SIAM J. MATRIX ANAL. APPL.
Vol. 42, No. 2, pp. 1045–1072

EFFICIENT ALGORITHMS FOR EIGENSYSTEM REALIZATION USING RANDOMIZED SVD*

RACHEL MINSTER†, ARVIND K. SAIBABA†, JISHNUDEEP KAR‡, AND ARANYA CHAKRABORTTY‡

Abstract. The eigensystem realization algorithm (ERA) is a data-driven approach for subspace system identification and is widely used in many areas of engineering. However, the computational cost of the ERA is dominated by a step that involves the singular value decomposition (SVD) of a large, dense matrix with block Hankel structure. This paper develops computationally efficient algorithms for reducing the computational cost of the SVD step by using randomized subspace iteration and exploiting the block Hankel structure of the matrix. We provide a detailed analysis of the error in the identified system matrices and the computational cost of the proposed algorithms. We demonstrate the accuracy and computational benefits of our algorithms on two test problems: the first involves a partial differential equation that models the cooling of steel rails, and the second is an application from power systems engineering.

Key words. eigensystem realization algorithm, system identification, singular value decomposition, randomized algorithms

AMS subject classifications. 93B30, 93B15, 65F15, 37M10, 15A18

DOI. 10.1137/20M1327616

1. Introduction.

Linear time invariant (LTI) systems form one of the most important classes of dynamic systems existing in the physical world. LTI systems closely approximate the linear behavior of any nonlinear physical process around a desired operating point, and provide insights on how the system will behave in response to any small-signal change in its equilibrium state. The state-space representation of a LTI discrete-time system can be written as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \qquad \boldsymbol{x}_0 = \boldsymbol{x}(0), \\ \boldsymbol{y}_k &= \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k, \end{aligned} \tag{1.1}$$

where n is the order of the system, $k = 0, 1, 2, \ldots$ is the sampling index, $\boldsymbol{x}_k \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}_k \in \mathbb{R}^m$ is the input vector, and $\boldsymbol{y}_k \in \mathbb{R}^\ell$ is the output vector at sampling instant k with m and ℓ being the number of inputs and outputs, respectively. The matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is referred to as the state matrix, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ is the input matrix, $\boldsymbol{C} \in \mathbb{R}^{\ell\times n}$ is the output matrix, and $\boldsymbol{D} \in \mathbb{R}^{\ell\times m}$ is the input-output feedthrough matrix. In the majority of practical applications, however, these four matrices may not be exactly known to the system modeler because of different kinds of model and operational uncertainties. In that case, one must estimate these matrices from sampled measurements of the input sequence $\{\boldsymbol{u}_k\}$ and the output sequence

*Received by the editors March 26, 2020; accepted for publication (in revised form) by P. Drineas March 18, 2021; published electronically June 29, 2021.
https://doi.org/10.1137/20M1327616
Funding: The work of the first author was partially supported by the National Science Foundation grant DMS-1745654. The work of the second author was partially supported by the National Science Foundation grant DMS-1821149. The work of the third and fourth authors was partially supported by the National Science Foundation grant ECS-1711004.
†Department of Mathematics, North Carolina State University, Raleigh, NC 27695 USA (rlminste@ncsu.edu, asaibab@ncsu.edu).
‡Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695 USA (jkar@ncsu.edu, achakra2@ncsu.edu).

$\{\boldsymbol{y}_k\}$. This process is referred to as system identification. Since the state may be represented in different coordinate frames without changing input-output characteristics, the more formal definition of system identification is given as the process of estimating $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ up to a similarity transformation $(\boldsymbol{TAT}^{-1}, \boldsymbol{TB}, \boldsymbol{CT}^{-1}, \boldsymbol{D})$, where $\boldsymbol{T} \in \mathbb{R}^{n \times n}$ is an invertible matrix. As mentioned, such a transformation does not change the input-output behavior or the transfer function of the system. We assume that the system to be identified is observable and reachable [14]. There exists a vast literature on system identification with a variety of numerical algorithms, developed for various applications such as electric power systems, process control, mechanical systems, aerospace applications, civil and architectural applications, and chemical and biological processes to name a few. For a survey of these methods, please see [1, 14, 22, 23]. In this paper, we focus on identifying the system matrices when the inputs $\{\boldsymbol{u}_k\}$ are in the form of impulse functions.

One of the most common identification methods used for identifying LTI models in practice is known as the eigensystem realization algorithm (ERA), initially proposed in [13], but can be found in many other sources such as [23]. ERA is a subset of a broader class of identification methods called subspace system identification, and is a purely data-driven approach consisting of two main steps. In the first step, this algorithm computes the singular value decomposition (SVD) of a block Hankel matrix, constructed using the impulse response data. In the second step, the truncated SVD is used to solve a least-squares optimization problem to identify the system matrices. The first step is computationally expensive since it has cubic complexity in the dimensions of the matrix. To reduce the computational cost, the tangential interpolation based ERA (TERA) [12] was proposed. This method reduces the number of inputs and outputs, thus reducing the dimensions of the matrix whose SVD needs to be computed. Another algorithm, known as CUR-ERA [11], uses the CUR decomposition to efficiently compute the low-rank decomposition of the block Hankel matrix. While these algorithms have successfully lowered the costs, important challenges remain.

Contributions and content. We propose new randomized algorithms for tackling the computational costs of ERA. The first algorithm (section 3.1) accelerates the standard randomized SVD by exploiting the block Hankel structure to efficiently perform matrix-vector products. The second algorithm (section 3.2) employs similar ideas as the first algorithm in combination with the TERA. The resulting algorithms are efficient both in terms of storage costs and computational costs. In section 4, we derive new error bounds that provide insight into the accuracy and stability of the system matrices identified using the approximation algorithms. The error analysis is not tied to any particular algorithm and in this sense is fairly general. Finally, we demonstrate the benefits of the proposed algorithms on two different applications: first, a heat transfer problem for cooling of steel rails, and second, a dynamic model of an electric power system with multiple generators and loads.

Comparison to related work. While TERA works with smaller matrices, it still has cubic complexity in the number of time measurements. This is computationally demanding when the dynamics of the system are slow, and many measurements in time are needed to fully resolve the dynamics. The CUR-ERA algorithm has similar storage and computational complexity as the algorithms we propose, but numerical experiments (see section 5) suggest that our algorithm is more accurate. Furthermore, the randomized algorithms developed here can exploit parallelism in multiple ways; the matrix-vector products can be parallelized across multiple random vectors as well as over the input/output pairs. This makes it attractive for implementations on high performance computing platforms. If the target rank for the low-rank decomposition

is not known in advance, one can use randomized range finding algorithms [16, 27] to estimate the target rank. Two other features make our contributions attractive: first, the block Hankel structure that we exploit here can be adapted to any other matrix-free low-rank approximation algorithm, for example, based on the Lanczos bidiagonalization [20], and second, the error analysis developed here is informative for any approximation algorithm. There are other randomized algorithms for system identification [25, 26] but they tackle slightly different settings.

2. Background. In this section, we first review the ERA (section 2.1), the associated computational costs, and motivate the need for efficient algorithms. We also review the key ingredients needed to construct our algorithms: algorithms for storing and computing with Hankel matrices (section 2.2), and randomized SVD (section 2.3) for computing the low-rank factorizations.

2.1. Eigensystem realization algorithm. ERA was first proposed in [13], but we follow the formulation in [12]. Other references for the ERA include [23, 22]. Now, assuming the initial condition $\boldsymbol{x}_0 = \boldsymbol{0} \in \mathbb{R}^n$, and given a sequence of inputs $\{\boldsymbol{u}_i\}$ for $i = 0, 1, \ldots$, the outputs observed are $\boldsymbol{y}_0 = \boldsymbol{h}_0 \boldsymbol{u}_0$ and

$$\boldsymbol{y}_k = \sum_{j=0}^{k-1} \boldsymbol{h}_{k-j-1} \boldsymbol{u}_j, \quad k = 1, 2, \ldots, \tag{2.1}$$

where the matrices $\{\boldsymbol{h}_k\}$ are called the *Markov parameters* and are given by

$$\boldsymbol{h}_k = \begin{cases} \boldsymbol{D}, & k = 0, \\ \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{B}, & k = 1, \ldots, 2s-1. \end{cases} \tag{2.2}$$

The system is excited m times using impulse input excitations. That is, we take

$$\boldsymbol{u}_k^{(j)} = \begin{cases} \boldsymbol{e}_j, & k = 0, \\ \boldsymbol{0}, & k > 0, \end{cases} \qquad j = 1, \ldots, m.$$

Here $\boldsymbol{e}_j$ is the jth column of the $m \times m$ identity matrix. The outputs from each impulse excitation can be used to construct the Markov parameters $\{\boldsymbol{h}_k\}$. The ERA uses the Markov parameters to recover the system matrices. If data using impulse inputs are not available, then one can estimate the Markov parameters from the general input data [10].

These Markov parameters are first arranged to form the block Hankel matrix $\boldsymbol{\mathcal{H}}_s \in \mathbb{R}^{(s\ell)\times(sm)}$ defined as

$$\begin{aligned} \boldsymbol{\mathcal{H}}_s &= \begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \ldots & \boldsymbol{h}_s \\ \boldsymbol{h}_2 & \boldsymbol{h}_3 & \ldots & \boldsymbol{h}_{s+1} \\ \vdots & \vdots & \iddots & \vdots \\ \boldsymbol{h}_s & \boldsymbol{h}_{s+1} & \ldots & \boldsymbol{h}_{2s-1} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{C}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \ldots & \boldsymbol{C}\boldsymbol{A}^{s-1}\boldsymbol{B} \\ \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{2}\boldsymbol{B} & \ldots & \boldsymbol{C}\boldsymbol{A}^{s}\boldsymbol{B} \\ \vdots & \vdots & \iddots & \vdots \\ \boldsymbol{C}\boldsymbol{A}^{s-1}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{s}\boldsymbol{B} & \ldots & \boldsymbol{C}\boldsymbol{A}^{2s-2}\boldsymbol{B} \end{bmatrix}. \end{aligned} \tag{2.3}$$

Here, s is a chosen parameter that determines the number of block rows and columns of $\boldsymbol{\mathcal{H}}_s$. It is known that this matrix can be factorized into the observability matrix $\boldsymbol{\mathcal{O}}_s$ and the controllability matrix $\boldsymbol{\mathcal{C}}_s$, as $\boldsymbol{\mathcal{H}}_s = \boldsymbol{\mathcal{O}}_s \boldsymbol{\mathcal{C}}_s$, where

$$\boldsymbol{\mathcal{O}}_s = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{s-1} \end{bmatrix}, \qquad \boldsymbol{\mathcal{C}}_s = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} & \dots & \boldsymbol{A}^{s-1}\boldsymbol{B} \end{bmatrix}.$$

If the system is assumed to be reachable and observable, then $\mathcal{R}(\boldsymbol{\mathcal{H}}_s) = \mathcal{R}(\boldsymbol{\mathcal{O}}_s)$, where $\mathcal{R}(\cdot)$ denotes the range (or column space) of a matrix. We can obtain a basis for $\boldsymbol{\mathcal{O}}_s$ using the reduced SVD of $\boldsymbol{\mathcal{H}}_s = \boldsymbol{U}_n \boldsymbol{\Sigma}_n \boldsymbol{V}_n^\top$. Then, partition the left singular vectors as

$$\boldsymbol{U}_n = \begin{bmatrix} \boldsymbol{\Upsilon}_f \\ * \end{bmatrix} = \begin{bmatrix} * \\ \boldsymbol{\Upsilon}_l \end{bmatrix},$$

where $\boldsymbol{\Upsilon}_f, \boldsymbol{\Upsilon}_l \in \mathbb{R}^{(s-1)\ell \times n}$ and $*$ denotes blocks that do not affect the remaining computations. We can obtain $\boldsymbol{A}$ using the formula $\boldsymbol{A} = \boldsymbol{\Upsilon}_f^\dagger \boldsymbol{\Upsilon}_l$, where † denotes the Moore–Penrose inverse. Furthermore, we can obtain the output matrix $\boldsymbol{C}$ and the input matrix $\boldsymbol{B}$ using the formulas

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_\ell & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Upsilon}_f, \qquad \boldsymbol{B} = \boldsymbol{\Sigma}_n \boldsymbol{V}_n^\top \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{0} \end{bmatrix}. \tag{2.4}$$

The matrix $\boldsymbol{D}$ can be identified as the Markov matrix $\boldsymbol{h}_0$ and, therefore, we will not discuss its estimation in future sections.

Reduced order model. In some applications, the goal is not only to identify the system but also obtain a reduced order model of the system. That is, we seek the reduced system matrices $\boldsymbol{A}_r \in \mathbb{R}^{r\times r}$, $\boldsymbol{B}_r \in \mathbb{R}^{r\times m}$, $\boldsymbol{C}_r \in \mathbb{R}^{\ell\times r}$, and $\boldsymbol{D}_r \in \mathbb{R}^{\ell\times m}$ which approximate the dynamics of the original system (1.1). To accomplish this, as before, we first compute a rank r approximation to the matrix $\boldsymbol{\mathcal{H}}_s$ as

$$\boldsymbol{\mathcal{H}}_s \approx \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top,$$

where $r \leq n$. We partition the left singular vectors $\boldsymbol{U}_r$ as

$$\boldsymbol{U}_r = \begin{bmatrix} \boldsymbol{\Upsilon}_f^{(r)} \\ * \end{bmatrix} = \begin{bmatrix} * \\ \boldsymbol{\Upsilon}_l^{(r)} \end{bmatrix},$$

such that $\boldsymbol{\Upsilon}_f^{(r)}, \boldsymbol{\Upsilon}_l^{(r)} \in \mathbb{R}^{(s-1)\ell\times r}$. Then, we compute the reduced order model $\boldsymbol{A}_r = [\boldsymbol{\Upsilon}_f^{(r)}]^\dagger \boldsymbol{\Upsilon}_l^{(r)}$ such that it minimizes the least-squares problem

$$\min_{\widehat{\boldsymbol{A}} \in \mathbb{R}^{r\times r}} \|\boldsymbol{\Upsilon}_f^{(r)} \widehat{\boldsymbol{A}} - \boldsymbol{\Upsilon}_l^{(r)}\|.$$

The reduced order output matrix $\boldsymbol{C}_r$ and the input matrix $\boldsymbol{B}_r$ are computed using the formulas

$$\boldsymbol{C}_r = \begin{bmatrix} \boldsymbol{I}_\ell & \boldsymbol{0} \end{bmatrix} \boldsymbol{U}_r, \qquad \boldsymbol{B}_r = \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{0} \end{bmatrix}. \tag{2.5}$$

Other papers, such as [11], use a slightly different representation for the system matrix $\boldsymbol{A}_r$ than the one used in [13]. In this alternate representation, $\boldsymbol{A}'_r$ is obtained using $\boldsymbol{A}'_r = \boldsymbol{\Sigma}_r^{-1/2}\boldsymbol{\Upsilon}_f^{\dagger}\boldsymbol{\Upsilon}_l\boldsymbol{\Sigma}_r^{1/2}$. Then, $\boldsymbol{B}'_r$ and $\boldsymbol{C}'_r$ are obtained using

$$\boldsymbol{C}'_r = \begin{bmatrix}\boldsymbol{I}_\ell & \boldsymbol{0}\end{bmatrix}\boldsymbol{\Upsilon}_f\boldsymbol{\Sigma}_r^{1/2}, \quad \boldsymbol{B}'_r = \boldsymbol{\Sigma}_r^{1/2}\boldsymbol{V}_r^{\top}\begin{bmatrix}\boldsymbol{I}_m \\ \boldsymbol{0}\end{bmatrix}.$$

Note that the two formulations are equivalent up to a similarity transformation, and this results in the same Markov parameters and input-output behavior of the system. If $r = n$, then the ERA determines the system matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ and there is no model reduction step. On the other hand, if the target rank $r < n$, then the reduced order model is guaranteed stability under the conditions of [11, Theorem 3].

Computational cost. We briefly review the range of possible parameters.

1. State space: In the power system applications, the dimension of the state space n is around 10^2, whereas in applications with partial differential equations (e.g., the heat transfer application in section 5) this dimension can be large, i.e., 10^3–10^4.
2. Input dimension: The number of inputs m is in the range 1–10^2.
3. Output dimension: The number of outputs ℓ is also in the range 1–10^2. In certain applications, the number of outputs is also the dimension of the state space.
4. Sample size: When the dynamics of the system are slow, the number of samples $2s - 1$ can be large, i.e., 10^2–10^5.

The dominant cost of ERA is the cost of storing and factorizing the matrix $\boldsymbol{\mathcal{H}}_s$, which is of size $s\ell \times sm$. The cost of storing the matrix $\boldsymbol{\mathcal{H}}_s$ is $\mathcal{O}(s^2\ell m)$ entries, whereas the cost of computing the SVD is $\mathcal{O}(s^3\ell m \min\{\ell, m\})$ floating point operations (flops). In the applications we consider, forming and factoring $\boldsymbol{\mathcal{H}}_s$ is expensive and is infeasible in some of the large-scale examples. The algorithms we propose are both efficient in storage and computational costs and make ERA applicable to larger problem sizes.

2.2. Hankel matrices. Structured matrices such as circulant and Hankel matrices are computationally efficient to work with since matrix-vector products (matvecs) can be accelerated using the fast Fourier transform (FFT). We first review circulant matrices. Circulant matrices are completely determined by their first column

$$\boldsymbol{x} = \begin{bmatrix}x_1 & \dots & x_N\end{bmatrix}^{\top},$$

and are diagonalized by the Fourier matrix. Let the circulant matrix $\boldsymbol{X}$ be defined as

$$\boldsymbol{X} = \begin{bmatrix} x_1 & x_N & \dots & x_2 \\ x_2 & x_1 & \dots & x_3 \\ \vdots & \vdots & \ddots & \vdots \\ x_N & x_{N-1} & \dots & x_1 \end{bmatrix},$$

and let $\boldsymbol{F}_N$ be the $N \times N$ Fourier matrix with entries $(\boldsymbol{F}_N)_{jk} = e^{2\pi i(j-1)(k-1)/N}$, where $i = \sqrt{-1}$ and $j, k = 1, \dots, N$. Then the eigenvalue decomposition of $\boldsymbol{X}$ is $\boldsymbol{X} = \boldsymbol{F}_N^{*}\mathrm{diag}(\mathbf{F}_N\boldsymbol{x})\mathbf{F}_N$, where $\boldsymbol{x} = \boldsymbol{X}(:, 1)$ is the first column of $\boldsymbol{X}$. This means that the circulant matrix has eigenvalues $\mathbf{F}_N\boldsymbol{x}$. This result implies that matvecs $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{v}$ can be performed efficiently using FFTs as $\boldsymbol{y} = \mathrm{IFFT}(\mathrm{FFT}(\boldsymbol{v}) \odot \mathrm{FFT}(\boldsymbol{x}))$, where $\odot$ is the elementwise product and $\mathrm{FFT}(\cdot)$ and $\mathrm{IFFT}(\cdot)$ denote the fast Fourier and inverse

FFTs, respectively. The computational cost involves 2 FFTs and one IFFT, and can be implemented efficiently in $\mathcal{O}(N \log_2 N)$ flops.

Hankel matrices have constant entries along every antidiagonal; that is, given the parameters $h_1, h_2, \ldots, h_{2s-1}$, the Hankel matrix is

$$\boldsymbol{H}_s = \begin{bmatrix} h_1 & h_2 & \ldots & h_s \\ h_2 & h_3 & \ldots & h_{s+1} \\ \vdots & \vdots & \iddots & \vdots \\ h_s & h_{s+1} & \ldots & h_{2s-1} \end{bmatrix}.$$

This corresponds to the single input/single output case. Note that permuting a Hankel matrix with the reverse identity permutation matrix $\boldsymbol{J}_s$ results in a Toeplitz matrix (constant entries along every diagonal). That is,

$$\boldsymbol{J}_s = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 1 & 0 \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ 0 & 1 & \ldots & 0 & 0 \\ 1 & 0 & \ldots & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{H}_s\boldsymbol{J}_s = \begin{bmatrix} h_s & h_{s-1} & \ldots & h_2 & h_1 \\ h_{s+1} & h_s & \ldots & h_3 & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{2s-2} & \vdots & \ldots & h_s & h_{s-1} \\ h_{2s-1} & h_{2s-2} & \ldots & h_{s+1} & h_s \end{bmatrix}.$$

We can use this to our advantage while computing matvecs with Hankel matrices. To compute the matvec $\boldsymbol{y} = \boldsymbol{H}_s\boldsymbol{v}$, we first write $\boldsymbol{y} = (\boldsymbol{H}_s\boldsymbol{J}_s)(\boldsymbol{J}_s\boldsymbol{v})$ so that

$$\boldsymbol{X}_{2s} \begin{bmatrix} \boldsymbol{J}_s\boldsymbol{v} \\ \boldsymbol{0}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_s\boldsymbol{J}_s & \boldsymbol{B}_s \\ \boldsymbol{B}_s & \boldsymbol{H}_s\boldsymbol{J}_s \end{bmatrix} \begin{bmatrix} \boldsymbol{J}_s\boldsymbol{v} \\ \boldsymbol{0}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ * \end{bmatrix},$$

where $*$ denotes the part of a computation which we can ignore and

$$\boldsymbol{B}_s = \begin{bmatrix} 0 & h_{2s-1} & \ldots & h_{s+2} & h_{s+1} \\ h_1 & 0 & h_{2s-1} & \ldots & h_{s+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{s-2} & \vdots & \ldots & 0 & h_{2s-1} \\ h_{s-1} & h_{s-2} & \ldots & h_1 & 0 \end{bmatrix}.$$

This means that we can also efficiently compute matvecs with Hankel matrices by embedding it within a $2s \times 2s$ circulant matrix $\boldsymbol{X}_{2s}$ defined by the vector

$$\boldsymbol{x}_{2s} = \begin{bmatrix} h_s & h_{s+1} & \ldots & h_{2s-1} & 0 & h_1 & h_2 & \ldots & h_{s-1} \end{bmatrix}^\top.$$

Thus, only the first row and the last column need to be stored to compute matvecs with the Hankel matrix $\boldsymbol{H}_s$. A summary of the algorithm to compute matvecs with $\boldsymbol{H}_s$ is given in Algorithm 2.1.

2.3. Randomized SVD. We can efficiently compute a rank r approximation of an $M \times N$ matrix $\boldsymbol{X}$ using a randomized version of the SVD [8] (henceforth called RandSVD). The idea is to find a matrix $\boldsymbol{Q}$ whose range approximates that of $\boldsymbol{X}$. This is done by first drawing a standard Gaussian random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{N \times (r+\rho)}$, where r is the desired target rank, and $\rho \geq 0$ is an oversampling parameter (typically, $\rho \leq 20$).

Algorithm 2.1. $\boldsymbol{y} = \text{Hankel-matvec}(\boldsymbol{h}_c, \boldsymbol{h}_r, \boldsymbol{v})$.

Input: last column $\boldsymbol{h}_c \in \mathbb{R}^s$, first row $\boldsymbol{h}_r \in \mathbb{R}^s$ of a Hankel matrix $\boldsymbol{H}_s$, vector $\boldsymbol{v} \in \mathbb{R}^s$
Output: matvec $\boldsymbol{y} = \boldsymbol{H}_s\boldsymbol{v} \in \mathbb{R}^s$

1: Form circulant vector $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{h}_c^\top & 0 & \boldsymbol{h}_r(1:\text{end}-1)\end{bmatrix}^\top$
2: Pad the vector $\boldsymbol{v}$ to get $\hat{\boldsymbol{v}} = \begin{bmatrix}\boldsymbol{v} & \boldsymbol{0}_s\end{bmatrix}^\top$
3: Take $\boldsymbol{z} = \text{IFFT}(\text{FFT}(\boldsymbol{x}) \odot \text{FFT}(\hat{\boldsymbol{v}}))$.
4: Extract $\boldsymbol{y} = \boldsymbol{z}(1:s)$

Then, the matrix $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega}$ consists of random linear combinations of the columns of $\boldsymbol{X}$. This means that we can get a matrix $\boldsymbol{Q}$ such that $\mathcal{R}(\boldsymbol{X}) \approx \mathcal{R}(\boldsymbol{Q})$ by taking a thin QR factorization $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$. If $\boldsymbol{X}$ has singular values that decay rapidly, or $\mathsf{rank}(\boldsymbol{X})$ is exactly r, then $\mathcal{R}(\boldsymbol{Q})$ is a good approximation for $\mathcal{R}(\boldsymbol{X})$. We can then approximate $\boldsymbol{X}$ by the low-rank representation $\boldsymbol{X} \approx \boldsymbol{Q}\boldsymbol{Q}^\top\boldsymbol{X}$; this can then be converted into the appropriate SVD format. The procedure is summarized in Algorithm 2.2.

Algorithm 2.2. $[\widehat{\boldsymbol{U}}, \widehat{\boldsymbol{\Sigma}}, \widehat{\boldsymbol{V}}] = \text{RandSVD}(\boldsymbol{X}, r, \rho)$.

Input: matrix $\boldsymbol{X} \in \mathbb{R}^{M\times N}$ with target rank r, oversampling parameter ρ such that $r + \rho \le \min\{M, N\}$, and number of subspace iterations $q \ge 0$
Output: $\widehat{\boldsymbol{U}} \in \mathbb{R}^{M\times r}$, $\widehat{\boldsymbol{\Sigma}} \in \mathbb{R}^{r\times r}$, and $\widehat{\boldsymbol{V}} \in \mathbb{R}^{N\times r}$

1: Draw standard Gaussian random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{N\times(r+\rho)}$
2: Multiply $\boldsymbol{Y} = (\boldsymbol{X}\boldsymbol{X}^\top)^q\boldsymbol{X}\boldsymbol{\Omega}$
3: Compute thin QR factorization $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$
4: Form $\boldsymbol{B} = \boldsymbol{Q}^\top\boldsymbol{X}$
5: Calculate thin SVD $\boldsymbol{B} = \boldsymbol{U}_{\boldsymbol{B}}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
6: Set $\widehat{\boldsymbol{U}} = \boldsymbol{Q}\boldsymbol{U}_{\boldsymbol{B}}(:, 1:r)$, $\widehat{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}(1:r, 1:r)$, and $\widehat{\boldsymbol{V}} = \boldsymbol{V}(:, 1:r)$.

RandSVD is computationally beneficial compared to the full SVD. In this paper, we use a variation of the RandSVD that uses $q \ge 0$ steps of the subspace iteration [8, Algorithm 4.4]. Note that for numerical stability, we perform orthogonalization during and in-between the subspace iterations. Assuming $M \ge N$, the cost of the full SVD is $\mathcal{O}(MN^2)$. On the other hand, if the cost of a matvec with $\boldsymbol{X}$ (or its transpose) is $T_{\boldsymbol{X}}$, then the cost of the randomized SVD can be expressed as

$$\text{Cost} = (2q+1)(r+\rho)T_{\boldsymbol{X}} + \mathcal{O}(r^2(M+N)) \text{ flops.} \tag{2.6}$$

We will use RandSVD in different ways in the system identification algorithms that we derive. We have found RandSVD with $q = 0$–2 subspace iterations to be computationally efficient and sufficiently accurate for our purposes; however, there are several new randomized algorithms developed that have been reviewed in the recent paper [15]. In section 5.1.4, we compare the performance of Algorithm 2.2 to other other randomized SVD algorithms.

3. Randomized algorithms for eigensystem realization.

In this section, we derive two randomized algorithms for efficient computation of the system matrices. The first algorithm accelerates the standard randomized SVD using the block Hankel structure of $\boldsymbol{\mathcal{H}}_s$ (section 3.1); the second algorithm is a randomized variant of the TERA and is applicable when the number of inputs and outputs are large.

3.1. Randomized ERA. Our first approach accelerates the computation of the system matrices by combining two ingredients: first, we replace a reduced SVD of $\boldsymbol{\mathcal{H}}_s$ by a RandSVD to obtain an approximate basis for $\boldsymbol{\mathcal{O}}_s$; second, we additionally exploit the block Hankel structure of $\boldsymbol{\mathcal{H}}_s$ to accelerate the matvecs involving $\boldsymbol{\mathcal{H}}_s$ and $\boldsymbol{\mathcal{H}}_s^\top$ in the RandSVD. As we will show, each of these steps decreases the computational complexity yielding an efficient algorithm overall.

3.1.1. Block Hankel matrices. We first explain how we exploit the block Hankel structure of the matrix $\boldsymbol{\mathcal{H}}_s$ defined in (2.3). The multiplication process for Hankel matrices can be extended to block Hankel matrices. Suppose we have to compute $\boldsymbol{y} = \boldsymbol{\mathcal{H}}_s \boldsymbol{x}$ for a given vector $\boldsymbol{x}$. Let us define the index sets

$$\begin{aligned} \mathcal{I}_i &= \{i, i+\ell, \ldots, i+(s-1)\ell\}, && i = 1, \ldots, \ell, \\ \mathcal{J}_j &= \{j, j+m, \ldots, j+(s-1)m\}, && j = 1, \ldots, m. \end{aligned}$$

If we denote $\boldsymbol{\mathcal{H}}_s(\mathcal{I}_i, \mathcal{J}_j)$ as the $s \times s$ submatrix obtained by extracting the rows and the columns defined by the appropriate index sets, then it is clear that $\boldsymbol{\mathcal{H}}_s(\mathcal{I}_i, \mathcal{J}_j)$ is a Hankel matrix, and that

$$\boldsymbol{y}(\mathcal{I}_i) = \sum_{j=1}^{m} \boldsymbol{\mathcal{H}}_s(\mathcal{I}_i, \mathcal{J}_j) \boldsymbol{x}(\mathcal{J}_j), \qquad i = 1, \ldots, \ell,$$

where $\boldsymbol{y}(\mathcal{I}_i)$ and $\boldsymbol{x}(\mathcal{J}_j)$ are the $s \times 1$ vectors obtained from $\boldsymbol{y}$ and $\boldsymbol{x}$, respectively.

Algorithm 3.1 gives the details of the procedure described here in MATLAB-like notation. It is important to note the following points. First, we need not actually form either the full block Hankel matrix or the intermediate Hankel matrices for each block element. Since the Hankel matrices are defined by the first row and the last column, we extract these quantities from the blocks $\{\boldsymbol{h}_k\}_{k=1}^{2s-1}$ as and when required. Second, since each Hankel matvec costs $\mathcal{O}(s \log_2 s)$ flops, the overall cost of one matvec is $\mathcal{O}(\ell m s \log_2 s)$ flops, compared to $\mathcal{O}(\ell m s^2)$ flops using the naïve approach. Finally, we can easily adapt this algorithm to compute $\boldsymbol{\mathcal{H}}_s^\top \boldsymbol{x}$ as well; the main difference involves taking as inputs the transpose of the Markov parameters $\{\boldsymbol{h}_k^\top\}_{j=1}^{2s-1}$ instead.

Algorithm 3.1. $\boldsymbol{y}$ = Block-Hankel-matvec($\{\boldsymbol{h}_k\}_{k=1}^{2s-1}, \boldsymbol{x}, s$).

Input: blocks $\{\boldsymbol{h}_k\}_{k=1}^{2s-1}$ of Hankel matrix $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{s\ell \times sm}$, vector $\boldsymbol{x} \in \mathbb{R}^{sm}$, dimension $s \geq 1$

Output: matvec $\boldsymbol{y} = \boldsymbol{\mathcal{H}}_s \boldsymbol{x}$

1: **for** $i = 1 : \ell$ **do**
2: **for** $j = 1 : m$ **do**
3: Extract first row $\boldsymbol{j}_r$ and last column $\boldsymbol{j}_c$ as

$$\boldsymbol{j}_r = \begin{bmatrix} \boldsymbol{h}_1(i,j) & \boldsymbol{h}_2(i,j) & \cdots & \boldsymbol{h}_s(i,j) \end{bmatrix},$$

$$\boldsymbol{j}_c = \begin{bmatrix} \boldsymbol{h}_s(i,j) & \boldsymbol{h}_{s+1}(i,j) & \cdots & \boldsymbol{h}_{2s-1}(i,j) \end{bmatrix}^\top$$

4: Compute $\hat{\boldsymbol{y}} =$ Hankel-matvec($\boldsymbol{j}_c, \boldsymbol{j}_r, \boldsymbol{x}(j : m : \text{end})$)
5: Compute $\boldsymbol{y}(i : \ell : \text{end}) = \boldsymbol{y}(i : \ell : \text{end}) + \hat{\boldsymbol{y}}(1 : s)$
6: **end for**
7: **end for**

3.1.2. Randomized ERA. We now incorporate the block Hankel multiplication algorithm, Algorithm 3.1, with RandSVD in order to accelerate the system identification process. This is simple to do; every time we need to multiply $\boldsymbol{\mathcal{H}}_s$ or $\boldsymbol{\mathcal{H}}_s^\top$, we use block Hankel multiplication instead. This is beneficial in two ways: to reduce the computational cost by two orders of magnitude (one from RandSVD and one from using block Hankel structure), and to reduce storage. We need not store $\boldsymbol{\mathcal{H}}_s$ explicitly or even form the full matrix to begin with. All we need are the blocks that make up $\boldsymbol{\mathcal{H}}_s$. This is a major benefit over the standard algorithms.

First, we use the RandSVD algorithm to compute a low-rank approximation of $\boldsymbol{\mathcal{H}}_s$ with target rank $r \leq n$ to obtain $\widehat{\boldsymbol{U}}_r \widehat{\boldsymbol{\Sigma}}_r \widehat{\boldsymbol{V}}_r^\top$. As mentioned earlier, the matvecs involving $\boldsymbol{\mathcal{H}}_s$ or $\boldsymbol{\mathcal{H}}_s^\top$ are handled using Algorithm 3.1. Then, in the system identification phase, we partition the left singular vectors as

$$\widehat{\boldsymbol{U}}_r = \begin{bmatrix} \widehat{\boldsymbol{\Upsilon}}_f \\ * \end{bmatrix} = \begin{bmatrix} * \\ \widehat{\boldsymbol{\Upsilon}}_l \end{bmatrix},$$

where $\widehat{\boldsymbol{\Upsilon}}_f$ and $\widehat{\boldsymbol{\Upsilon}}_l$ are both $(s-1)\ell \times r$. The system matrices can then be recovered as $\widehat{\boldsymbol{A}}_r = \widehat{\boldsymbol{\Upsilon}}_f^\dagger \widehat{\boldsymbol{\Upsilon}}_l$,

$$\widehat{\boldsymbol{C}}_r = \begin{bmatrix} \boldsymbol{I}_\ell & \boldsymbol{0} \end{bmatrix} \widehat{\boldsymbol{U}}_r, \qquad \widehat{\boldsymbol{B}}_r = \widehat{\boldsymbol{\Sigma}}_r \widehat{\boldsymbol{V}}_r^\top \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{0} \end{bmatrix}.$$

As before, the matrix $\boldsymbol{D}_r$ is simply the first Markov parameter $\boldsymbol{h}_0$. We will refer to this randomized ERA that uses block Hankel multiplication as RandSVD-H. Now, we review the computational cost of this algorithm and in section 2.1, we derive error bounds for the recovered system matrices.

Computational cost. We now examine the computational cost of the three system identification algorithms described so far, namely the ERA using a full SVD, ERA using RandSVD, and RandSVD-H. The dominant cost of each algorithm is the SVD step, so we focus our attention there. The complexity of the SVD step for these algorithms is shown in Table 3.1. Recall that the size of the matrix we are computing with is $s\ell \times sm$, the target rank for each RandSVD is r, and the size of the system is n. The cost of the full SVD is then $\mathcal{O}(s^3 m\ell \min\{\ell, m\})$ flops. To analyze the RandSVD based algorithms, we follow the analysis of cost in (2.6). For a standard RandSVD, the cost of a matvec is $\mathcal{O}(s^2 \ell m)$ flops. If the matrix $\boldsymbol{\mathcal{H}}_s$ is stored explicitly as we do in our implementation, then the storage cost is $\mathcal{O}(ms^2\ell)$ entries; however, RandSVD can be implemented without storing $\boldsymbol{\mathcal{H}}_s$ explicitly, in which case the cost of storage is $\mathcal{O}(ms\ell)$ entries. When we use block Hankel structure to accelerate the RandSVD, the cost of a matvec is reduced to $\mathcal{O}(\ell ms \log_2 s)$ flops and the storage cost is also $\mathcal{O}(ms\ell)$ entries. This shows that, as anticipated, replacing the SVD with a

TABLE 3.1

Computational complexity of the dominant step, the SVD step, in each of the four algorithms. Recall that s is the number of block rows and columns in the block Hankel matrix, m is the number of inputs, ℓ is the number of outputs, r is the target rank, and κ and c are the number of iterations used in cross approximation and the dominant volume submatrix parts of the CUR-ERA algorithm.

System ID algorithm	Computational cost	Storage cost
Full SVD	$\mathcal{O}(s^3 m\ell \min\{\ell, m\})$	$\mathcal{O}(s^2 m\ell)$
RandSVD	$\mathcal{O}(rs^2\ell m + r^2 s(\ell + m)$	$\mathcal{O}(s^2 m\ell)$
RandSVD-H	$\mathcal{O}(r\ell ms \log_2 s + r^2 s(\ell + m))$	$\mathcal{O}(sm\ell)$
CUR-ERA	$\mathcal{O}(\kappa r^3 + r^2 s\kappa c(\ell + m))$	$\mathcal{O}(sm\ell)$

RandSVD reduces the cost, and then exploiting the block Hankel structure reduces the cost even further. We also include, for comparison purposes, the computational and storage costs of CUR-ERA [11].

CUR-ERA. The CUR-ERA algorithm relies on the principle of finding a maximum volume submatrix for computing a low-rank approximation, that is, a submatrix of specified dimensions with the largest determinant in absolute magnitude. Finding the maximum volume submatrix is a combinatorial optimization problem and, hence, one has to settle for heuristics to compute a nearly maximum volume submatrix in a reasonable computational time. CUR-ERA uses certain heuristics for finding the cross approximation of a matrix and finding a dominant volume submatrix. Note that in the computational costs in Table 3.1, κ and c are the number of iterations used in the cross approximation and the dominant volume submatrix, respectively. It is clear from the table that our algorithms have a comparable computational cost. Also, the randomized SVD algorithm does not involve a combinatorial optimization problem, is known to be computationally efficient and accurate for a range of problems, and has well-developed error analysis. This makes the RandSVD-H beneficial in practical applications. In addition, as we will show in numerical experiments in section 5, our algorithms are more accurate.

3.2. Randomized TERA. This approach is inspired by the tangential interpolation approach for model reduction. The goal of TERA is to reduce the dimension of the Markov parameters by projecting the parameters into a lower dimensional space. This reduces the size of the block Hankel matrix but preserves the block Hankel structure, therefore, reducing the computational cost of the SVD step [12]. We briefly review the TERA approach and describe our acceleration using RandSVD.

TERA. In this approach, we seek two orthogonal projection matrices

$$\begin{aligned} \boldsymbol{P}_1 &= \boldsymbol{W}_1 \boldsymbol{W}_1^\top \qquad \mathsf{rank}\,(\boldsymbol{W}_1) = \ell', \\ \boldsymbol{P}_2 &= \boldsymbol{W}_2 \boldsymbol{W}_2^\top \qquad \mathsf{rank}\,(\boldsymbol{W}_1) = m', \end{aligned}$$

where the matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ have orthonormal columns. To compute $\boldsymbol{P}_1$, we first arrange the Markov parameters in the matrix

$$\boldsymbol{\mathcal{H}}_w = \begin{bmatrix} \boldsymbol{h}_1 & \dots & \boldsymbol{h}_{2s-1} \end{bmatrix} \in \mathbb{R}^{\ell \times m(2s-1)}. \tag{3.1}$$

We then solve the optimization problem

$$\min_{\mathsf{rank}\,(\boldsymbol{P})=\ell'} \|\boldsymbol{P}\boldsymbol{\mathcal{H}}_w - \boldsymbol{\mathcal{H}}_w\|_F^2.$$

The optimal solution can be constructed using the SVD of $\boldsymbol{\mathcal{H}}_w = \boldsymbol{U}_w \boldsymbol{\Sigma}_w \boldsymbol{V}_w^\top$. We take $\boldsymbol{W}_1 = \boldsymbol{U}_w(:, 1:\ell')$; that is, we take $\boldsymbol{W}_1$ to be the first ℓ' left singular vectors of $\boldsymbol{\mathcal{H}}_w$. Similarly, to construct the matrix $\boldsymbol{W}_2$, we first arrange the Markov parameters into the matrix

$$\boldsymbol{\mathcal{H}}_e = \begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_{2s-1} \end{bmatrix} \in \mathbb{R}^{\ell(2s-1) \times m}. \tag{3.2}$$

We then solve the optimization problem

$$\min_{\mathsf{rank}\,(\boldsymbol{P})=m'} \|\boldsymbol{\mathcal{H}}_e - \boldsymbol{\mathcal{H}}_e \boldsymbol{P}\|_F^2.$$

The optimal solution can be constructed using the SVD of $\boldsymbol{\mathcal{H}}_e = \boldsymbol{U}_e \boldsymbol{\Sigma}_e \boldsymbol{V}_e^\top$. We take $\boldsymbol{W}_2 = \boldsymbol{V}_e(:, 1:m')$; that is, we take $\boldsymbol{W}_2$ to be the first m' right singular vectors of $\boldsymbol{\mathcal{H}}_e$. Having obtained the matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, we construct the projected Markov parameters as

$$\widetilde{\boldsymbol{h}}_i = \boldsymbol{W}_1^\top \boldsymbol{h}_i \boldsymbol{W}_2 \in \mathbb{R}^{\ell' \times m'}, \qquad i = 1, \ldots, 2s-1.$$

The block Hankel matrix $\boldsymbol{\mathcal{H}}_s$ is "projected" using the matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ arranged in diagonal blocks to obtain the reduced-size block Hankel matrix

(3.3)
$$\widetilde{\boldsymbol{\mathcal{H}}}_s = \begin{bmatrix} \boldsymbol{W}_1^\top & & & \\ & \boldsymbol{W}_1^\top & & \\ & & \ddots & \\ & & & \boldsymbol{W}_1^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \ldots & \boldsymbol{h}_s \\ \boldsymbol{h}_2 & \boldsymbol{h}_3 & \ldots & \boldsymbol{h}_{s+1} \\ \vdots & \vdots & \iddots & \vdots \\ \boldsymbol{h}_s & \boldsymbol{h}_{s+1} & \ldots & \boldsymbol{h}_{2s-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_2 & & & \\ & \boldsymbol{W}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{W}_2 \end{bmatrix}.$$

It is important to observe that due to this projection, the block Hankel structure is preserved, and we can express $\widetilde{\boldsymbol{\mathcal{H}}}_s$ in terms of the projected Markov parameters as

(3.4)
$$\widetilde{\boldsymbol{\mathcal{H}}}_s = \begin{bmatrix} \widetilde{\boldsymbol{h}}_1 & \widetilde{\boldsymbol{h}}_2 & \ldots & \widetilde{\boldsymbol{h}}_s \\ \widetilde{\boldsymbol{h}}_2 & \widetilde{\boldsymbol{h}}_3 & \ldots & \widetilde{\boldsymbol{h}}_{s+1} \\ \vdots & \vdots & \iddots & \vdots \\ \widetilde{\boldsymbol{h}}_s & \widetilde{\boldsymbol{h}}_{s+1} & \ldots & \widetilde{\boldsymbol{h}}_{2s-1} \end{bmatrix} \in \mathbb{R}^{(\ell' s) \times (m' s)}.$$

If $\ell' \ll \ell$ and $m' \ll m$, then the size of the matrix $\widetilde{\boldsymbol{\mathcal{H}}}_s$ is much less than $\boldsymbol{\mathcal{H}}_s$. The dimensions ℓ' and m' are determined by the singular value decay of the matrices $\boldsymbol{\mathcal{H}}_w$ and $\boldsymbol{\mathcal{H}}_e$. Retaining a larger number of singular vectors (that is, large ℓ' and m') results in a more accurate approximation to $\boldsymbol{\mathcal{H}}_s$ but results in a larger matrix $\widetilde{\boldsymbol{\mathcal{H}}}_s$ and in a higher computational cost.

Recovering system matrices. The next steps mimic the standard ERA approach to reconstruct the system matrices but we must carefully account for the dimensions of the projected system. We compute the approximate SVD of $\widetilde{\boldsymbol{\mathcal{H}}}_s = \widetilde{\boldsymbol{U}}_r \widetilde{\boldsymbol{\Sigma}}_r \widetilde{\boldsymbol{V}}_r^\top$, and partition the left singular vectors as

$$\widetilde{\boldsymbol{U}}_r = \begin{bmatrix} \widetilde{\boldsymbol{\Upsilon}}_f \\ * \end{bmatrix} = \begin{bmatrix} * \\ \widetilde{\boldsymbol{\Upsilon}}_l \end{bmatrix}.$$

The system matrices can be recovered as

(3.5)
$$\widetilde{\boldsymbol{A}}_r = \widetilde{\boldsymbol{\Upsilon}}_f^\dagger \widetilde{\boldsymbol{\Upsilon}}_l \in \mathbb{R}^{r \times r}, \qquad \widetilde{\boldsymbol{C}}_r = \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{0} \end{bmatrix} \widetilde{\boldsymbol{U}}_r, \qquad \widetilde{\boldsymbol{B}}_r = \widetilde{\boldsymbol{\Sigma}}_r \widetilde{\boldsymbol{V}}_r^\top \begin{bmatrix} \boldsymbol{W}_2^\top \\ \boldsymbol{0} \end{bmatrix}.$$

Computational cost. To motivate the need for accelerating TERA with RandSVD, we first review the computational cost which has three components. Computing the projection matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ involves a computational cost of $\mathcal{O}\left(s(\ell m^2 + m\ell^2)\right)$ flops. The SVD of $\widetilde{\boldsymbol{\mathcal{H}}}_s$ now costs

$$\mathcal{O}(s^3 \ell' m' \min\{\ell', m'\}) \text{ flops},$$

and, similarly to earlier algorithms, the cost recovery of the system matrices is negligible compared to the cost of the SVD. Although the computational cost is significantly reduced (assuming $\ell' \ll \ell$ and $m' \ll m$), the cost of the SVD still dominates the computational cost and has cubic scaling with s. We now propose a new algorithm to lower this cost.

Algorithm 3.2. RandTERA.

Input: blocks $\{\boldsymbol{h}_k\}_{k=1}^{2s-1}$ of the block Hankel matrix $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{s\ell \times sm}$, target rank $1 \leq r \leq n$, integers $\ell' \leq \ell$, $m' \leq m$.

1: Form $\boldsymbol{\mathcal{H}}_w$ using (3.1) and compute its SVD $\boldsymbol{U}_w \boldsymbol{\Sigma}_w \boldsymbol{V}_w^\top$. Set $\boldsymbol{W}_1 = \boldsymbol{U}_w(:, 1:\ell')$

2: Form $\boldsymbol{\mathcal{H}}_e$ using (3.2) and compute its SVD $\boldsymbol{U}_e \boldsymbol{\Sigma}_e \boldsymbol{V}_e^\top$. Set $\boldsymbol{W}_2 = \boldsymbol{V}_e(:, 1:m')$

3: Form $\widehat{\boldsymbol{h}}_i = \boldsymbol{W}_1^\top \boldsymbol{h}_i \boldsymbol{W}_2$ for $i = 1, \ldots, 2s-1$.

4: Compute $[\widetilde{\boldsymbol{U}}_r, \widetilde{\boldsymbol{\Sigma}}_r, \widetilde{\boldsymbol{V}}_r]$=RandSVD($\widetilde{\boldsymbol{\mathcal{H}}}_s, r, \rho$). {Matvecs are computed using Algorithm 3.1}

5: Compute system matrices using (3.5)

TABLE 3.2
Computational cost of computing RandTERA.

Stage	Computational cost (flops)
Computing $\boldsymbol{W}_1, \boldsymbol{W}_2$	$\mathcal{O}\left(s(\ell m^2 + m\ell^2)\right)$
SVD step	$\mathcal{O}\left(r\ell' m' s \log_2 s + r^2 s(m' + \ell')\right)$

RandTERA. It is worth pointing out that the projected Hankel matrix $\widetilde{\boldsymbol{\mathcal{H}}}_s$ still retains its block Hankel structure. To reduce the computational cost, we combine the following ingredients that were used previously: we use the RandSVD to compute the rank r approximation to $\widetilde{\boldsymbol{\mathcal{H}}}_s$, and the matvecs involving $\widetilde{\boldsymbol{\mathcal{H}}}_s$ can be accelerated using the block Hankel structure (as described in section 3.1.1). The main difference is that we use the projected Markov parameters $\{\widetilde{\boldsymbol{h}}_i\}_{i=1}^{2s-1}$ rather than the Markov parameters $\{\boldsymbol{h}_i\}_{i=1}^{2s-1}$. As a result, the computational cost of the SVD step is reduced to

$$\mathcal{O}\left(r\ell' m' s \log_2 s + r^2 s(m' + \ell')\right) \text{ flops.}$$

We call this algorithm RandTERA, and a complete description of this algorithm is given in Algorithm 3.2. A breakdown of the computational cost of the RandTERA is given below in Table 3.2.

Regarding the storage cost, since only the projected Markov parameters need to be stored, the storage cost is $\mathcal{O}(sm'\ell')$ entries, which is potentially lower than RandSVD-H which requires $\mathcal{O}(sm\ell)$ entries.

4. Error analysis. In this section, we analyze the accuracy of RandERA in section 3.1, and derive bounds on the accuracy of the recovered system matrix $\boldsymbol{A}_r$. Our derivation is not tied to the randomized algorithms used to compute the approximations; this makes the analysis applicable to more general settings.

4.1. Background and assumptions. We first review necessary background information and clearly state our assumptions needed for the analysis.

Canonical angles. Given $\boldsymbol{\mathcal{H}}_s = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, partition $\boldsymbol{U}_r = \boldsymbol{U}(:, 1:r)$, the left singular vectors corresponding to the top r singular values of $\boldsymbol{\mathcal{H}}_s$, as

$$\boldsymbol{U}_r = \begin{bmatrix} \boldsymbol{\Upsilon}_f \\ * \end{bmatrix} = \begin{bmatrix} * \\ \boldsymbol{\Upsilon}_l \end{bmatrix} \in \mathbb{R}^{s\ell \times r}.$$

Note that we have dropped the superscripts (r), compared to section 2.1, to make the notation manageable. We can recover the matrix $\boldsymbol{A}_r$ as $\boldsymbol{A}_r = \boldsymbol{\Upsilon}_f^\dagger \boldsymbol{\Upsilon}_l$. Similarly, let $\widehat{\boldsymbol{\mathcal{H}}}_s = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}\widehat{\boldsymbol{V}}^\top$ be an approximation to $\boldsymbol{\mathcal{H}}_s$ with left singular vectors $\widehat{\boldsymbol{U}}_r$ partitioned

as

$$\widehat{\boldsymbol{U}}_r = \begin{bmatrix} \widehat{\boldsymbol{\Upsilon}}_f \\ * \end{bmatrix} = \begin{bmatrix} * \\ \widehat{\boldsymbol{\Upsilon}}_l \end{bmatrix} \in \mathbb{R}^{s\ell \times r},$$

and we can compute the approximate system matrix $\widehat{\boldsymbol{A}}_r = \widehat{\boldsymbol{\Upsilon}}_f^{\dagger} \widehat{\boldsymbol{\Upsilon}}_l \in \mathbb{R}^{r \times r}$. Let the canonical angles between the subspaces $\mathcal{R}(\boldsymbol{U}_r)$ and $\mathcal{R}(\widehat{\boldsymbol{U}}_r)$ be denoted by

$$0 \leq \theta_1 \leq \cdots \leq \theta_{max} \leq \pi/2.$$

If we collect the angles into a diagonal matrix $\boldsymbol{\Theta}$, then $\boldsymbol{U}_r^{\top} \widehat{\boldsymbol{U}}_r$ has the SVD

$$\boldsymbol{U}_r^{\top} \widehat{\boldsymbol{U}}_r = \boldsymbol{P}(\cos \boldsymbol{\Theta}) \boldsymbol{Q}^{\top}.$$

Let $\boldsymbol{\mathcal{P}}_{\boldsymbol{U}_r}$ denote the orthogonal projector onto $\mathcal{R}(\boldsymbol{U}_r)$; similarly, let $\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{U}}_r}$ denote the orthogonal projector onto $\mathcal{R}(\widehat{\boldsymbol{U}}_r)$. Then, the distance between the two subspace $\mathcal{R}(\boldsymbol{U}_r)$ and $\mathcal{R}(\widehat{\boldsymbol{U}}_r)$ is given by

$$\|\boldsymbol{\mathcal{P}}_{\boldsymbol{U}_r} - \boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{U}}_r}\|_2 = \|(\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{U}_r}) \boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{U}}_r}\|_2 = \|\sin \boldsymbol{\Theta}\|_2 = \sin \theta_{\max}.$$

See [21, Chapter II.4] for more details on canonical angles between subspaces.

Pseudoinverses. We recall some facts about the perturbation of pseudoinverses [4, section 2.2.2]. If $\boldsymbol{M}, \boldsymbol{E} \in \mathbb{R}^{n \times r}$ such that $\mathsf{rank}(\boldsymbol{M}) = \mathsf{rank}(\boldsymbol{M} + \boldsymbol{E}) = r$, then

$$\|(\boldsymbol{M} + \boldsymbol{E})^{\dagger} - \boldsymbol{M}^{\dagger}\|_2 \leq \sqrt{2} \|\boldsymbol{M}^{\dagger}\|_2 \|(\boldsymbol{M} + \boldsymbol{E})^{\dagger}\|_2 \|\boldsymbol{E}\|_2. \tag{4.1}$$

Furthermore, if $\|\boldsymbol{E}\|_2 \|\boldsymbol{M}^{\dagger}\|_2 < 1$, then

$$\|(\boldsymbol{M} + \boldsymbol{E})^{\dagger}\|_2 \leq \frac{\|\boldsymbol{M}^{\dagger}\|_2}{1 - \|\boldsymbol{E}\|_2 \|\boldsymbol{M}^{\dagger}\|_2}. \tag{4.2}$$

If $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times n}$ such that $\mathsf{rank}(\boldsymbol{A}) = \mathsf{rank}(\boldsymbol{B}) = r$, then by [4, Theorem 2.2.3]

$$(\boldsymbol{AB})^{\dagger} = \boldsymbol{B}^{\dagger} \boldsymbol{A}^{\dagger}. \tag{4.3}$$

Accuracy of eigenvalues. A norm based approach, i.e., $\|\boldsymbol{A} - \widehat{\boldsymbol{A}}_r\|$, is not meaningful since $\boldsymbol{A}_r$ (and $\widehat{\boldsymbol{A}}_r$) can only be determined up to a similarity transformation. To compare the approximate system matrix $\widehat{\boldsymbol{A}}_r$ with $\boldsymbol{A}_r$, we compare the eigenvalues of these two matrices since the eigenvalues remain unchanged by a similarity transformation. We measure the accuracy of the eigenvalues of $\widehat{\boldsymbol{A}}$ using the spectral variation, which we now define. Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$; the spectral variation between $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined as [3, section VI.3]

$$\mathsf{sv}(\psi(\boldsymbol{B}), \psi(\boldsymbol{A})) = \max_{1 \leq j \leq n} \min_{1 \leq i \leq n} |\lambda_i(\boldsymbol{A}) - \lambda_j(\boldsymbol{B})|, \tag{4.4}$$

where $\psi(\cdot)$ denotes the spectrum of the matrix.

We briefly list the various assumptions that are required for our main result.

Assumption 4.1. We assume the following:

A1. System matrix: Assume that $\boldsymbol{A}_r = \boldsymbol{\Upsilon}_f^{\dagger} \boldsymbol{\Upsilon}_l \in \mathbb{R}^{r \times r}$ is diagonalizable and let $\boldsymbol{W} \in \mathbb{R}^{r \times r}$ be the matrix of eigenvectors.

A2. Singular vectors: Assume that the subblocks $\boldsymbol{\Upsilon}_f$ of the singular vectors $\boldsymbol{U}_r$ are full rank, that is $\mathsf{rank}\,(\boldsymbol{\Upsilon}_f) = r$.

A3. Markov parameters: Assume the Markov parameters converge to $\boldsymbol{h}_i \to \boldsymbol{0}$ for $i > s$.

A4. Canonical angles: Assume that the canonical angles are sufficiently small and satisfy

$$\eta \equiv 2 \sin \theta_{\max} \|\boldsymbol{\Upsilon}_f^\dagger\|_2 < 1. \tag{4.5}$$

Assumption A1 is rather strong but can be weakened if different perturbation results are used; see [3, Chapter VIII.1]. Assumption A2 ensures that the least-squares problem $\min_{\boldsymbol{A}} \|\boldsymbol{\Upsilon}_f \boldsymbol{A} - \boldsymbol{\Upsilon}_l\|_F$ has a unique solution. Assumption A3 is also made in [13], and is necessary to ensure the stability of the system. Finally, Assumption A4 ensures that the approximate singular vectors are sufficiently accurate.

4.2. Main result. We are ready to state our main theorem.

THEOREM 4.2. *With the notation in section* 4.1 *and Assumption* 4.1

$$\mathsf{sv}\,(\psi(\widehat{\boldsymbol{A}}_r), \psi(\boldsymbol{A}_r)) \le \kappa_2(\boldsymbol{W}) \eta \left(1 + \frac{\sqrt{2}\|\boldsymbol{\Upsilon}_f^\dagger\|_2}{1-\eta}\right),$$

where $\kappa_2(\boldsymbol{W}) = \|\boldsymbol{W}\|_2 \|\boldsymbol{W}^{-1}\|_2$ *is the condition number of the eigenvectors* $\boldsymbol{W}$ *of* $\boldsymbol{A}_r$.

The theorem identifies several factors that can cause large errors during the identification step. First, the eigenvectors of $\boldsymbol{W}$ may be ill-conditioned, i.e., $\kappa_2(\boldsymbol{W})$ can be large. The best case scenario is when $\boldsymbol{A}$ is normal, so that the condition number is 1. Second, the norm of the pseudoinverse $\|\boldsymbol{\Upsilon}_f^\dagger\|_2 \ge 1$ can be large. However, Assumption A3 ensures that $\lim_{s\to\infty} \|\boldsymbol{\Upsilon}_f\|_2 = 1$; see [11, Lemma 5] for a detailed argument. Finally, if the singular vectors of the low-rank approximation are not computed accurately, then the canonical angles can be large, i.e., $\sin\theta_{\max}$ can be close to 1. This can be mitigated, for example, by taking several subspace iterations. The first three assumptions are intrinsic to the system $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ and only Assumption A4 depends on the choice of the numerical method. The theorem is applicable to the randomized algorithms developed here, namely, RandERA and RandTERA. However, it is important in a larger context, since it is applicable to any approximation algorithm for ERA, including TERA.

Proof of Theorem 4.2. There are several steps involved in this proof.

1. *Introducing a similarity transformation.* Let $\boldsymbol{Z} \in \mathbb{R}^{r\times r}$ be an orthogonal matrix; we will leave the specific choice of this matrix to step 2. Since $\boldsymbol{A}_r$ is diagonalizable, using a perturbation theorem for diagonalizable matrices [3, Theorem VIII.3.1], we have

$$\mathsf{sv}\,(\psi(\widehat{\boldsymbol{A}}_r), \psi(\boldsymbol{A}_r)) = \mathsf{sv}\,(\psi(\boldsymbol{Z}\widehat{\boldsymbol{A}}_r\boldsymbol{Z}^\top), \psi(\boldsymbol{A})) \le \kappa_2(\boldsymbol{W}) \|\boldsymbol{A}_r - \boldsymbol{Z}\widehat{\boldsymbol{A}}_r\boldsymbol{Z}^\top\|_2. \tag{4.6}$$

The equality in the above equation is because the eigenvalues of $\widehat{\boldsymbol{A}}_r$ and $\boldsymbol{Z}\widehat{\boldsymbol{A}}_r\boldsymbol{Z}^\top$ are the same. We first discuss the choice of $\boldsymbol{Z}$ before analyzing the error in the term $\|\boldsymbol{A}_r - \boldsymbol{Z}\widehat{\boldsymbol{A}}_r\boldsymbol{Z}^\top\|_2$.

2. *Choosing* $\boldsymbol{Z}$. Recall that the SVD of $\boldsymbol{U}_r^\top \widehat{\boldsymbol{U}}_r$ is $\boldsymbol{P}(\cos\boldsymbol{\Theta})\boldsymbol{Q}^\top$. Using the unitary invariance of the 2-norm, $\|\boldsymbol{U}_r - \widehat{\boldsymbol{U}}_r\boldsymbol{Z}^\top\|_2 = \|\boldsymbol{U}_r\boldsymbol{Z} - \widehat{\boldsymbol{U}}_r\|_2$. Now, we choose $\boldsymbol{Z} = \boldsymbol{P}\boldsymbol{Q}^\top$ and verify that $\boldsymbol{Z}$ is orthogonal. By the triangle inequality and the unitary invariance

of the 2-norm,

$$\begin{aligned}\|\boldsymbol{U}_r\boldsymbol{Z}-\widehat{\boldsymbol{U}}_r\|_2 &\le \|\boldsymbol{U}_r\boldsymbol{Z}-\boldsymbol{U}_r\boldsymbol{U}_r^\top\widehat{\boldsymbol{U}}_r\|_2+\|(\boldsymbol{I}-\boldsymbol{U}_r\boldsymbol{U}_r^\top)\widehat{\boldsymbol{U}}_r\|_2\\ &\le \|\boldsymbol{U}_r\boldsymbol{Z}-\boldsymbol{U}_r\boldsymbol{P}(\cos\boldsymbol{\Theta})\boldsymbol{Q}^\top\|_2+\|(\boldsymbol{I}-\boldsymbol{U}_r\boldsymbol{U}_r^\top)\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{U}}_r}\widehat{\boldsymbol{U}}_r\|_2\\ &\le \|\boldsymbol{P}\boldsymbol{Q}^\top-\boldsymbol{P}(\cos\boldsymbol{\Theta})\boldsymbol{Q}^\top\|_2+\|(\boldsymbol{I}-\boldsymbol{\mathcal{P}}_{\boldsymbol{U}_r})\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{U}}_r}\|_2\|\widehat{\boldsymbol{U}}_r\|_2\\ &\le \|\boldsymbol{I}-\cos\boldsymbol{\Theta}\|_2+\|\sin\boldsymbol{\Theta}\|_2.\end{aligned}\tag{4.7}$$

Since the canonical angles satisfy $0\le\theta_i\le\pi/2$, we have $\cos^2\theta_i\le\cos\theta_i$ and, therefore, for $i=1,\dots,r$,

$$(1-\cos\theta_i)^2=1-2\cos\theta_i+\cos^2\theta_i\le 1-\cos^2\theta_i=\sin^2\theta_i.$$

Therefore, $\|\boldsymbol{I}-\cos\boldsymbol{\Theta}\|_2\le\|\sin\boldsymbol{\Theta}\|_2$. Together with (4.7), we have

$$\|\boldsymbol{U}_r-\widehat{\boldsymbol{U}}_r\boldsymbol{Z}^\top\|_2=\|\boldsymbol{U}_r\boldsymbol{Z}-\widehat{\boldsymbol{U}}_r\|_2\le 2\|\sin\boldsymbol{\Theta}\|_2.\tag{4.8}$$

3. *Ensuring* $\mathsf{rank}\,(\widehat{\boldsymbol{\Upsilon}}_f)=r$. Choose $\boldsymbol{M}=\boldsymbol{\Upsilon}_f$, which has rank r, and $\boldsymbol{M}+\boldsymbol{E}=\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top$. We now show that $\boldsymbol{M}+\boldsymbol{E}$ also has rank r. Using Weyl's theorem on perturbation of singular values [4, Theorem 2.2.8],

$$|\sigma_r(\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top)-\sigma_r(\boldsymbol{\Upsilon}_f)|\le\|\boldsymbol{E}\|_2=\|\boldsymbol{\Upsilon}_f-\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top\|_2\le\|\boldsymbol{U}_r-\widehat{\boldsymbol{U}}_r\boldsymbol{Z}^\top\|_2\le 2\|\sin\boldsymbol{\Theta}\|_2.$$

We have used the fact that $\boldsymbol{\Upsilon}_f$ and $\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top$ are submatrices of $\boldsymbol{U}_r$ and $\widehat{\boldsymbol{U}}_r\boldsymbol{Z}^\top$, respectively. Since $\sigma_r(\boldsymbol{\Upsilon}_f)=1/\|\boldsymbol{\Upsilon}_f^\dagger\|_2$, we can rearrange to get

$$\sigma_r(\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top)\ge\frac{1}{\|\boldsymbol{\Upsilon}_f^\dagger\|_2}-2\|\sin\boldsymbol{\Theta}\|_2>0,$$

since by assumption, $\eta=2\|\sin\boldsymbol{\Theta}\|_2\|\boldsymbol{\Upsilon}_f^\dagger\|_2<1$. This shows that both $\widehat{\boldsymbol{\Upsilon}}_f$ and $\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top$ have rank r. Therefore, by (4.3), $(\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top)^\dagger=\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger$. Furthermore, (4.2) applies and

$$\|\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger\|_2\le\frac{\|\boldsymbol{\Upsilon}_f^\dagger\|_2}{1-2\|\sin\boldsymbol{\Theta}\|_2\|\boldsymbol{\Upsilon}_f^\dagger\|_2}.\tag{4.9}$$

4. *Error in* $\|\boldsymbol{A}_r-\boldsymbol{Z}\widehat{\boldsymbol{A}}_r\boldsymbol{Z}^\top\|_2$. Write both matrices in terms of their factors, and add and subtract the term $\boldsymbol{\Upsilon}_f^\dagger\widehat{\boldsymbol{\Upsilon}}_l\boldsymbol{Z}^\top$, to get

$$\begin{aligned}\|\boldsymbol{A}_r-\boldsymbol{Z}\widehat{\boldsymbol{A}}_r\boldsymbol{Z}^\top\|_2&=\|\boldsymbol{\Upsilon}_f^\dagger\boldsymbol{\Upsilon}_l-\boldsymbol{\Upsilon}_f^\dagger\widehat{\boldsymbol{\Upsilon}}_l\boldsymbol{Z}^\top+\boldsymbol{\Upsilon}_f^\dagger\widehat{\boldsymbol{\Upsilon}}_l\boldsymbol{Z}^\top-\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger\widehat{\boldsymbol{\Upsilon}}_l\boldsymbol{Z}^\top\|_2\\ &\le\|\boldsymbol{\Upsilon}_f^\dagger\|_2\|\boldsymbol{\Upsilon}_l-\widehat{\boldsymbol{\Upsilon}}_l\boldsymbol{Z}^\top\|_2+\|\boldsymbol{\Upsilon}^\dagger-\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger\|_2\|\widehat{\boldsymbol{\Upsilon}}_l\boldsymbol{Z}^\top\|_2.\end{aligned}\tag{4.10}$$

Note that, by Assumption A2, $\mathsf{rank}\,(\boldsymbol{\Upsilon}_f)=r$. In step 3, we showed that $\mathsf{rank}\,(\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top)=r$. As before, with $\boldsymbol{M}=\boldsymbol{\Upsilon}_f$ and $\boldsymbol{M}+\boldsymbol{E}=\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top$, (4.1) applies and

$$\|\boldsymbol{\Upsilon}_f^\dagger-\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger\|_2\le\sqrt{2}\|\boldsymbol{\Upsilon}_f^\dagger\|_2\|\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger\|_2\|\boldsymbol{\Upsilon}_f-\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top\|_2.$$

Plugging this into (4.10), we have

$$\|\boldsymbol{A}_r-\boldsymbol{Z}\widehat{\boldsymbol{A}}_r\boldsymbol{Z}^\top\|_2\le\|\boldsymbol{\Upsilon}_f^\dagger\|_2\left(\|\boldsymbol{\Upsilon}_l-\widehat{\boldsymbol{\Upsilon}}_l\boldsymbol{Z}^\top\|_2+\sqrt{2}\|\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger\|_2\|\boldsymbol{\Upsilon}_f-\widehat{\boldsymbol{\Upsilon}}_f\boldsymbol{Z}^\top\|_2\right).$$

We have used the fact that $\|\widehat{\boldsymbol{\Upsilon}}_l \boldsymbol{Z}^\top\|_2 \leq \|\widehat{\boldsymbol{U}}_r \boldsymbol{Z}^\top\|_2 = 1$; the inequality is because $\widehat{\boldsymbol{\Upsilon}}_l \boldsymbol{Z}^\top$ is a submatrix of $\widehat{\boldsymbol{U}}_r \boldsymbol{Z}^\top$ and the equality follows since $\widehat{\boldsymbol{U}}_r \boldsymbol{Z}^\top$ has orthonormal columns. Similarly,

$$\|\boldsymbol{\Upsilon}_l - \widehat{\boldsymbol{\Upsilon}}_l \boldsymbol{Z}^\top\|_2 \leq \|\boldsymbol{U}_r - \widehat{\boldsymbol{U}}_r \boldsymbol{Z}^\top\|_2, \qquad \|\boldsymbol{\Upsilon}_f - \widehat{\boldsymbol{\Upsilon}}_f \boldsymbol{Z}^\top\|_2 \leq \|\boldsymbol{U}_r - \widehat{\boldsymbol{U}}_r \boldsymbol{Z}^\top\|_2,$$

so that

$$\|\boldsymbol{A}_r - \boldsymbol{Z}\widehat{\boldsymbol{A}}_r \boldsymbol{Z}^\top\|_2 \leq \|\boldsymbol{\Upsilon}_f^\dagger\|_2 \|\boldsymbol{U}_r - \widehat{\boldsymbol{U}}_r \boldsymbol{Z}^\top\|_2 \left(1 + \sqrt{2}\|\boldsymbol{Z}\widehat{\boldsymbol{\Upsilon}}_f^\dagger\|_2\|\right). \tag{4.11}$$

5. *Finishing the proof.* Plug (4.8) and (4.9) into (4.11) to obtain

$$\|\boldsymbol{A}_r - \boldsymbol{Z}\widehat{\boldsymbol{A}}_r \boldsymbol{Z}^\top\|_2 \leq (2\|\sin\boldsymbol{\Theta}\|_2 \|\boldsymbol{\Upsilon}_f^\dagger\|_2) \left(1 + \frac{\sqrt{2}\|\boldsymbol{\Upsilon}_f^\dagger\|_2}{1 - 2\|\sin\boldsymbol{\Theta}\|_2 \|\boldsymbol{\Upsilon}_f^\dagger\|_2}\right).$$

Plug this bound into (4.6) to finish the proof. ∎

An important aspect of the proof of Theorem 4.2 is the choice of the orthogonal matrix $\boldsymbol{Z}$ that determines the similarity transformation. Our proof used this idea from [9, Theorem 5]. It is worth pointing out that the matrix $\boldsymbol{Z} = \boldsymbol{P}\boldsymbol{Q}^\top$ is also the solution to the orthogonal Procrustes problem

$$\min_{\substack{\boldsymbol{Z} \in \mathbb{R}^{r \times r} \\ \boldsymbol{Z}^\top \boldsymbol{Z} = \boldsymbol{I}_r}} \|\boldsymbol{U}_r \boldsymbol{Z} - \widehat{\boldsymbol{U}}_r\|_F.$$

Therefore, the matrix $\boldsymbol{Z} = \boldsymbol{P}\boldsymbol{Q}^\top$ is the matrix that "best rotates" the columns $\boldsymbol{U}_r$ to align with the columns of $\widehat{\boldsymbol{U}}_r$. Note, however, that in the proof we use the 2-norm instead of the Frobenius norm.

4.3. Accuracy of the singular vectors. Theorem 4.2 shows that the accuracy of the approximation algorithms to the ERA depend on the canonical angles between the exact and approximate subspaces $\mathcal{R}(\boldsymbol{U}_r)$ and $\mathcal{R}(\widehat{\boldsymbol{U}}_r)$. Insight into the accuracy between these subspaces can be obtained by standard perturbation theory. Suppose there are numbers $\alpha \geq 0$ and $\delta > 0$ such that

$$\sigma_{r+1}(\widehat{\boldsymbol{\mathcal{H}}}_s) \geq \alpha + \delta, \qquad \sigma_r(\boldsymbol{H}_s) \leq \alpha.$$

Let $\widehat{\boldsymbol{U}}_r, \widehat{\boldsymbol{V}}_r$ be the matrices containing the left and right singular vectors corresponding to the first r singular values of $\widehat{\boldsymbol{\mathcal{H}}}_s$, which are taken to be the diagonals of the matrix $\widehat{\boldsymbol{\Sigma}}_r$. Then, by [21, Chapter V.4, Theorem 4.4],

$$\|\sin\boldsymbol{\Theta}\|_2 \leq \frac{\max\{\|\boldsymbol{\mathcal{H}}_s \widehat{\boldsymbol{V}}_r - \widehat{\boldsymbol{U}}_r \widehat{\boldsymbol{\Sigma}}_r\|_2, \|\boldsymbol{\mathcal{H}}_s^\top \widehat{\boldsymbol{U}}_r - \widehat{\boldsymbol{V}}_r \widehat{\boldsymbol{\Sigma}}_r\|_2\}}{\delta}.$$

In section 3.1, the randomized subspace iteration is used to construct the low-rank approximation $\widehat{\boldsymbol{\mathcal{H}}}_s$. We can use more special purpose error bounds to quantify the accuracy of the singular vectors. Specifically, one may use techniques developed in section 3.2 of [19]. We omit a detailed statement of the results.

4.4. Stability. The discrete dynamical system with the system matrix $\boldsymbol{A}_r$ is stable if the eigenvalues of $\boldsymbol{A}_r$ lie inside the unit circle; that is, the spectral radius $\rho(\boldsymbol{A}_r) < 1$. Assumption A3 in section 4.1 means that the Markov parameters decay after some finite time. Then, by [11, Theorem 3], there exists a positive integer s

such that the identified reduced order model is a stable discrete dynamical system. Theorem 4.2 can also be used to determine when the approximate discrete dynamical system with the system matrix $\widehat{\boldsymbol{A}}_r$ is stable.

We consider the spectral radius of $\widehat{\boldsymbol{A}}_r$, and write

$$\rho(\widehat{\boldsymbol{A}}_r) = \max_{1\leq j\leq r} |\lambda_j(\widehat{\boldsymbol{A}}_r)| = |\lambda_{j^*}(\widehat{\boldsymbol{A}}_r)|$$

for some index j^*. Similarly, let i^* be the index such that $\lambda_{i^*}(\boldsymbol{A}_r)$ is the closest eigenvalue to $\lambda_{j^*}(\widehat{\boldsymbol{A}}_r)$. If i^* and j^* are not unique, break the tie in some fashion. Therefore, we have the series of inequalities

$$\rho(\widehat{\boldsymbol{A}}_r) \leq |\lambda_{j^*}(\widehat{\boldsymbol{A}}_r) - \lambda_{i^*}(\boldsymbol{A}_r)| + |\lambda_{i^*}(\boldsymbol{A}_r)| \leq \mathsf{sv}\left(\Psi(\widehat{\boldsymbol{A}}_r), \Psi(\boldsymbol{A}_r)\right) + \rho(\boldsymbol{A}_r).$$

By Theorem 4.2, the approximate discrete dynamical system is stable if

$$\kappa_2(\boldsymbol{W})\eta\left(1 + \frac{\sqrt{2}\|\boldsymbol{\Upsilon}_f^\dagger\|_2}{1-\eta}\right) < 1 - \rho(\boldsymbol{A}_r).$$

By assumption, $\eta < 1$; the above equation gives an additional condition on η for stability. This condition can be informative for the numerical method used to compute the singular vectors $\widehat{\boldsymbol{U}}_r$.

5. Numerical results. To test our algorithms, we consider two test problems which pose different challenges. The first test problem is generated from an LTI model of a controlled heat transfer process for optimal cooling of a steel profile [18]. The second test problem is generated from an LTI state-space model of an electrical power generation system with 50 generators [2]. The dimensions of the variables related to both test problems are shown in Table 5.1. The heat transfer test problem has a relatively large system size n but a small number of inputs and outputs, while the power system test problem has a relatively small system size but a much larger number of inputs and outputs. All our experiments were run in MATLAB, and our timing experiments were run on the NCSU HPC Cluster with 72 GB memory. MATLAB code for the algorithms, test models, and some of the numerical experiments included in this section can be found at https://github.com/rlminste/RandSysID.

5.1. Heat transfer. We first test our algorithms on the heat transfer test problem, which has a large system size but a relatively small number of inputs and outputs. This system also has slow dynamics, so a large value of s is needed to capture the transient behavior. The model for this test problem is not in standard form, so we will need to convert it to standard form before we are able to use our algorithms. The original continuous form after spatial discretization is

$$\begin{aligned}\boldsymbol{E}\frac{d\boldsymbol{x}}{dt} &= \boldsymbol{A}_o\boldsymbol{x} + \boldsymbol{B}_o\boldsymbol{u},\\ \boldsymbol{y} &= \boldsymbol{C}_o\boldsymbol{x} + \boldsymbol{D}_o\boldsymbol{u}.\end{aligned}$$

TABLE 5.1

Details of the test problems we will use in our numerical experiments, as well as the size of the largest Hankel matrix $\boldsymbol{\mathcal{H}}_s$ that occurs for these test problems.

System	System size n	Inputs m	Outputs ℓ	Largest $\boldsymbol{\mathcal{H}}_s$ dim.
Heat transfer	1357	7	6	24,000 × 28,000
Power system	155	50	155	155,000 × 50,000

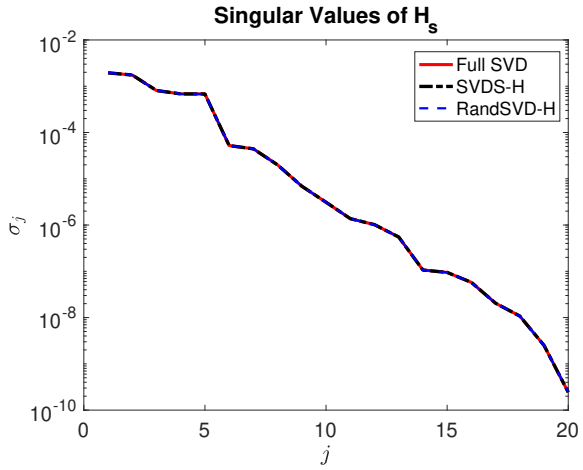

FIG. 5.1. *Singular values of the block Hankel matrix* $\boldsymbol{\mathcal{H}}_s$ *as computed by a full SVD, SVDS-H, and RandSVD-H. We computed the first* 20 *singular values using SVDS-H and RandSVD-H and plotted them against the first* 20 *singular values computed by the full SVD.*

To convert it to standard form, we use the Cholesky factorization of $\boldsymbol{E} = \boldsymbol{L}\boldsymbol{L}^\top$. Then, the continuous standard form system matrices are found by computing $\boldsymbol{A}_c = \boldsymbol{L}^{-1}\boldsymbol{A}_o\boldsymbol{L}^{-\top}$, $\boldsymbol{B}_c = \boldsymbol{L}^{-1}\boldsymbol{B}_o$, and $\boldsymbol{C}_c = \boldsymbol{C}_o\boldsymbol{L}^{-\top}$. Note that $\boldsymbol{D}_c = \boldsymbol{D}_o$ remains unchanged. The continuous-time matrices are converted into appropriate discrete-time matrices using the MATLAB `c2d` command. It is worth noting that the continuous-time matrices are only used in the construction of the Markov parameters but are not used explicitly in the system identification algorithms.

For this heat transfer problem, we will compare accuracy and speed of our algorithms that take advantage of the block Hankel structure of $\boldsymbol{\mathcal{H}}_s$, i.e., RandSVD-H (see section 3.1), to the standard ERA that uses a full SVD. We also contrast these two algorithms with another algorithm we call SVDS-H, which uses the MATLAB command `svds`, but we instead use the block Hankel multiplication described in Algorithm 3.1.

5.1.1. Accuracy. We consider several numerical experiments to test the accuracy of our algorithms. For each experiment, we used $s = 1000$ and a target rank of $r = 20$; that is, we are performing model reduction in addition to the system identification. For RandSVD-H, we used $q = 1$ for the number of subspace iterations and $\rho = 20$ for the oversampling parameter. In our first experiment, we compare the first 20 singular values of the block Hankel matrix $\boldsymbol{\mathcal{H}}_s \in \mathbb{R}^{6000\times 7000}$ as computed by a full SVD to the first 20 singular values computed by the SVDS-H and RandSVD-H algorithms. The singular values are plotted in Figure 5.1; we see that they are in good agreement with each other.

Next we compare the eigenvalues of the identified $\widehat{\boldsymbol{A}}_r$ matrices using the SVDS-H and RandSVD-H algorithms to the eigenvalues of $\boldsymbol{A}_r$ identified using the full SVD ERA algorithm. The results are shown in Figure 5.2, where we can see that the eigenvalues of $\widehat{\boldsymbol{A}}_r$ computed using SVDS-H and RandSVD-H align well with those of $\boldsymbol{A}_r$ computed using the full SVD accurately.

In order to quantitatively evaluate the accuracy of the identified eigenvalues, we consider the Hausdorff distance metric, defined between the spectra of the two

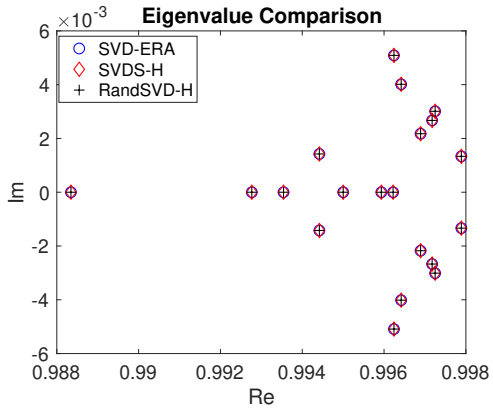

FIG. 5.2. *Eigenvalues of the identified $\boldsymbol{A}_r$ matrix using SVD-ERA, SVDS-H, and RandSVD-H algorithms on the heat transfer problem. We used $s = 1000$ and a target system size of $r = 20$ as inputs for each algorithm.*

matrices $\boldsymbol{A}_r$ and $\widehat{\boldsymbol{A}}_r$,

$$h_d(\psi(\boldsymbol{A}_r), \psi(\widehat{\boldsymbol{A}}_r)) = \max\left\{\mathsf{sv}\left(\psi(\boldsymbol{A}_r), \psi(\widehat{\boldsymbol{A}}_r)\right), \mathsf{sv}\left(\psi(\widehat{\boldsymbol{A}}_r), \psi(\boldsymbol{A}_r)\right)\right\}. \tag{5.1}$$

Note that the Hausdorff distance is related to the spectral variation, defined in (4.4), as it measures how close two sets are to each other. We use Hausdorff distance h_d instead of spectral variation since the spectral variation of two sets depends on the order of the sets in question, while the Hausdorff distance does not. For both algorithms SVDS-H and RandSVD-H, we computed the Hausdorff distance between the spectra of $\widehat{\boldsymbol{A}}_r$ and $\boldsymbol{A}_r$ and we saw that for both algorithms $h_d \approx 10^{-14}$.

The final measure of accuracy we will consider for this test problem is the relative error in the Markov parameters. Let $\boldsymbol{A}_r, \boldsymbol{B}_r, \boldsymbol{C}_r$ be the system matrices identified by the full SVD-ERA, and let $\widehat{\boldsymbol{A}}_r, \widehat{\boldsymbol{B}}_r, \widehat{\boldsymbol{C}}_r$ be the system matrices identified using another method. The relative error in the Markov parameters is then defined as

$$M_k = \frac{\|\boldsymbol{C}_r \boldsymbol{A}_r^k \boldsymbol{B}_r - \widehat{\boldsymbol{C}}_r \widehat{\boldsymbol{A}}_r^k \widehat{\boldsymbol{B}}_r\|_2}{\|\boldsymbol{C}_r \boldsymbol{A}_r^k \boldsymbol{B}_r\|_2}, \quad k = 1, \ldots, 2s-1. \tag{5.2}$$

We compute this error for the identified matrices using the SVDS-H and RandSVD-H algorithms, and plot the results in Figure 5.3. Both algorithms produce comparable error in the Markov parameters, and this error is very low in both cases. These results combined with those from Figure 5.2 show that our algorithms are very accurate compared to the standard algorithms.

5.1.2. Timing. We now consider the average run time of the three algorithms on the heat transfer problem. The results, shown in Figure 5.4, clearly indicate that using the block Hankel structure as in SVDS-H and RandSVD-H significantly reduces the computational cost, and using a randomized algorithm reduces that cost further, as RandSVD-H is computationally the least expensive of the three. We also see from the plots that the SVDS and RandSVD-H algorithms, which both exploit the block Hankel structure, show approximately linear scaling with s confirming the analysis of the computational costs.

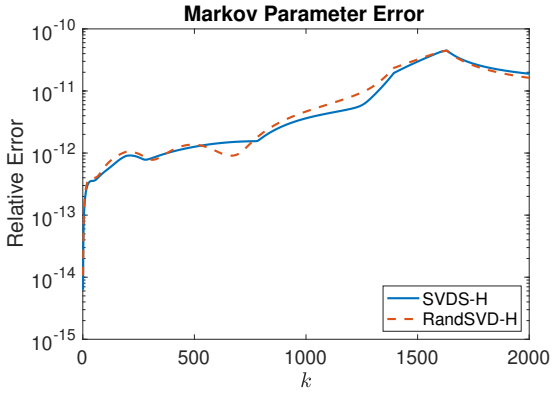

FIG. 5.3. *Relative error in recreated Markov parameters compared to a full SVD ERA for both the SVDS-H and RandSVD-H algorithms. We used* $s = 1000$ *and a target rank of* $r = 20$ *as inputs.*

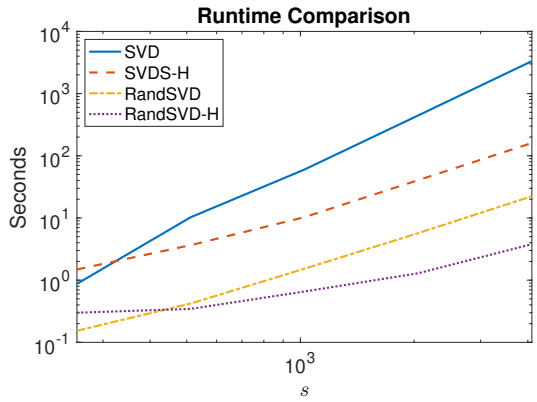

FIG. 5.4. *Average run time of the SVD, SVDS, RandSVD, and RandSVD-H algorithms on the heat transfer problem. We averaged the run time over three runs, and a target system size of* $r = 20$ *as inputs for each algorithm.*

5.1.3. Comparison to CUR-ERA. The final test we consider with this application is how CUR-ERA compares to the randomized algorithms we analyzed in the previous experiments. For CUR-ERA, we used the default parameters suggested in the code provided by the authors of [11]. The maxvol tolerance is 10^{-4}, and the iterations stop when the relative error of the subsequent iterations is less than 10^{-4}. For our experiments, we start with the accuracy of the eigenvalues, plotted in Figure 5.5. In this figure, we see that the eigenvalues of $\boldsymbol{A}_r$ identified by CUR-ERA are not as accurate as those identified by RandSVD-H. To quantify this, we compute the Hausdorff distance between the spectrum of the $\boldsymbol{A}_r$ matrix identified by CUR-ERA and the spectrum of the $\widehat{\boldsymbol{A}}_r$ matrix identified by SVD-ERA when $s = 1000$. We find that this distance is approximately 4.0×10^{-4}, supporting the claim that CUR-ERA is less accurate than RandSVD-H (we experimented with several different choices of CUR-ERA parameters and are reporting the choice which gave the smallest error). At the same time, the CUR-ERA algorithm took 0.83 seconds compared to 0.66 seconds for RandSVD-H (averaged over three runs). Finally, we show the relative error in the Markov parameters, computed as defined in (5.2), for CUR-ERA, RandSVD-H, and

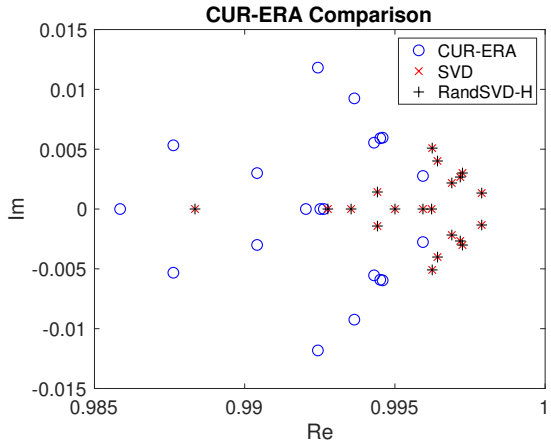

FIG. 5.5. *Eigenvalues of the identified $\boldsymbol{A}_r$ matrix using CUR-ERA, compared to those identified by SVD and RandSVD-H on the heat transfer problem with $s = 1000$ and a target system size of $r = 20$.*

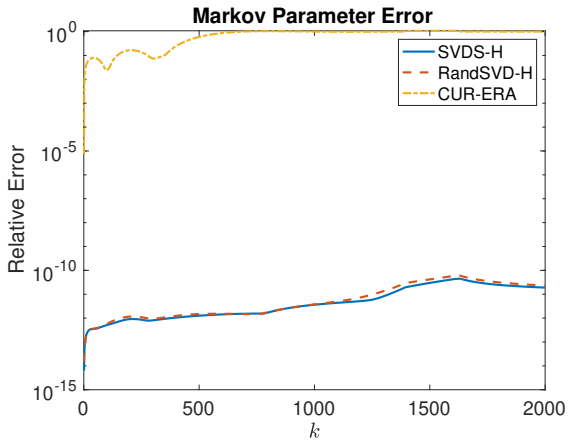

FIG. 5.6. *Relative error in recreated Markov parameters compared to a full SVD-ERA for RandSVD-H, SVDS-H, and CUR-ERA.*

SVDS-H. The results are shown in Figure 5.6, and we see that the error shown here confirms the accuracy comparison, thus far. The accuracy of CUR-ERA is lower than that obtained from either RandSVD-H or SVDS-H. In all cases, CUR-ERA took two iterations for the cross approximation to converge, and a maximum of 10 iterations to find the dominant volume submatrix, i.e., $\kappa = 2$ and $c = 10$.

5.1.4. Comparing randomized SVD algorithms. We now compare our algorithms with more recent randomized SVD algorithms available in the literature. Specifically, we considered Algorithms 4 and 5 from RSVDPACK [24], which are similar to Algorithm 2.2 but differ in how they postprocess the low-rank approximation to obtain the approximate SVD. We adapted these algorithms to incorporate block Hankel multiplication, and we denote the resulting eigenvalue decomposition based algorithm as RSVDeig-H [24, Algorithm 4], and the resulting QR decomposition based algorithm as RSVDqr-H [24, Algorithm 5]. We use the same parameters

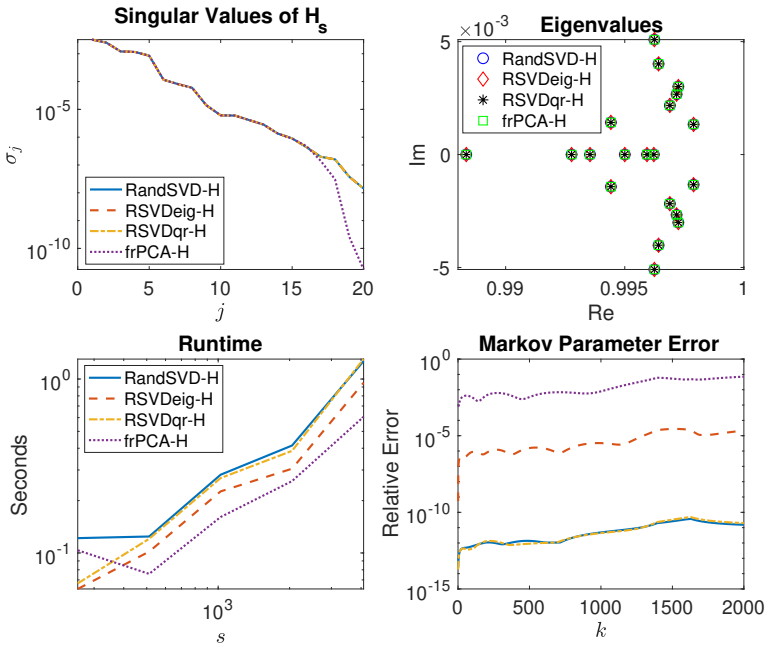

FIG. 5.7. *Experiments with the heat transfer test problem comparing RandSVD-H to RSVDeig-H, RSVDqr-H, and frPCA-H. The top left figure shows the singular values of $\boldsymbol{\mathcal{H}}_s$ computed by all four algorithms, the top right figure shows the identified eigenvalues of $\boldsymbol{A}_r$ by using the randomized algorithms, the bottom left figure shows the average run time in seconds of ERA using the four randomized algorithms, and the bottom right figure shows the relative error in the Markov parameters computed using the identified system matrices.*

for RSVDeig-H and RSVDqr-H as we do for RandSVD-H. We also compared our approaches to another recent randomized SVD algorithm called frPCA [7]. Specifically, we use frPCAt [7, Algorithm 6] with the pass parameter $q = 4$. After adapting to incorporate block Hankel multiplication we denote the resulting algorithm frPCA-H. First, we compare the singular values of $\boldsymbol{\mathcal{H}}_s$, the identified system matrix $\boldsymbol{A}_r$ using RandSVD-H, and the three randomized algorithms RSVDeig-H, RSVDqr-H, and frPCA-H. These results are shown in Figure 5.7. We observe from the plot that all the algorithms produce similar results, both in terms of the singular values of $\boldsymbol{\mathcal{H}}_s$ and the eigenvalues of $\boldsymbol{A}_r$. We then compare the average runtime using all the randomized algorithms. We observe that all four algorithms are competitive but frPCA-H is the fastest, and for smaller s values, the other two algorithms are also faster than RandSVD-H. Finally, we consider the error in the Markov parameters for all four randomized algorithms. We can see that RandSVD-H and RSVDqr-H perform accurately here, while RSVDeig-H and frPCA-H have less accuracy. Therefore, since the randomized SVD algorithm as described in Algorithm 2.2 is accurate, numerically stable, and competitive in runtime with these variants, we continue to use it in the rest of our experiments, but we note that a minor improvement in timing could be made by using these variants.

5.2. Power system. For our next application, we consider the system identification of an electric power system model. Identification is an important component of power system modeling as the operating points in a power grid keep changing due

to changes in loads, renewable generation profiles, and the network topology. All of this motivates the need for fast algorithms that can quickly update the small-signal model of the grid about these changing operating points from time to time [5]. The updated models, in turn, enable operators to make real-time control decisions for ensuring system stability. For the test problem, we consider a prototype model of reasonably large dimension, namely, the IEEE 50-generator model with 145 buses, $n = 155$ state variables, and $m = 50$ input variables, as listed in Table 5.1. Details about the physical meaning of the states and model parameters of this test system can be found in [6]. The original IEEE 50-generator model is nonlinear. The model is, therefore, first linearized about a chosen power flow solution. The resulting LTI model is excited by 50 impulse functions injected through the excitation input terminals of the generators. All $\ell = 155$ states are measured, and are considered as outputs. The output matrix for this example is, therefore, the identity matrix.

An important feature of this problem is that it has a significantly larger number of inputs and outputs with a relatively small system size $n = 155$. Since the computational cost of RandSVD-H scales linearly with the product of the number of inputs and outputs, this suggests that RandTERA may be computationally advantageous if the number of inputs and outputs can be reduced. Therefore, our experiments will compare RandTERA to RandSVD-H both in terms of accuracy and computational costs. For this problem, we again perform system identification as well as model reduction, taking the target rank to be $r = 75$. We found that model reduction (by using a truncated rank) is necessary for this power system test problem to achieve reasonable accuracy. If no model reduction is performed or if the target rank is too large, we encounter numerical instability in the eigenvalue reconstructions both in SVD-ERA and RandSVD-H.

5.2.1. Accuracy. First, we will examine the accuracy of both RandTERA and RandSVD-H on the power system test problem. For all the experiments for testing accuracy, we will use $s = 500$, and target rank $r = 75$. For the randomized algorithms, we use an oversampling parameter $\rho = 20$, and $q = 1$ subspace iterations. Note that the full Hankel matrix for the accuracy experiments is of size 77500×25000.

Choice of ℓ' and m'. In our first experiment, we will investigate the choice of the reduced number of inputs and outputs ℓ' and m' for the tangential algorithms (TERA and RandTERA). To choose these parameters, we compute the singular values of $\boldsymbol{\mathcal{H}}_w$ and $\boldsymbol{\mathcal{H}}_e$ defined in (3.1), (3.2). The indices ℓ' and m' are chosen to be the largest integers such that

$$\ell' = \arg\min_k \left\{ 1 \le k \le \ell \,|\, \sigma_k(\boldsymbol{\mathcal{H}}_w) \ge \epsilon \sigma_1(\boldsymbol{\mathcal{H}}_w) \right\},$$

$$m' = \arg\min_k \left\{ 1 \le k \le m \,|\, \sigma_k(\boldsymbol{\mathcal{H}}_e) \le \epsilon \sigma_1(\boldsymbol{\mathcal{H}}_e) \right\}.$$

That is, they are chosen to be the smallest indices for which the singular values are within a threshold ϵ of the largest singular value. The values of ℓ' and m' for different ϵ thresholds are shown in Table 5.2.

Accuracy of eigenvalues. We first compare the accuracy of the singular values of the computed block Hankel matrix $\boldsymbol{\mathcal{H}}_s$. First, we show in Figure 5.8 the effect of different ϵ values on the accuracy of singular values computed by TERA and RandTERA. We can see that the smaller the tolerance ϵ, the more accurate the singular values. Also, the singular values computed using TERA and RandTERA are less accurate compared to RandSVD-H, but the singular values computed by RandTERA are com-

TABLE 5.2
Reduced number of inputs m' and outputs ℓ' based on the singular value threshold ϵ for use in TERA and RandTERA.

ϵ	m'	ℓ'
0.1	14	18
0.05	21	33
0.01	30	59

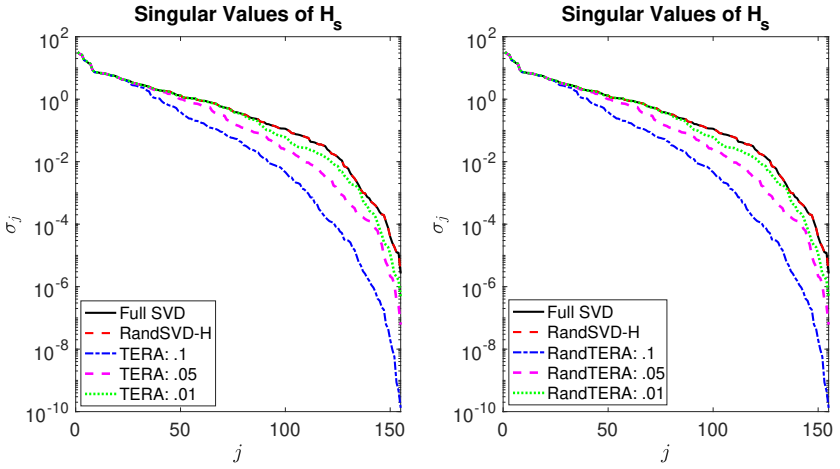

FIG. 5.8. *Singular values of block Hankel matrix $\mathcal{H}_s$ as computed by a full SVD and RandSVD-H versus those computed by (left) TERA with varying ϵ tolerances and (right) RandTERA with varying ϵ tolerances.*

parable with those of TERA. This shows that the randomized algorithms are accurate compared to their deterministic counterparts.

Next, we consider the eigenvalues of the identified $\widehat{\boldsymbol{A}}_r$ matrix using the RandSVD-H and RandTERA algorithms compared with the eigenvalues of the original $\boldsymbol{A}_r$ matrix. We use the parameters listed above, but we fix $\epsilon = 10^{-2}$. These results are shown in Figure 5.9. We can see that our algorithms identify the state matrix with eigenvalues very close to the true eigenvalues. To quantify how close they are, we consider the Hausdorff distance again as defined in (5.1). We computed the Hausdorff distance between the spectrum of the $\boldsymbol{A}_r$ matrix identified by an SVD-ERA and the spectra of the $\boldsymbol{A}_r$ matrices identified by RandSVD-H and RandTERA. We used $s = 500$ for all algorithms, and saw that for RandSVD-H, $h_d \approx 2.7 \times 10^{-4}$. For RandTERA, $h_d \approx 3.0 \times 10^{-3}$.

Our final measure of accuracy is the relative error in the Markov parameters as defined in (5.2). We plot this relative error in Figure 5.10, and make two observations. The first is that RandSVD-H has a much lower relative error than either TERA or RandTERA. The second is that the error from RandTERA is very accurate compared to the error from TERA, so the randomization is not causing a significant loss in the accuracy. We can see in Figure 5.10, that the relative error increases with increasing k. This is because the identification techniques using impulse response fail to identify the well damped poles (i.e., the poles whose real part is significantly lesser than 1) accurately, since information corresponding to these modes have to be obtained from the first few Markov parameters only. In Figure 5.11, we compare the impulse response of the original full, discrete-time model with that of the models identified

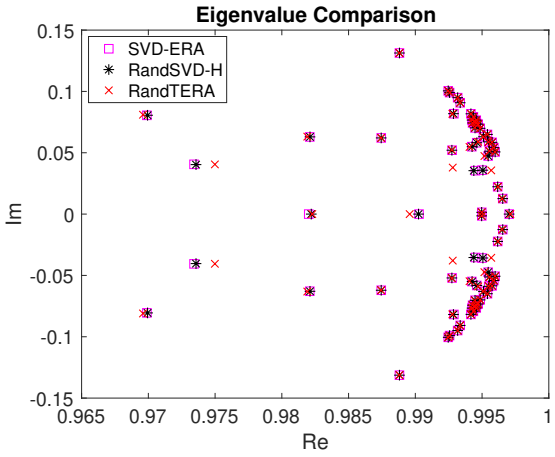

FIG. 5.9. *Eigenvalues of the identified matrix $\widehat{\boldsymbol{A}}_r$ from SVD-ERA compared to the identified matrix $\widehat{\boldsymbol{A}}_r$ from RandSVD-H and RandTERA on the power system test problem with target rank $r = 75$.*

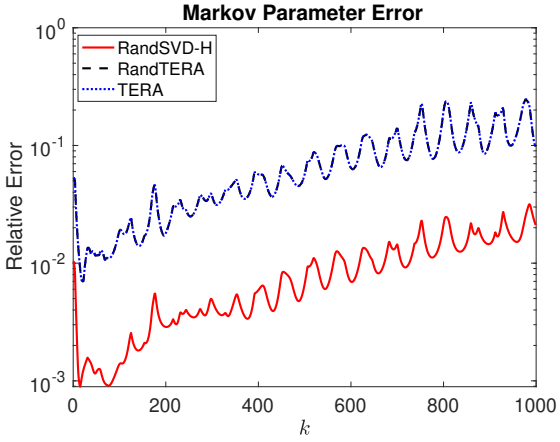

FIG. 5.10. *Relative error in the reconstructed Markov parameters using RandTERA, TERA, and RandSVD-H algorithms. The relative error is computed using the system identified using the SVD-ERA.*

using RandSVD-H and RandTERA (an impulse input was fed into Generator 1). The top plot shows the speed response of Generator 1 while the bottom plot shows that for Generator 2. We see that both the identified models have a similar performance for Generator 1 and are close to the original trajectory, whereas in Generator 2, the trajectories from all algorithms match each other closely, with only minor deviations from the "true" trajectory in the the first 2 seconds. The results for the other generators were observed to follow similar matching trends.

5.2.2. Timing. We next examine the computational cost of RandSVD-H and RandTERA on the power system test problem. We show in Table 5.3 the average CPU time of both of these algorithms compared to a full SVD as s increases. As before, we averaged over three different runs, and used the same parameters for the randomized algorithms, $r = 75$, $\rho = 20$, and $q = 1$. We see that RandTERA is much

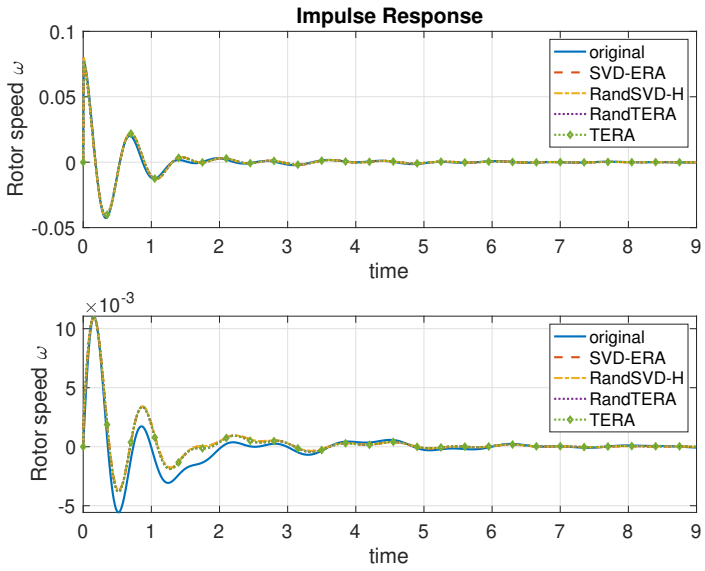

FIG. 5.11. *Impulse response of the original discrete-time system compared against that for the systems identified using SVD-ERA, RandSVD-H, TERA, and RandTERA and excited with impulse input at Generator* 1*, with responses from Generator* 1 *(top) and Generator* 2 *(bottom).*

TABLE 5.3
Computational time in seconds of the SVD step in the identification algorithms SVD-ERA, RandSVD-H, TERA, and RandTERA. Note for $s = 1000$*, the matrix was too large to store explicitly, so we only show the run times for the other algorithms.*

s	Size of $\boldsymbol{\mathcal{H}}_s$	SVD	RandSVD-H	TERA	RandTERA
500	$77{,}500 \times 25{,}000$	1634.4	20.3	339.7	6.6
700	$108{,}500 \times 35{,}000$	4292.0	28.4	795.2	11.7
1000	$155{,}000 \times 50{,}000$	N/A	41.9	1983.9	18.0

cheaper compared to RandSVD-H, and both are much cheaper than a full SVD as expected. Note that for $s = 1000$, the matrix size was $155{,}000 \times 50{,}000$ and since this matrix was too large to store explicitly, we do not report the computational cost using the SVD.

6. Conclusions and future work. We have presented two different randomized algorithms for accelerating the computational cost of system identification using ERA. The first algorithm accelerates the randomized subspace iteration by efficiently computing matrix vector products using the block Hankel structure. The second algorithm uses the previous algorithm, but on a matrix with a reduced number of inputs and outputs using tangential interpolation. The algorithms no longer have the cubic scaling with s, which controls the number of time steps. The error analysis relates the accuracy of the eigenvalues of the identified system matrix to the accuracy of the singular vectors of the block Hankel matrix $\boldsymbol{\mathcal{H}}_s$. Numerical experiments from different applications show that our algorithms are both accurate and computationally efficient.

There are several avenues for future work. First, the analysis in section 4 is general and does not make assumptions on the specific algorithm used to compute the low-

rank approximation to $\boldsymbol{\mathcal{H}}_s$. The analysis can be specialized by using specific results for the accuracy of the singular vectors, for example, following the approaches in [17, 19, 20]. Second, the paper assumes that we have access to the Markov parameters. This is not the case when it is possible to excite the system using general inputs. One possibility, as mentioned earlier, is to use the approach in [10] which may not be feasible when a large number of Markov parameters need to be estimated. Algorithms such as MOESP and N4SID (see [23, Chapter 9]) can be used for general inputs but also suffer from high computational costs. However, we can combine the block Hankel structure with randomized algorithms for the general input case as well. Such techniques which involve identification using general input can be extremely accurate in identifying the well-damped poles as they constantly excite the system with a persistently exciting signal. This is an ongoing research direction.

Acknowledgment. We would like to thank Eric Hallman for useful comments and for generously sharing his code with us.

REFERENCES

[1] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, Adv. Des. Control 6, SIAM, Philadelphia, 2005.
[2] P. Benner and J. Saak, *Linear-quadratic regulator design for optimal cooling of steel profiles*, Technical report SFB393/05-05, Sonderforschungsbereich, TU Chemnitz, Chemnitz, Germany, 2005, http://nbn-resolving.de/urn:nbn:de:swb:ch1-200601597.
[3] R. Bhatia, *Matrix Analysis*, Grad. Texts in Math. 169, Springer, New York, 2013.
[4] Å. Björck, *Numerical Methods in Matrix Computations*, Texts Appl. Math. 59, Springer, Cham, Switzerland, 2015.
[5] G. Chavan, M. Weiss, A. Chakrabortty, S. Bhattacharya, A. Salazar, and F. Ashrafi, *Identification and predictive analysis of a multi-area WECC power system model using synchrophasors*, IEEE Trans. Smart Grid, 8 (2017), pp. 1977–1986.
[6] J. H. Chow and K. W. Cheung, *A toolbox for power system dynamics and control engineering education and research*, IEEE Trans. Power Systems, 7 (1992), pp. 1559–1564.
[7] X. Feng, Y. Xie, M. Song, W. Yu, and J. Tang, *Fast randomized PCA for sparse data*, in Asian Conference on Machine Learning, Proc. Mach. Learn. Res. (PMLR), 95 (2018), pp. 710–725.
[8] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.
[9] B. Hunter and T. Strohmer, *Performance Analysis of Spectral Clustering on Compressed, Incomplete and Inaccurate Measurements*, preprint, arXiv:1011.0997, 2010.
[10] J.-N. Juang, M. Phan, L. G. Horta, and R. W. Longman, *Identification of observer/Kalman filter Markov parameters-theory and experiments*, J. Guid. Control Dyn., 16 (1993), pp. 320–329.
[11] B. Kramer and A. A. Gorodetsky, *System identification via CUR-factored Hankel approximation*, SIAM J. Sci. Comput., 40 (2018), pp. A848–A866.
[12] B. Kramer and S. Gugercin, *Tangential interpolation-based eigensystem realization algorithm for MIMO systems*, Math. Comput. Model. Dyn. Syst., 22 (2016), pp. 282–306.
[13] S.-Y. Kung, *A new identification and model reduction algorithm via singular value decomposition*, in Proceedings of the 12th Asilomar Conferemce on Circuits, Systems and Computers, Pacific Grove, CA, IEEE, New York, 1978, pp. 705–714.
[14] L. Ljung, *System Identification: Theory for the User*, Prentice Hall, Upper Saddle River, NJ, 1999.
[15] P.-G. Martinsson and J. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), pp. 403–572.
[16] P.-G. Martinsson and S. Voronin, *A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices*, SIAM J. Sci. Comput., 38 (2016), pp. S485–S507.
[17] Y. Nakatsukasa, *Accuracy of singular vectors obtained by projection-based SVD methods*, BIT, 57 (2017), pp. 1137–1152.

[18] Oberwolfach Benchmark Collection, *Steel Profile*, http://modelreduction.org/index.php/Steel_Profile, 2005.
[19] A. K. Saibaba, *Randomized subspace iteration: Analysis of canonical angles and unitarily invariant norms*, SIAM J. Matrix Anal. Appl., 40 (2019), pp. 23–48.
[20] H. D. Simon and H. Zha, *Low-rank matrix approximation using the Lanczos bidiagonalization process with applications*, SIAM J. Sci. Comput., 21 (2000), pp. 2257–2274.
[21] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Comput. Sci. Sci. Comput., Academic Press, Boston, 1990.
[22] P. Van Overschee and B. De Moor, *Subspace Identification for Linear Systems: Theory—Implementation—Applications*, Springer, Boston, 1996.
[23] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach*, Cambridge University Press, Cambridge, 2007.
[24] S. Voronin and P.-G. Martinsson, *RSVDPACK: An Implementation of Randomized Algorithms for Computing the Singular Value, Interpolative, and CUR Decompositions of Matrices on Multi-core and GPU Architectures*, preprint, arXiv:1502.05366 (2015).
[25] T. Wu, V. M. Venkatasubramanian, and A. Pothen, *Fast parallel stochastic subspace algorithms for large-scale ambient oscillation monitoring*, IEEE Trans. Smart Grid, 8 (2016), pp. 1494–1503.
[26] D. Yu and S. Chakravorty, *A computationally optimal randomized proper orthogonal decomposition technique*, in 2016 American Control Conference (ACC), IEEE, Piscataway, NJ, 2016, pp. 3310–3315.
[27] W. Yu, Y. Gu, and Y. Li, *Efficient randomized algorithms for the fixed-precision low-rank matrix approximation*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 1339–1359.