential sum again but we also introduced noise. The data were taken for $d = 3$, $n = 20$, and $m = 5$, as

$$f(k)(1 + \delta_k),$$

where the relative perturbation δ_k is bounded by some tolerance ε. The criteria for the rank determination in this case was the first i that satisfies $\sigma_i < \text{tol} \cdot \sigma_1$, where

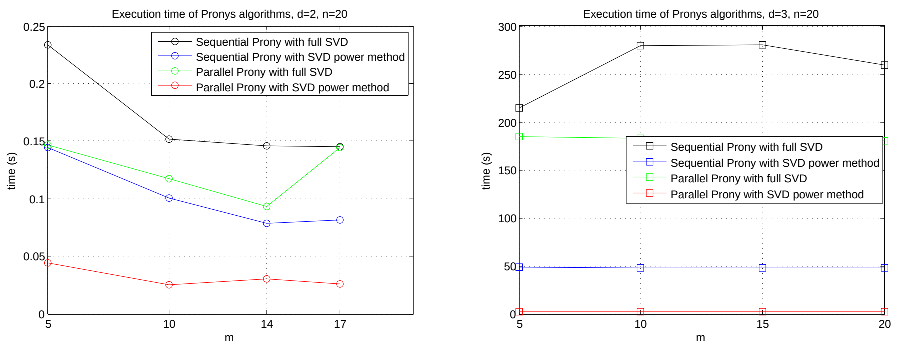

FIG. 3. *Execution time of four variants of Prony's method, for $d = 2$ and $d = 3$.*

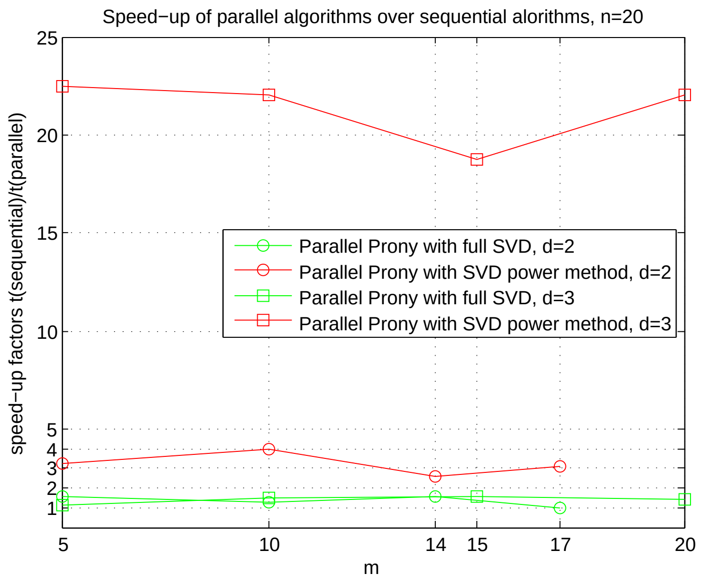

FIG. 4. *Speed-up factors of parallel over sequential versions of Prony's method, for $d = 2$ and $d = 3$.*

TABLE 2
Accuracy results for Prony's method in case when $d = 3$, $n = 20$, and $m = 5$.

ε	tol	$\|\tilde{A}^T\tilde{c} - \tilde{f}\|_2/\|\tilde{f}\|_2$	$\max_{\substack{j=1,\dots,d \\ i=1,\dots,d}} \|\tilde{t}_j(i) - t_j(i)\|$	$\|\tilde{c} - c\|_2/\|c\|_2$
0	$N\varepsilon_M$	1.40484e-14	4.38538e-15	7.67293e-13
1e-9	1e-9	3.00100e-10	1.13784e-11	9.50551e-10
1e-6	1e-6	3.00100e-07	1.13789e-08	9.50556e-07
1e-3	1e-4	2.99893e-04	1.13424e-05	9.52641e-04

in the most cases tol $= \varepsilon$. We measured relative residual norm for the least squares problem in line 7 of Algorithm 1.1, maximal error in the components of vectors t_j for $j = 1, \dots, m$, and the relative error for the coefficient vector c. The results are displayed in Table 2.

As we can see, errors in parameters t_j and coefficients c_j are proportional to the error in the sample values, as numerical analysis in section 4 suggested. Only for

$\varepsilon = 10^{-3}$ the tolerance for rank determination has to be smaller, because otherwise the algorithm detected smaller numerical rank $m = 4$ and the errors were large. The relative residual norm presented in Table 2 can be used as operational accuracy estimate within the algorithm, implying whether the rank is well determined or not.

6. Conclusion.

In this paper we proposed a parallel version of Prony's method with multivariate matrix pencil approach, which with appropriate choice of the SVD algorithm for the matrix T (1.2) turned out to be up to 120 times faster than the original algorithm on our computing platform. Since we expect matrix T to have low rank, the most convenient SVD algorithms are the Lanczos bidiagonalization method and the block power method for SVD. The subtasks of Prony's method are distributed over CPU threads and GPU block of threads, where the CPU is dedicated to more complex tasks such as computing SVD and the GPU to more massive tasks. We also provided a detailed numerical analysis of the method's performance in case of noisy input data, showing that the forward error is proportional to the error in input data, and this was confirmed by numerical tests.

REFERENCES

[1] F. Andersson and M. Carlsson, *ESPRIT for multidimensional general grids*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 1470–1488.

[2] A. H. Bentbib and A. Kanber, *Block power method for SVD decomposition*, An. St. Univ. Ovidius Constanta, 23 (2015), pp. 45–58.

[3] M. Ehler, S. Kunis, T. Peter, and C. Richter, *A randomized multivariate matrix pencil method for superresolution microscopy*, Electron. Trans. Numer. Anal., 51 (2019), pp. 63–74.

[4] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, J. SIAM Numer. Anal., 2 (1965), pp. 205–224.

[5] G. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, 1996.

[6] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Second Edition, SIAM, Philadelphia, 2002.

[7] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.

[8] Y. Hua and T. K. Sarkar, *Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise*, IEEE Trans. Acoust. Speech Signal Process., 38 (1990), pp. 814–824.

[9] I. C. F. Ipsen, *Relative perturbation results for matrix eigenvalues and singular values*, Acta Numerica, 7 (1998), pp. 151–201.

[10] I. C. F. Ipsen, *Absolute and relative perturbation bounds for invariant subspaces of matrices*, Linear Algebra Appl., 308 (2000), pp. 45–56.

[11] S. Kunis, T. Peter, T. Römer, and U. von der Ohe, *A multivariate generalization of Prony's method*, Lin. Alg. Appl., 490 (2016), pp. 31–47.

[12] R. M. Larsen, *Lanczos Bidiagonalization with Partial Reorthogonalization*, DAIMI Report Series 27(537), Computer Science Department, Aarhus University, 1998.

[13] R.-C. Li, *Relative perturbation theory: eigenspace and singular subspace variations*, LAPACK working note #85, 1996.

[14] Netlib: BLAS (Basic Linear Algebra Subprograms), http://www.netlib.org/blas (2017).

[15] NVIDIA: CUBLAS Library DU-06702-001_v10.2, User Guide, http://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf (2019).

[16] NVIDIA: CUDA Toolkit Documentation v10.2.89, https://docs.nvidia.com/cuda/index.html (2019).

[17] C. C. Paige, *Bidiagonalization of matrices and solution of linear equations*, SAIM J. Numer. Anal., 11 (1974), pp. 197–209.

[18] D. Potts and M. Tasche, *Parameter estimation for exponential sums by approximate Prony method*, Sig. Proc., 90 (2010), pp. 1631–1642.

[19] D. Potts and M. Tasche, *Parameter estimation for nonincreasing exponential sums by Prony-like methods*, Linear Algebra Appl., 439 (2013), pp. 1024–1039.

[20] D. Potts and M. Tasche, *Fast ESPRIT algorithms based on partial singular value decompositions*, Appl. Numer. Math., 88 (2015), pp. 31–45.

[21] R. Roy and T. Kailath, *ESPRIT - estimation of signal parameters via rotational invariance techniques*, Part II, Signal Processing, 23 (1990), pp. 369–411.

[22] S. Sahnoun, K. Usevich, and P. Comon, *Multidimensional ESPRIT for damped and undamped signals: algorithm, computations, and perturbation analysis*, IEEE Trans. Signal Process., 65 (2017), pp. 5897–5910.

[23] R. Schmidt, *Multiple emitter location and signal parameter estimation*, IEEE Trans. Antennas Propag, 34 (1986), pp. 276–280.

[24] P. Å. Wedin, *Perturbation bounds in connection with singular value decomposition*, BIT, 12 (1972), pp. 99–111.

[25] G. Xu, R. H. Roy, and T. Kailath, *Fast ESPRIT – application of fast signal-subspace decomposition in array signal processing*, in 1990 Conference Record Twenty-Fourth Asilomar Conference on Signals, Systems and Computers, 2 (1990), pp. 961–965.

[26] G. Xu and T. Kailath, *Fast subspace decomposition*, IEEE Trans. Signal Process., 42 (1994), pp. 539–551.