SIAM J. SCI. COMPUT.
Vol. 40, No. 5, pp. A3267–A3292

HIERARCHICAL APPROXIMATE PROPER ORTHOGONAL DECOMPOSITION*

CHRISTIAN HIMPE†, TOBIAS LEIBNER‡, AND STEPHAN RAVE‡

Abstract. Proper Orthogonal Decomposition (POD) is a widely used technique for the construction of low-dimensional approximation spaces from high-dimensional input data. For large-scale applications and an increasing number of input data vectors, however, computing the POD often becomes prohibitively expensive. This work presents a general, easy-to-implement approach to compute an approximate POD based on arbitrary tree hierarchies of worker nodes, where each worker computes a POD of only a small number of input vectors. The tree hierarchy can be freely adapted to optimally suit the available computational resources. In particular, this hierarchical approximate POD (HAPOD) allows for both simple parallelization with low communication overhead, as well as incremental POD computation under constrained memory capacities. Rigorous error estimates ensure the reliability of our approach, and extensive numerical examples underline its performance.

Key words. model reduction, proper orthogonal decomposition, singular value decomposition, parallel algorithms, distributed algorithms

AMS subject classifications. 65Y99, 65M22, 65F99, 68W10, 68W15

DOI. 10.1137/16M1085413

1. Introduction.

The construction of low-dimensional subspaces from high-dimensional data, dynamics, or operators is an essential mechanism in many applications, with the aim to accelerate or merely enable numerical computations of large-scale models. In the discipline of model reduction, this methodology is the central problem under investigation.

A well-known and popular approach for subspace construction is the Proper Orthogonal Decomposition (POD), i.e., the computation of the left-singular vectors associated with the dominant singular values of a given set of input column vectors concatenated to a matrix. An important field of application for the POD is the reduction of ordinary differential equation (ODE) models [35] and partial differential equation (PDE) models [27, 28]. A landmark work in this context is the use of the POD for compression of simulation data [46] where the dominant modes are extracted from flow simulation time series by the *method of snapshots*. For an elaborate review of the POD method, see, for example, [16, 23].

Due to technical limitations of computational resources, such as memory-space and acceptable computational complexities, not only the evaluation of a large-scale

*Submitted to the journal's Methods and Algorithms for Scientific Computing section July 18, 2016; accepted for publication (in revised form) July 6, 2018; published electronically October 4, 2018.

http://www.siam.org/journals/sisc/40-5/M108541.html

Funding: This work was supported by the Deutsche Forschungsgemeinschaft, DFG EXC 1003, Cells in Motion (CiM) Cluster of Excellence, Münster, Germany, by the Center for Developing Mathematics in Interaction, DEMAIN, Münster, Germany, by Cells in Motion (CiM) Cluster of Excellence in flexible funds project FF-2015-07, by the German Federal Ministry of Education and Research (BMBF) under contract 05M13PMA, and by the German Federal Ministry for Economic Affairs and Energy (BMWi) in the joint project "MathEnergy – Mathematical Key Technologies for Evolving Energy Grids," sub-project Model Order Reduction (grant 0324019**B**).

†Computational Methods in Systems and Control Theory Group at the Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, D-39106 Magdeburg, Germany (himpe@mpi-magdeburg.mpg.de).

‡Applied Mathematics, University of Münster, Einsteinstrasse 62, D-48149 Münster, Germany (tobias.leibner@uni-muenster.de, stephan.rave@uni-muenster.de).

problem, but even the computation of a low-rank approximation by existing methods may be infeasible. This is particularly true for the POD, as the (truncated) singular value decomposition (SVD) of large matrices is a computationally demanding task. In order to speed up the computation, various parallel algorithms are available for SVD computation [5]; more recently, partitioning approaches were developed to obtain the SVD, or an approximation thereof, such as [47, 48], [11, 12], [4], and [50], as well as a related parallel QR decomposition in [43]. A commonality of these methods is the horizontal slicing of the argument matrix, which is similar to the partitioning of the spatial domain of a discretized PDE model. However, such an approach is only possible when complete horizontal slices of the argument matrix are available. This usually means that all input data vectors have to be computed and stored before starting the POD computation. For large problems, this might be impossible due to insufficient memory or even mass storage space. Also, for parametrized problems the input data might be distributed columnwise among several workers, and horizontal slicing of the input would require heavy communication between the workers, which may be impossible, as in, for instance, grid-computing environments.

In comparison, the herein proposed Hierarchical Approximate Proper Orthogonal Decomposition (HAPOD) is based on a vertical slicing of the input matrix and is targeted to extend POD-based methods, which were designed with "tall and skinny" matrices in mind towards settings where, due to enhanced requirements such as parametrization, the actual matrix dimension is "tall and not-so-skinny."

Our method is based upon the simple idea of replacing subsets of input vectors by POD approximations of these, which then form the input of additional POD steps. As such, our algorithm can be applied additionally to any pre-existing POD implementation. Formulated for arbitrary tree hierarchies of workers, it allows sequential and parallel decompositions, as well as combinations thereof, based on the partitioning of the time domain or parameter space.

The HAPOD is a *single pass* method in the sense that the input vectors at a given HAPOD node are only required for a single local POD computation and can be discarded afterwards. Rigorous error estimates allow a priori control of the final ℓ^2-approximation error for the input data. At the same time, bounds for the number of generated HAPOD modes guarantee quasi-optimality of the generated approximation space. As long as the final depth of the HAPOD tree is known, local PODs can be computed as soon as all input data for a given node is available. As such, the HAPOD can also be seen as a general methodology for approximation quality control when updating POD spaces with additional input data.

Stochastic methods for SVD computation, e.g., [13, 17, 22, 42], share many benefits with the HAPOD. In particular, these methods are easily parallelizable with comparable communication requirements (at least when no power iteration is performed), and single pass formulations do exist. However, most algorithms are designed for a prescribed fixed approximation rank. Those which do guarantee spaces with prescribed approximation error (see [17, section 4.4], [22], or the preprint [25]) are based on iterative procedures which require multiple passes over the input data. Also, our approach can be implemented more easily than already existing POD codes.

For the incremental (updated) computation of an SVD we refer to the work of Brand [6, 7], which allows the update of an existing SVD given new data. Geared towards (POD-based) model reduction, [37] uses Brand's algorithm for an incremental POD algorithm. In this context, the HAPOD framework provides local choices of truncation error tolerances to rigorously control the overall approximation error, given that the maximum number of updates is known. A similar updated POD algorithm

is employed for the experiment in section 4.3. In [3], another family of rank-based approaches for incremental SVD computations is presented.

Given the simplicity of the HAPOD, we do not claim to be first in investigating this concept. In fact, we recently became aware of [38], wherein special cases of the HAPOD (i.e., a distributed and an incremental HAPOD in the sense of section 3.2) are briefly discussed. Balanced n-ary tree structures are investigated in [24]. In both cases, error bounds for prescribed truncation ranks are derived. Another application of the distributed HAPOD is discussed in [8], which uses the error bound derived in [38]. In the context of principal component analysis, distributed methods have been introduced [32, 40, 41] which, apart from the centering of the data set, correspond to a distributed HAPOD. No rigorous error bounds are derived, however.

The main contribution of this work is a thorough study of the HAPOD with the aim of showing that it should be a standard part in the toolbox of every model reduction practitioner. In particular, in contrast to [38, 24, 8], we formally analyze the algorithm in a general setting with arbitrary tree topologies, making it suitable for more complex applications (cf. section 4.3), and give for prescribed local POD truncation error tolerances estimates for both the approximation error as well as the obtained (local and final) numbers of POD modes. Based on these estimates we provide rules for the selection of the local error tolerances to achieve a given global target (mean) approximation error, with a user-definable tradeoff between optimality of the generated approximation space and computational efficiency. We show the performance of our method for input data with quickly decaying singular values, as it is typically the case in model reduction applications (cf. Remarks 11 and 12 and section 3.4). Section 4 contains extensive numerical experiments highlighting the applicability of our method.

Before introducing the HAPOD in section 3, we start with a concise summary of the POD and its properties in section 2.

2. Proper orthogonal decomposition. Proper Orthogonal Decomposition is a technique for finding low-order approximation spaces for a given set of snapshot (data) vectors by computing the left-singular vectors corresponding with the dominant singular values of the matrix formed by the columnwise concatenation of the snapshot vectors. Designations used in other fields are *Principal Component Analysis*, *Empirical Eigenfunctions*, *Empirical Orthogonal Functions,* or *Karhunen–Loève Decomposition*. A more formal definition of the POD, which also applies to infinite-dimensional spaces, is given as follows.

DEFINITION 1 (proper orthogonal decomposition (POD)). *Let $\mathcal{S}$ be a finite multiset of vectors contained in a Hilbert space V and denote by $|\mathcal{S}|$ its cardinality. With $e_1, \dots, e_{|\mathcal{S}|} \in \mathbb{R}^{|\mathcal{S}|}$ the canonical basis of $\mathbb{R}^{|\mathcal{S}|}$, and with $\{s_1, \dots, s_{|\mathcal{S}|}\} = \mathcal{S}$ an arbitrary enumeration of the elements of $\mathcal{S}$, we call sequences $\varphi_1, \dots, \varphi_{|\mathcal{S}|} \in V$, $\sigma_1, \dots, \sigma_{|\mathcal{S}|} \in \mathbb{R}$ POD modes and singular values of $\mathcal{S}$ if φ_m, σ_m are the left-singular vectors and singular values of the linear mapping $\underline{\mathcal{S}}$ given by*

$$\underline{\mathcal{S}} : \mathbb{R}^{|\mathcal{S}|} \to V, \quad e_m \mapsto \underline{\mathcal{S}}(e_m) := s_m, \qquad 1 \le m \le |\mathcal{S}|. \tag{1}$$

Remark 2. Due to the uniqueness properties of the SVD, the POD singular values of a given multiset $\mathcal{S}$ are uniquely defined. The POD modes are uniquely defined up to orthogonal mappings of subspaces of V spanned by modes with the same singular value.

Remark 3. A simple yet numerically robust algorithm for the computation of the SVD of $\underline{\mathcal{S}}$ is based on computing the eigenvalue decomposition of the Gramian

$G := (s_i, s_j)_{i,j}$ to the snapshot set $\mathcal{S} = \{s_1, \dots, s_{|\mathcal{S}|}\}$. The kth POD mode φ_k is then obtained as

$$\varphi_k = \frac{1}{\sqrt{\lambda_k}} \sum_{i=1}^{|\mathcal{S}|} \psi_{k,i} \cdot s_i,$$

where λ_k is the kth largest eigenvalue of G, and $\psi_{k,i}$ the ith component of the corresponding eigenvector.[1]

The basic idea of the algorithm outlined in Remark 3, which in the context of model reduction is also known as the *method of snapshots* [46], is to replace the difficult task of computing the SVD of a large snapshot matrix with the easier task of computing the eigenvalue decomposition of the much smaller (symmetric) Gramian, which can be obtained efficiently by optimized matrix-matrix multiplication algorithms.

While this approach performs well if there are relatively few snapshot vectors (i.e., "tall and skinny" snapshot matrices), it suffers from the quadratic growth in computational complexity for computing the Gramian when the number of snapshots increases. However, using this method in conjunction with the herein proposed HAPOD algorithm can drastically reduce the overall required computational effort, making it feasible even for large snapshot sets $\mathcal{S}$ (see section 3.4).

The main reason for the emphasis on the POD is the fact that it produces the best approximating spaces in the ℓ^2-sense:

THEOREM 4 (Schmidt–Eckhard–Young–Mirsky). *Let (σ_m, φ_m), $1 \le m \le |\mathcal{S}|$, be the singular values and modes of a POD of a given snapshot multiset $\mathcal{S}$. Then for each $1 \le N \le |\mathcal{S}|$, $V_N := \operatorname{span}\{\varphi_1, \dots, \varphi_N\}$ is an ℓ^2-best approximating space for $\mathcal{S}$ in the sense that*

$$\sum_{s \in \mathcal{S}} \|s - P_{V_N}(s)\|^2 = \min_{\substack{X \subseteq V \\ \dim X = N}} \sum_{s \in \mathcal{S}} \|s - P_X(s)\|^2 = \sum_{m=N+1}^{|\mathcal{S}|} \sigma_m^2, \tag{2}$$

where $\|\cdot\|$ denotes the norm on V and P_X is the V-orthogonal projection onto the linear subspace X.

The HAPOD algorithm presented in section 3 can be based on any pre-existing POD implementation. We formalize the concept of a POD algorithm as follows.

DEFINITION 5. *For a given Hilbert space V, let* POD *be the mapping*

$$(\mathcal{S}, \varepsilon) \mapsto \mathrm{POD}(\mathcal{S}, \varepsilon) := \{(\sigma_n, \varphi_n)\}_{n=1}^{N},$$

which assigns to each finite multiset $\mathcal{S} \subseteq V$ and each $\varepsilon > 0$ the set given by the first N pairs of singular values σ_n and modes ϕ_n of the POD of $\mathcal{S}$, where N is the smallest nonnegative integer such that the ℓ^2-best-approximation error is bounded by ε, i.e., $\sum_{s \in \mathcal{S}} \|s - P_{V_N}(s)\|^2 \le \varepsilon^2$. According to (2), N is thus given as

$$N = \min\left\{ N' \in \{0, \dots, |\mathcal{S}|\} \;\middle|\; \sum_{n=N'+1}^{|\mathcal{S}|} \sigma_n^2 \le \varepsilon^2 \right\}.$$

Assuming that no SVD is performed for $\varepsilon = 0$ and the original snapshot multiset is returned, we also define $\mathrm{POD}(\mathcal{S}, 0) := \{(1, s) \mid s \in \mathcal{S}\}$.

[1] Note that the condition number of G is the square of the condition number of $\mathcal{S}$, limiting the numerical accuracy of this method in comparison to other SVD algorithms.

3. Hierarchical approximate POD (HAPOD). In this section we introduce the HAPOD algorithm (section 3.1) and provide estimates that allow one to control the approximation error as well as the number of computed POD modes (section 3.3). Special cases for distributed and incremental HAPOD computation are discussed in section 3.2. A further discussion of the advantages of the HAPOD is contained in section 3.4, whereas proofs of our main theorems can be found in section 3.5. The notation used in this section is summarized in Table 1.

TABLE 1
Key notation. Additional notation required in the proofs of Theorems 8 *and* 9 *is given in Definition* 14.

$\mathcal{C}_\mathcal{T}(\alpha)$	children of node α in tree $\mathcal{T}$	$\mathcal{N}_\mathcal{T}$	node set of tree $\mathcal{T}$
D	snapshot-to-leaf map	$\mathcal{N}_\mathcal{T}(\alpha)$	nodes below α in tree $\mathcal{T}$
$\varepsilon(\alpha)$	error tolerance at node α	$\rho_\mathcal{T}$	root node of tree $\mathcal{T}$
$L_\mathcal{T}$	depth of tree $\mathcal{T}$	$\mathcal{S}$	snapshot set
$L_\mathcal{T}(\alpha)$	level of node α in tree $\mathcal{T}$	$\mathcal{S}_\alpha$	input snapshots at node α
$\mathcal{L}_\mathcal{T}$	leaf set of tree $\mathcal{T}$	$\widetilde{\mathcal{S}}_\alpha$	snapshots below α in the tree

3.1. Definition of the HAPOD. The basic idea of the HAPOD algorithm is to replace the task of computing a POD of a given large snapshot set $\mathcal{S}$ by several small PODs which only depend on small subsets of $\mathcal{S}$ and previously computed PODs. To formalize this procedure, we consider rooted trees where each node of the tree is associated with a local POD.

A rooted tree is a connected acyclic graph of which one node is designated as the root of the tree. The following equivalent definition will better suit our needs.

DEFINITION 6 (rooted tree). *For an arbitrary set X, denote by* $\mathrm{Pow}(X)$ *its power set. We then call a triple $\mathcal{T} = (\mathcal{N}_\mathcal{T}, \mathcal{C}_\mathcal{T}, \rho_\mathcal{T})$, where $\mathcal{N}_\mathcal{T}$ is a finite set, $\rho_\mathcal{T} \in \mathcal{N}_\mathcal{T}$, and $\mathcal{C}_\mathcal{T} : \mathcal{N}_\mathcal{T} \to \mathrm{Pow}(\mathcal{N}_\mathcal{T} \setminus \{\rho_\mathcal{T}\})$, a rooted tree if the mapping $\mathcal{C}_\mathcal{T}$ satisfies the following properties:*

$$\forall \alpha,\ \beta \in \mathcal{N}_\mathcal{T} : \alpha \neq \beta \Rightarrow \mathcal{C}_\mathcal{T}(\alpha) \cap \mathcal{C}_\mathcal{T}(\beta) = \emptyset, \tag{3}$$

$$\forall\, \emptyset \neq X \subseteq \mathcal{N}_\mathcal{T} \setminus \{\rho_\mathcal{T}\}\ \ \exists \alpha \in \mathcal{N}_\mathcal{T} \setminus X :\ \mathcal{C}_\mathcal{T}(\alpha) \cap X \neq \emptyset. \tag{4}$$

We call elements $\alpha \in \mathcal{N}_\mathcal{T}$ the nodes of $\mathcal{T}$ and the elements of $\mathcal{C}_\mathcal{T}(\alpha)$ the children of α. Condition (3) *states that every node of $\mathcal{T}$ is the child of at most one node, whereas condition* (4) *ensures that every node is connected to the root node $\rho_\mathcal{T}$. Together,* (3) *and* (4) *imply that there are no cycles in $\mathcal{T}$.*

The leaf set $\mathcal{L}_\mathcal{T}$ of $\mathcal{T}$ is given by

$$\mathcal{L}_\mathcal{T} := \{\alpha \in \mathcal{N}_\mathcal{T} \mid \mathcal{C}_\mathcal{T}(\alpha) = \emptyset\}.$$

For each node $\alpha \in \mathcal{N}_\mathcal{T}$ we define the nodes below α, $\mathcal{N}_\mathcal{T}(\alpha)$, recursively by the relation

$$\mathcal{N}_\mathcal{T}(\alpha) := \{\alpha\} \ \cup \bigcup_{\beta \in \mathcal{C}_\mathcal{T}(\alpha)} \mathcal{N}_\mathcal{T}(\beta).$$

Finally, we define the level map $L_\mathcal{T} : \mathcal{N}_\mathcal{T} \to \mathbb{N}$ recursively as

$$L_\mathcal{T}(\alpha) := \max(\{L_\mathcal{T}(\beta) \mid \beta \in \mathcal{C}_\mathcal{T}(\alpha)\} \cup \{0\}) + 1$$

and call $L_\mathcal{T} := L_\mathcal{T}(\rho_\mathcal{T})$ the depth of $\mathcal{T}$.

Given a tree $\mathcal{T}$, the HAPOD algorithm works by first assigning vectors of a given snapshot set $\mathcal{S}$ to the leaves of the tree. Then, starting with the leaves, a POD of the local input data is computed at each node. The resulting modes are scaled by their corresponding singular values and passed on as input to the parent node. The final HAPOD modes are collected as the output of the root node $\rho_{\mathcal{T}}$ (cf. Figures 1 and 8). The precise definition is given as follows.

DEFINITION 7 (hierarchical approximate POD (HAPOD)). *Let $\mathcal{S} \subseteq V$ be a finite multiset of snapshot vectors in a Hilbert space V. Given a rooted tree $\mathcal{T}$ and mappings*

$$D : \mathcal{S} \to \mathcal{L}_{\mathcal{T}}, \qquad \varepsilon : \mathcal{N}_{\mathcal{T}} \to \mathbb{R}^{\geq 0},$$

define recursively for each $\alpha \in \mathcal{N}_{\mathcal{T}}$

$$\mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha) := \mathrm{POD}(\mathcal{S}_\alpha, \varepsilon(\alpha)),$$

where the local input data multiset $\mathcal{S}_\alpha$ is given by

$$\mathcal{S}_\alpha := \begin{cases} D^{-1}(\{\alpha\}), & \alpha \in \mathcal{L}_{\mathcal{T}}, \\ \bigcup_{\beta \in \mathcal{C}_{\mathcal{T}}(\alpha)} \Big\{ \sigma_n \cdot \varphi_n \mid (\sigma_n, \varphi_n) \in \mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\beta) \Big\} & \text{otherwise}, \end{cases}$$

with $D^{-1}(\{\alpha\}) := \{s \in \mathcal{S} \,|\, D(s) \in \{\alpha\}\} = \{s \in \mathcal{S} \,|\, D(s) = \alpha\}$ being the multiset of all snapshot vectors assigned to the leaf node α. We call $\mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon] := \mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\rho_{\mathcal{T}})$ the HAPOD of $\mathcal{S}$ for the tree $\mathcal{T}$, the snapshot distribution D, and the local tolerances ε.

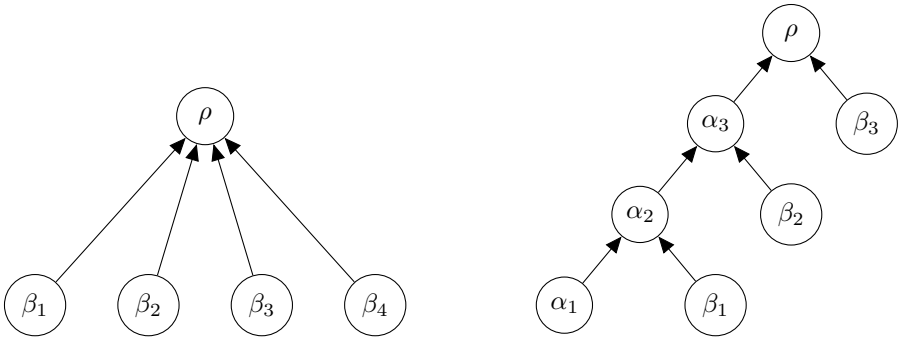
3.2. Special cases: Distributed and incremental HAPOD. The HAPOD is defined for arbitrary rooted trees, yet two classes of tree topologies present important special cases due to their ease of application. Both cases have also been discussed in [38].

One special case of the HAPOD constitutes a "flat" tree (star), in which all leaf nodes are the children of the root node, i.e., $\mathcal{C}_{\mathcal{T}}(\rho_{\mathcal{T}}) = \mathcal{N}_{\mathcal{T}} \setminus \{\rho_{\mathcal{T}}\}$, and the snapshot set $\mathcal{S}$ is distributed evenly among the leaf nodes (see Figure 1a). For such a tree the HAPOD is given as

$$\mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\rho_{\mathcal{T}}) = \mathrm{POD}\Big(\bigcup_{\beta \in \mathcal{L}_{\mathcal{T}}} \Big\{ \sigma_n \cdot \varphi_n \;\Big|\; (\sigma_n, \varphi_n) \in \mathrm{POD}(D^{-1}(\{\beta\}), \varepsilon(\beta)) \Big\}, \varepsilon(\rho_{\mathcal{T}}) \Big).$$

From a numerical linear algebra perspective this *distributed HAPOD* is closely related to the "horizontal slicing" distributed SVD methods [4, 11, 12, 43, 47, 48, 50]. The key algorithmic difference is the horizontal partitioning of the data vectors forming the columns of the snapshot matrix into fat chunks as opposed to the vertical partitioning into thin chunks of complete data vectors considered here.

A second special case of the HAPOD is a "skinny" tree (totally unbalanced binary tree). Each node of this tree is either a leaf or has exactly one leaf and one nonleaf as children (see Figure 1b). Formally, we then have $\mathcal{N}_{\mathcal{T}} = (\{\alpha_1, \ldots, \alpha_L\} \cup \{\beta_1, \ldots, \beta_{L-1}\})$, $\rho_{\mathcal{T}} = \alpha_L$, $\mathcal{C}_{\mathcal{T}}(\beta_l) = \emptyset$ for all $1 \leq l \leq L-1$, $\mathcal{C}_{\mathcal{T}}(\alpha_1) = \emptyset$, and $\mathcal{C}_{\mathcal{T}}(\alpha_l) = \{\alpha_{l-1}, \beta_{l-1}\}$ for $2 \leq l \leq L$. Typically, one will perform no additional PODs on the input data, so $\varepsilon(\beta_l) = 0$. In this case, the HAPOD is given as



(a) Distributed approximate POD. The PODs at the leaves β_i can be computed in parallel. Afterwards an additional POD is performed at the root node ρ.

(b) Incremental HAPOD. New snapshot data enters at the nodes β_i which is then combined with the current modes by PODs at the nodes α_i.

FIG. 1. *Trees corresponding to distributed and incremental HAPOD computation.*

$$\mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon](\alpha_1) = \mathrm{POD}(D^{-1}(\{\alpha_1\}),\varepsilon(\alpha_1)) \text{ and}$$

$$\begin{aligned}&\mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon](\alpha_l)\\ &= \mathrm{POD}\Big(\big\{\sigma_n\cdot\varphi_n \mid (\sigma_n,\varphi_n)\in \mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon](\alpha_{l-1})\big\}\cup D^{-1}(\{\beta_{l-1}\}),\,\varepsilon(\alpha_l)\Big)\end{aligned}$$

for $2\le l\le L$. Thus, the HAPOD can be computed incrementally by a simple iterative procedure, where in each update step a POD of the current (scaled) HAPOD modes together with the new input data is computed, whereas old input data can be removed from memory.

To accelerate the computation of this *incremental HAPOD*, an incremental SVD algorithm such as [6] might be used for the local POD computations. In this case, the main theorems in section 3.3 then provide a means to select truncation error tolerances for the individual SVD updates that guarantee final approximation spaces of prescribed quality.

3.3. Main theorems. Two central questions about the HAPOD are answered by the following theorems: Given error tolerances ε, what is the approximation error for the computed HAPOD modes (Theorem 8)? How many modes does the HAPOD produce in comparison to a direct POD computation (Theorem 9)? Only by controlling both quantities simultaneously can we arrive at an efficient approximation scheme. The proofs to the following theorems are given in section 3.5.

THEOREM 8. *Let $\mathcal{S},\mathcal{T},D,\varepsilon$ be given as in Definition 7, let the multiset of all snapshots subordinate to the node α be given by $\widetilde{\mathcal{S}}_\alpha := \bigcup_{\gamma\in\mathcal{L}_\mathcal{T}\cap\mathcal{N}_\mathcal{T}(\alpha)} D^{-1}(\{\gamma\})$, and let P_α be the V-orthogonal projection onto the linear space spanned by the modes of* $\mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon](\alpha)$. *The ℓ^2-approximation error for the HAPOD space at node α is then bounded by*

$$\sum_{s\in\widetilde{\mathcal{S}}_\alpha}\|s-P_\alpha(s)\|^2 \le \sum_{\gamma\in\mathcal{N}_\mathcal{T}(\alpha)}\varepsilon(\gamma)^2. \tag{5}$$

THEOREM 9. *With the same notation as in Theorem 8 we have for each $\alpha\in\mathcal{N}_\mathcal{T}$*

the following bound for the number of HAPOD modes:

$$\Big|\mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon](\alpha)\Big| \le \Big|\mathrm{POD}\Big(\widetilde{\mathcal{S}}_\alpha,\varepsilon(\alpha)\Big)\Big|. \tag{6}$$

In model reduction applications, the ℓ^2-*mean* approximation error is often the desired quantity to optimize for, since in many cases neither the number of POD input vectors is known a priori (think of adaptive time stepping schemes) nor the number of vectors which are to be approximated by the generated POD space (i.e., the number of reduced model evaluations). Thus, we want to define ε such that the mean ℓ^2-error is bounded by a desired target tolerance ε^*, independently from the total number of input modes $|\mathcal{S}|$. At the same time, the number of HAPOD output modes should not be much larger than the optimal quantity $|\overline{\mathrm{POD}}(\mathcal{S},\varepsilon^*)|$, where

$$\overline{\mathrm{POD}}(\mathcal{S},\varepsilon^*) := \mathrm{POD}(\mathcal{S},\sqrt{|\mathcal{S}|}\cdot\varepsilon^*).$$

In view of the above results, this motivates the following choice for ε, where the parameter ω allows us to choose a trade-off between the efficiency of the HAPOD and the optimality of the resulting approximation space.

THEOREM 10. *Using the same notation as in Theorem* 8*, let for* $\varepsilon^* > 0$ *the HAPOD tolerances* $\varepsilon(\rho_\mathcal{T})$, $\varepsilon(\alpha)$, $\alpha \in \mathcal{N}_\mathcal{T} \setminus \{\rho_\mathcal{T}\}$ *be given by*

$$\varepsilon(\rho_\mathcal{T}) := \sqrt{|\mathcal{S}|}\cdot\omega\cdot\varepsilon^*, \qquad \varepsilon(\alpha) := \sqrt{|\widetilde{\mathcal{S}}_\alpha|}\cdot(L_\mathcal{T}-1)^{-1/2}\cdot\sqrt{1-\omega^2}\cdot\varepsilon^*,$$

where $0 \le \omega \le 1$ *is an arbitrary parameter. Then we have the following bounds for the final* ℓ^2*-mean approximation error and number of HAPOD modes:*

$$\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\|s-P_{\rho_\mathcal{T}}(s)\|^2 \le \varepsilon^{*2} \quad \textit{and} \quad \Big|\mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon]\Big| \le \Big|\overline{\mathrm{POD}}(\mathcal{S},\omega\cdot\varepsilon^*)\Big|. \tag{7}$$

Moreover, the number of HAPOD modes at the intermediate stages α *is bounded by*

$$\Big|\mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon](\alpha)\Big| \le \Big|\overline{\mathrm{POD}}(\widetilde{\mathcal{S}}_\alpha,(L_\mathcal{T}-1)^{-1/2}\cdot\sqrt{1-\omega^2}\cdot\varepsilon^*)\Big|. \tag{8}$$

Remark 11. Note that the number of local POD modes $|\,\mathrm{HAPOD}[\mathcal{S},\mathcal{T},D,\varepsilon](\alpha)|$ determines the size of the input $\mathcal{S}_\beta$ for the next POD at the parent node β and hence the effort required for its computation. Choosing a large $\omega \to 1$ will reduce the number of final HAPOD modes at the price of larger local PODs. A small $\omega \to 0$ will minimize the costs for computing the HAPOD in exchange for a larger number of final modes to guarantee the prescribed error bound.

Remark 12. Since we consider the *mean* square approximation error, it is possible for the bound (8) that we have

$$|\overline{\mathrm{POD}}(\widetilde{\mathcal{S}}_\alpha,\delta)| > |\overline{\mathrm{POD}}(\mathcal{S},\delta)|,$$

where $\delta := (L_\mathcal{T}-1)^{-1/2}\cdot\sqrt{1-\omega^2}\cdot\varepsilon^*$. This might be the case when the principal directions of the snapshot set $\widetilde{\mathcal{S}}_\alpha$ are underrepresented in the full snapshot set $\mathcal{S}$. However, if $N' := \min\{N\in\mathbb{N} \,\big|\, d_N(\mathcal{S}) \le \delta\}$, where

$$d_N(\mathcal{S}) := \min_{\substack{X\subseteq V \text{ lin subsp.}\\ \dim X\le N}} \max_{s\in\mathcal{S}} \|s-P_X(s)\|$$

is the so-called Kolomogorov N-width of $\mathcal{S}$, and $X_{N'}$ is a minimizer for $d_{N'}(\mathcal{S})$, then we always have

$$|\widetilde{\mathcal{S}}_\alpha|^{-1} \sum_{s\in\widetilde{\mathcal{S}}_\alpha} \|s - P_{X_{N'}}(s)\|^2 \le \max_{s\in\widetilde{\mathcal{S}}_\alpha} \|s - P_{X_{N'}}(s)\|^2 \le \max_{s\in\mathcal{S}} \|s - P_{X_{N'}}(s)\|^2 \le \delta.$$

Thus, due to the optimality of the POD (Theorem 4) we have $|\overline{\mathrm{POD}}(\widetilde{\mathcal{S}}_\alpha, \delta)| \le N'$, and the number of modes at α can be bounded by

$$\left|\mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)\right| \le \min\left\{N \in \mathbb{N} \;\middle|\; d_N(\mathcal{S}) \le (L_\mathcal{T} - 1)^{-1/2} \cdot \sqrt{1-\omega^2} \cdot \varepsilon^*\right\}. \tag{9}$$

In many cases it is known theoretically or heuristically that $d_N(\mathcal{S})$ shows rapid (sub-)exponential decay for increasing N. In these cases, (9) will be an effective upper bound for the number of local HAPOD modes, independent of the chosen snapshot distribution D.

Remark 13 (low-rank approximation of the snapshot mapping). By additionally keeping track of the local right-singular vectors appearing in the HAPOD algorithm, we easily obtain a low-rank approximation of the global snapshot mapping $\widetilde{\mathcal{S}}_\alpha : \mathbb{R}^{|\widetilde{\mathcal{S}}_\alpha|} \to V$ defined in Definition 14. More precisely, by Lemma 15 and (17) we immediately have the rank-$\left|\mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)\right|$ approximation:

$$\|\widetilde{\mathcal{S}}_\alpha - \Psi_\alpha \circ \widetilde{\Lambda}_\alpha^*\|_2^2 \;\le \sum_{\gamma\in\mathcal{N}_\mathcal{T}(\alpha)} \varepsilon(\gamma)^2 \tag{10}$$

in the Hilbert–Schmidt (Frobenius) norm, with Ψ_α, $\widetilde{\Lambda}_\alpha$ given as in Definition 14.

3.4. Algorithmic benefits. Theorems 8 and 9 show that, with an appropriate choice of local error tolerances ε (Theorem 10), the HAPOD produces approximation spaces of a quality comparable to a POD with the same target error tolerance. At the same time, the HAPOD offers several benefits, which for problems with fast decaying singular values can lead to dramatic speedups in computation time.

Reduced memory requirements. If the input data for a POD cannot be kept completely in memory, huge performance penalties are to be expected, since for standard POD algorithms, repeated access of every snapshot vector is required. If the data is kept on a mass storage device, the overall performance of the algorithm will usually be bounded by the data transfer speed.

For the HAPOD, at each node α, only the vectors $\mathcal{S}_\alpha$ are required as input to a local POD, where, typically, $|\mathcal{S}_\alpha| \ll |\mathcal{S}|$ so that $\mathcal{S}_\alpha$ can be kept completely in memory.

If only the POD, and not the snapshots themselves, is targeted by the computation, the HAPOD can obtain the result without accessing mass storage whatsoever (cf. section 4.3). In particular, an incremental HAPOD of a time series may be computed even if the whole time series would not fit into memory (cf. sections 3.2 and 4.1).

Simple parallelization. To compute the local POD at node α, only the output of the PODs at the child nodes $\mathcal{C}_\mathcal{T}(\alpha)$ is required. In particular, for each $1 \le l \le L_\mathcal{T}$, all PODs at the nodes $\{\alpha \in \mathcal{N}_\mathcal{T} \mid L_\mathcal{T}(\alpha) = l\}$ can be computed in parallel without any communication, which is typically the bottleneck for distributed computations. Intermediate results have to be communicated only vertically up the tree, and the communicated data encompasses only low-rank quantities of computed POD modes and singular values (cf. sections 3.2, 4.2, and 4.3).

Generality. The HAPOD can be applied using any pre-existing, optimized POD algorithm. For instance, the HAPOD could be used to perform incremental data compression for an MPI (Message Passing Interface) [33] distributed model, where each sub-POD is computed via a parallelized SVD algorithm. In section 4.3 we speed up the POD algorithm in Remark 3 by exploiting the block structure of the local Gramian similar to Brand's algorithm [7].

Lower algorithmic complexity. A widely-used, simple, and reliable algorithm for POD computation is to compute the eigenvalue decomposition of the Gramian to $\mathcal{S}$ (cf. Remark 3). In the case of $|\mathcal{S}| \ll d := \dim(V)$, the Gramian computation dominates the overall runtime for the algorithm with a computational complexity of $\mathcal{O}(|\mathcal{S}|^2 d)$. For larger snapshots sets $\mathcal{S}$, the quadratic increase in complexity makes this method expensive in comparison to more advanced algorithms (such as Lanczos or randomized methods [10, 17]), which scale only linearly in the number of snapshot vectors

Application of the HAPOD algorithm largely mitigates this issue. In particular, for a balanced n-ary tree $\mathcal{T}$ with single vectors attached to the leaves, the HAPOD using this POD algorithm requires at most $\mathcal{O}(|\mathcal{S}| \log(|\mathcal{S}|) \widehat{N}^2 d)$ operations for Gramian computation, where $\widehat{N} := \max_{\alpha \in \mathcal{N}_\mathcal{T}} |\operatorname{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)|$ denotes the maximum number of local output modes. Assuming that the error tolerances ε are chosen according to Theorem 10 for fixed ε^*, ω, and assuming that the Kolmogorov widths $d_N(\mathcal{S})$ are bounded for growing $\mathcal{S}$, then, due to (9), $\widehat{N}$ will only depend on the depth $L_\mathcal{T}$ of $\mathcal{T}$. If we furthermore assume that $d_N(\mathcal{S})$ decays exponentially with increasing N, we have $\widehat{N} = \mathcal{O}(\log(L_\mathcal{T})) = \mathcal{O}(\log(\log(|\mathcal{S}|)))$. Thus, the overall effort for computing the Gramians is reduced to $\mathcal{O}(|\mathcal{S}| \log(|\mathcal{S}|) \log(\log(|\mathcal{S}|))^2 d)$.

3.5. Proofs of main theorems. In this section we prove our main results (Theorems 8 and 9). We will require some additional notation.

DEFINITION 14 (additional notation). *For each $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$ fix an arbitrary enumeration $\mathcal{C}_\mathcal{T}(\alpha, 1), \dots, \mathcal{C}_\mathcal{T}(\alpha, |\mathcal{C}_\mathcal{T}(\alpha)|)$ of $\mathcal{C}_\mathcal{T}(\alpha)$. For each $\alpha \in \mathcal{N}_\mathcal{T}$ we define mappings*

$$\underline{\mathcal{S}}_\alpha : \mathbb{R}^{|\mathcal{S}_\alpha|} \to V, \quad \Psi_\alpha : \mathbb{R}^{N_\alpha} \to V, \quad \Lambda_\alpha : \mathbb{R}^{N_\alpha} \to \mathbb{R}^{|\mathcal{S}_\alpha|},$$

$N_\alpha := |\operatorname{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)|$, as follows.

As in (1)*, let $\underline{\mathcal{S}}_\alpha$ map the nth canonical basis vector of $\mathbb{R}^{|\mathcal{S}_\alpha|}$ to the nth element of $\mathcal{S}_\alpha$ for a given enumeration of $\mathcal{S}_\alpha$. For $\alpha \in \mathcal{L}_\mathcal{T}$, the enumeration of $\mathcal{S}_\alpha = D^{-1}(\{\alpha\})$ is chosen arbitrarily. For $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$, the enumeration is chosen such that the following compatibility relation is satisfied:*

$$\underline{\mathcal{S}}_\alpha = [\Psi_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \Psi_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}]. \tag{11}$$

For $\varepsilon(\alpha) > 0$, let Ψ_α, Λ_α be the linear mappings given by

$$\Psi_\alpha(e_n) := \sigma_n \cdot \varphi_n, \quad \Lambda_\alpha(e_n) := \lambda_n,$$

where e_n is the nth canonical basis vector of $\mathbb{R}^{N_\alpha}$, and σ_n, φ_n, λ_n denote the nth singular value, left-singular vector, and right-singular vector of $\underline{\mathcal{S}}_\alpha$. Thus, $\Psi_\alpha \circ \Lambda_\alpha^*$ is the truncated SVD of $\underline{\mathcal{S}}_\alpha$. In particular, we have

$$P_\alpha \circ \underline{\mathcal{S}}_\alpha = \Psi_\alpha \circ \Lambda_\alpha^*, \qquad \Lambda_\alpha^* \circ \Lambda_\alpha = 1. \tag{12}$$

For $\varepsilon(\alpha) = 0$ (in which case $N_\alpha = |\mathcal{S}_\alpha|$), we simply let $\Psi_\alpha := \underline{\mathcal{S}}_\alpha$, and let Λ_α be the identity on $\mathbb{R}^{N_\alpha}$ such that (12) *holds as well.*

Note that since $\mathcal{S}_\alpha$ consists exactly of elements $\Psi_\beta(e_n)$ with $\beta \in \mathcal{C}_\mathcal{T}(\alpha)$, $1 \le n \le N_\beta$, it is clear that (11) *can always be satisfied.*

Finally, we define cumulative mappings $\widetilde{\underline{\mathcal{S}}}_\alpha, \widetilde{\underline{\mathcal{R}}}_\alpha : \mathbb{R}^{|\widetilde{\mathcal{S}}_\alpha|} \to V$, $\widetilde{\Lambda}_\alpha : \mathbb{R}^{N_\alpha} \to \mathbb{R}^{|\widetilde{\mathcal{S}}_\alpha|}$ recursively as

$$\widetilde{\underline{\mathcal{S}}}_\alpha := \underline{\mathcal{S}}_\alpha, \qquad \widetilde{\underline{\mathcal{R}}}_\alpha := \underline{\mathcal{S}}_\alpha, \qquad \widetilde{\Lambda}_\alpha := \Lambda_\alpha,$$

for $\alpha \in \mathcal{L}_\mathcal{T}$ and

$$\widetilde{\underline{\mathcal{S}}}_\alpha := [\widetilde{\underline{\mathcal{S}}}_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\underline{\mathcal{S}}}_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}], \quad \widetilde{\Lambda}_\alpha := \operatorname{diag}(\widetilde{\Lambda}_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}) \circ \Lambda_\alpha,$$
$$\widetilde{\underline{\mathcal{R}}}_\alpha := [P_{\mathcal{C}_\mathcal{T}(\alpha,1)} \circ \widetilde{\underline{\mathcal{R}}}_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, P_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)} \circ \widetilde{\underline{\mathcal{R}}}_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}]$$

for all $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$. Similar to the definition of $\underline{\mathcal{S}}_\alpha$, the map $\widetilde{\underline{\mathcal{S}}}_\alpha$ is of the form (1) *with respect to a specific enumeration of $\widetilde{\mathcal{S}}_\alpha$.*

As a first step towards the proof of our main theorems, we will extend the decomposition (12) to the accumulated mapping of projected snapshots $\widetilde{\underline{\mathcal{R}}}_\alpha$.

LEMMA 15. *With the same notation as in Definition* 14 *we have for all $\alpha \in \mathcal{N}_\mathcal{T}$:*

$$P_\alpha \circ \widetilde{\underline{\mathcal{R}}}_\alpha = \Psi_\alpha \circ \widetilde{\Lambda}_\alpha^*, \qquad \widetilde{\Lambda}_\alpha^* \circ \widetilde{\Lambda}_\alpha = 1. \tag{13}$$

In particular, it follows for $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$ that

$$\widetilde{\underline{\mathcal{R}}}_\alpha = \underline{\mathcal{S}}_\alpha \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}). \tag{14}$$

Proof. We show the claim via induction over $\mathcal{T}$. To this end, first note that for $\alpha \in \mathcal{L}_\mathcal{T}$, (13) is precisely (12) by definition of $\widetilde{\underline{\mathcal{R}}}_\alpha, \widetilde{\Lambda}_\alpha$. For $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$, we obtain using the induction hypothesis the definition of $\underline{\mathcal{S}}_\alpha$ and (12):

$$\begin{aligned}
P_\alpha \circ \widetilde{\underline{\mathcal{R}}}_\alpha &= P_\alpha \circ [P_{\mathcal{C}_\mathcal{T}(\alpha,1)} \circ \widetilde{\underline{\mathcal{R}}}_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, P_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)} \circ \widetilde{\underline{\mathcal{R}}}_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}] \\
&= P_\alpha \circ [\Psi_{\mathcal{C}_\mathcal{T}(\alpha,1)} \circ \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \Psi_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)} \circ \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}] \\
&= P_\alpha \circ \underline{\mathcal{S}}_\alpha \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}) \\
&= \Psi_\alpha \circ \Lambda_\alpha^* \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}) \\
&= \Psi_\alpha \circ \widetilde{\Lambda}_\alpha^*.
\end{aligned}$$

Moreover,

$$\widetilde{\Lambda}_\alpha^* \circ \widetilde{\Lambda}_\alpha = \Lambda_\alpha^* \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)} \circ \widetilde{\Lambda}_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)} \circ \widetilde{\Lambda}_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}) \circ \Lambda_\alpha = 1.$$

Thus, (13) is proved, and we have

$$\begin{aligned}
\widetilde{\underline{\mathcal{R}}}_\alpha &= [P_{\mathcal{C}_\mathcal{T}(\alpha,1)} \circ \widetilde{\underline{\mathcal{R}}}_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, P_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)} \circ \widetilde{\underline{\mathcal{R}}}_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}] \\
&= [\Psi_{\mathcal{C}_\mathcal{T}(\alpha,1)} \circ \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \Psi_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)} \circ \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}] \\
&= \underline{\mathcal{S}}_\alpha \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)}).
\end{aligned}$$

∎

As a final preparatory step, we show the following orthogonality lemma.

LEMMA 16. *With the same notation as in Definition* 14 *we have for all $\alpha \in \mathcal{N}_\mathcal{T}$ and arbitrary continuous linear maps $X, Y : V \to V$:*

$$(X \circ (\widetilde{\underline{\mathcal{S}}}_\alpha - \widetilde{\underline{\mathcal{R}}}_\alpha), Y \circ \widetilde{\underline{\mathcal{R}}}_\alpha)_2 = 0, \tag{15}$$

where $(A, B)_2$ is the Hilbert–Schmidt inner product given by $\operatorname{tr}(A^* B)$.

Proof. We prove the claim again via induction over $\mathcal{T}$. For $\alpha \in \mathcal{L}_\mathcal{T}$, the statement is obvious since $\widetilde{\underline{\mathcal{S}}}_\alpha = \underline{\mathcal{S}}_\alpha = \widetilde{\underline{\mathcal{R}}}_\alpha$. For $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$, we have

$$
\begin{aligned}
(16)\quad & (X \circ (\widetilde{\underline{\mathcal{S}}}_\alpha - \widetilde{\underline{\mathcal{R}}}_\alpha), Y \circ \widetilde{\underline{\mathcal{R}}}_\alpha)_2 \\
&= \sum_{\beta \in \mathcal{C}_\mathcal{T}(\alpha)} (X \circ (\widetilde{\underline{\mathcal{S}}}_\beta - P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta), Y \circ P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta)_2 \\
&= \sum_{\beta \in \mathcal{C}_\mathcal{T}(\alpha)} (X \circ (\widetilde{\underline{\mathcal{S}}}_\beta - \widetilde{\underline{\mathcal{R}}}_\beta), Y \circ P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta)_2 \\
&\qquad + \sum_{\beta \in \mathcal{C}_\mathcal{T}(\alpha)} (X \circ (1 - P_\beta) \circ \widetilde{\underline{\mathcal{R}}}_\beta, Y \circ P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta)_2.
\end{aligned}
$$

The first sum on the right-hand side of (16) vanishes by the induction hypothesis (with $Y := Y \circ P_\beta$). To handle the second sum note that for $\beta \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$, $\varepsilon(\beta) > 0$ we can use (14) to write

$$
\begin{aligned}
(1 - P_\beta) \circ \widetilde{\underline{\mathcal{R}}}_\beta &= (1 - P_\beta) \circ \underline{\mathcal{S}}_\beta \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\beta,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\beta,|\mathcal{C}_\mathcal{T}(\beta)|)}) \\
&= \Psi^c_\beta \circ \Lambda^{c*}_\beta \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\beta,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\beta,|\mathcal{C}_\mathcal{T}(\beta)|)}),
\end{aligned}
$$

where $\Psi^c_\beta : \mathbb{R}^{|\mathcal{S}_\beta| - N_\beta} \to V, \Lambda^c_\beta : \mathbb{R}^{|\mathcal{S}_\beta| - N_\beta} \to \mathbb{R}^{|\mathcal{S}_\beta|}$ map the kth canonical basis vector to the $(N_\beta + k)$th scaled left (unscaled right) singular vector of $\underline{\mathcal{S}}_\beta$. In particular, $\Lambda^*_\beta \circ \Lambda^c_\beta = 0$. Using (13) and the invariance of the trace under cyclic permutations, we obtain

$$
\begin{aligned}
&(X \circ (1 - P_\beta) \circ \widetilde{\underline{\mathcal{R}}}_\beta, Y \circ P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta)_2 \\
&\quad = \operatorname{tr}(\{(1 - P_\beta) \circ \widetilde{\underline{\mathcal{R}}}_\beta\}^* \circ X^* \circ Y \circ P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta) \\
&\quad = \operatorname{tr}(X^* \circ Y \circ P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta \circ \{(1 - P_\beta) \circ \widetilde{\underline{\mathcal{R}}}_\beta\}^*) \\
&\quad = \operatorname{tr}(X^* \circ Y \circ \Psi_\beta \circ \Lambda^*_\beta \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\beta,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\beta,|\mathcal{C}_\mathcal{T}(\beta)|)}) \\
&\qquad\qquad \circ \operatorname{diag}(\widetilde{\Lambda}_{\mathcal{C}_\mathcal{T}(\beta,1)}, \dots, \widetilde{\Lambda}_{\mathcal{C}_\mathcal{T}(\beta,|\mathcal{C}_\mathcal{T}(\beta)|)}) \circ \Lambda^c_\beta \circ \Psi^{c*}_\beta) \\
&\quad = \operatorname{tr}(X^* \circ Y \circ \Psi_\beta \circ \{\Lambda^*_\beta \circ \Lambda^c_\beta\} \circ \Psi^c_\beta) = 0.
\end{aligned}
$$

The same line of argument holds for $\beta \in \mathcal{L}_\mathcal{T}$, where we have $(1 - P_\beta) \circ \widetilde{\underline{\mathcal{R}}}_\beta = \Psi^c_\beta \circ \Lambda^{c*}_\beta$. Since for $\varepsilon(\beta) = 0$ we trivially have $1 - P_\beta = 0$, we see that the second sum in (16) always vanishes, proving the claim. ∎

Proof of Theorem 8. First note that, due to the best approximation property of the orthogonal projection P_α, we have

$$
\begin{aligned}
\sum_{s \in \widetilde{\mathcal{S}}_\alpha} \|s - P_\alpha(s)\|^2 &= \sum_{n=1}^{|\widetilde{\mathcal{S}}_\alpha|} \|\widetilde{\underline{\mathcal{S}}}_\alpha(e_n) - P_\alpha(\widetilde{\underline{\mathcal{S}}}_\alpha(e_n))\|^2 \\
&\le \sum_{n=1}^{|\widetilde{\mathcal{S}}_\alpha|} \|\widetilde{\underline{\mathcal{S}}}_\alpha(e_n) - P_\alpha(\widetilde{\underline{\mathcal{R}}}_\alpha(e_n))\|^2 \\
&= \|\widetilde{\underline{\mathcal{S}}}_\alpha - P_\alpha \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2,
\end{aligned}
$$

where $\|A\|_2 = \sqrt{(A,A)_2} = \sqrt{\operatorname{tr}(A^*A)}$ denotes the Hilbert–Schmidt norm of A. Thus, the theorem is proven if we can show that for all $\alpha \in \mathcal{N}_\mathcal{T}$ the following estimate holds:

$$\|\widetilde{\underline{\mathcal{S}}}_\alpha - P_\alpha \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 \le \sum_{\gamma \in \mathcal{N}_\mathcal{T}(\alpha)} \varepsilon(\gamma)^2. \tag{17}$$

We show (17) again via induction over $\mathcal{T}$. For $\alpha \in \mathcal{L}_\mathcal{T}$ we immediately have

$$\|\widetilde{\underline{\mathcal{S}}}_\alpha - P_\alpha \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 = \|\underline{\mathcal{S}}_\alpha - P_\alpha \circ \underline{\mathcal{S}}_\alpha\|_2^2 \le \varepsilon(\alpha)^2 = \sum_{\gamma \in \mathcal{N}_\mathcal{T}(\alpha)} \varepsilon(\gamma)^2$$

according to Definition 5.

Now, let us assume that (17) holds for all $\beta \in \mathcal{C}_\mathcal{T}(\alpha)$ for some $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$. Using Lemma 16 with $Y = I - P_\alpha$, we have

$$\|\widetilde{\underline{\mathcal{S}}}_\alpha - P_\alpha \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 = \|\widetilde{\underline{\mathcal{S}}}_\alpha - \widetilde{\underline{\mathcal{R}}}_\alpha + (I - P_\alpha) \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 = \|\widetilde{\underline{\mathcal{S}}}_\alpha - \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 + \|(I - P_\alpha) \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2.$$

Using the induction hypothesis, we can bound the first summand by

$$\begin{aligned}\|\widetilde{\underline{\mathcal{S}}}_\alpha - \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 &= \sum_{\beta \in \mathcal{C}_\mathcal{T}(\alpha)} \|\widetilde{\underline{\mathcal{S}}}_\beta - P_\beta \circ \widetilde{\underline{\mathcal{R}}}_\beta\|_2^2 \\ &\le \sum_{\beta \in \mathcal{C}_\mathcal{T}(\alpha)} \sum_{\gamma \in \mathcal{N}_\mathcal{T}(\beta)} \varepsilon(\gamma)^2 \\ &= \sum_{\gamma \in \mathcal{N}_\mathcal{T}(\alpha) \setminus \{\alpha\}} \varepsilon(\gamma)^2.\end{aligned}$$

To bound the second summand, we use Lemma 15, the fact that $\|T \circ S\|_2 \le \|T\|_2 \cdot \|S\|$ (for arbitrary T, S), and Definition 5 to obtain

$$\begin{aligned}\|(I - P_\alpha) \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 &= \|(I - P_\alpha) \circ \underline{\mathcal{S}}_\alpha \circ \operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)})\|_2^2 \\ &\le \|(I - P_\alpha) \circ \underline{\mathcal{S}}_\alpha\|_2^2 \cdot \|\operatorname{diag}(\widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,1)}, \dots, \widetilde{\Lambda}^*_{\mathcal{C}_\mathcal{T}(\alpha,|\mathcal{C}_\mathcal{T}(\alpha)|)})\|^2 \\ &\le \varepsilon(\alpha)^2.\end{aligned}$$

Thus, (17) follows, which completes the proof. ∎

Proof of Theorem 9. For $\alpha \in \mathcal{L}_\mathcal{T}$ there is nothing to show, so let us assume that $\alpha \in \mathcal{N}_\mathcal{T} \setminus \mathcal{L}_\mathcal{T}$. According to Lemma 15, $\widetilde{\underline{\mathcal{R}}}_\alpha$ and $\underline{\mathcal{S}}_\alpha$ have the same singular values. Thus, with $\widetilde{\mathcal{R}}_\alpha := \{\widetilde{\underline{\mathcal{R}}}_\alpha(e_n) \mid 1 \le n \le |\widetilde{\mathcal{S}}_\alpha|\}$ we have

$$|\operatorname{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)| = |\operatorname{POD}(\mathcal{S}_\alpha, \varepsilon(\alpha))| = |\operatorname{POD}(\widetilde{\mathcal{R}}_\alpha, \varepsilon(\alpha))|.$$

Let $\widetilde{P}_\alpha$ be the orthogonal projection onto the linear span of the modes selected by $\operatorname{POD}(\widetilde{\mathcal{S}}_\alpha, \varepsilon(\alpha))$. Due to Lemma 16 with $X = Y = 1 - \widetilde{P}_\alpha$, we have

$$\begin{aligned}\varepsilon(\alpha)^2 &\ge \|(1 - \widetilde{P}_\alpha) \circ \widetilde{\underline{\mathcal{S}}}_\alpha\|_2^2 \\ &= \|(1 - \widetilde{P}_\alpha) \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2 + \|(1 - \widetilde{P}_\alpha) \circ (\widetilde{\underline{\mathcal{S}}}_\alpha - \widetilde{\underline{\mathcal{R}}}_\alpha)\|_2^2 \\ &\ge \|(1 - \widetilde{P}_\alpha) \circ \widetilde{\underline{\mathcal{R}}}_\alpha\|_2^2.\end{aligned}$$

According to Definition 5 and due to the optimality of the POD, we therefore have

$$|\operatorname{POD}(\widetilde{\mathcal{R}}_\alpha, \varepsilon(\alpha))| \le |\operatorname{POD}(\widetilde{\mathcal{S}}_\alpha, \varepsilon(\alpha))|,$$

which concludes the proof. ∎

Proof of Theorem 10. According to Theorem 8 we have

$$\begin{aligned}\sum_{s\in\mathcal{S}} \|s - P_{\rho_\mathcal{T}}(s)\|^2 &\leq |\mathcal{S}| \cdot \omega^2 \cdot \varepsilon^{*2} + \sum_{l=1}^{L_\mathcal{T}-1} \sum_{\substack{\gamma\in\mathcal{N}_\mathcal{T} \\ L_\mathcal{T}(\gamma)=l}} |\widetilde{\mathcal{S}}_\gamma| \cdot (L_\mathcal{T}-1)^{-1} \cdot (1-\omega^2) \cdot \varepsilon^{*2} \\ &\leq |\mathcal{S}| \cdot \omega^2 \cdot \varepsilon^{*2} + \sum_{l=1}^{L_\mathcal{T}-1} |\mathcal{S}| \cdot (L_\mathcal{T}-1)^{-1} \cdot (1-\omega^2) \cdot \varepsilon^{*2} \\ &= |\mathcal{S}| \cdot \varepsilon^{*2}.\end{aligned}$$

The stated bounds for the number of HAPOD modes follow directly from Theorem 9 and the definition of $\overline{\text{POD}}$. ∎

4. Numerical results. To demonstrate the applicability of the HAPOD, three numerical examples comparing the POD with the HAPOD are presented and evaluated in terms of accuracy and complexity. The first two experiments are implemented in the MATLAB language and performed using Octave [14]. For the POD and HAPOD,[2] the built-in SVD of Octave is utilized, which in turn uses LAPACK [2]. The third experiment is implemented in Python using the POD implementation of the pyMOR library [39], which utilizes the method of snapshots by SciPy's [26] symmetric eigenvalue computation, also via LAPACK.

4.1. Incremental data compression. The first numerical experiment compares the POD and HAPOD through compressing a trajectory of a randomly excited system. As an underlying system, a forced one-dimensional inviscid Burgers equation is chosen:

$$\begin{aligned}\partial_t z(x,t) + z(x,t)\cdot\partial_x z(x,t) = b(x,t), &\qquad (x,t)\in(0,1)\times(0,1),\\ z(x,0) = 0, &\qquad x\in[0,1],\\ z(0,t) = 0, &\qquad t\in[0,1],\end{aligned}$$

with force term $b \in L^2([0,1]\times[0,1])$. A spatial discretization using a conservative finite difference upwind scheme with $N = 500$ equidistant nodes yields a system of nonlinear ordinary differential equations in time [30]:

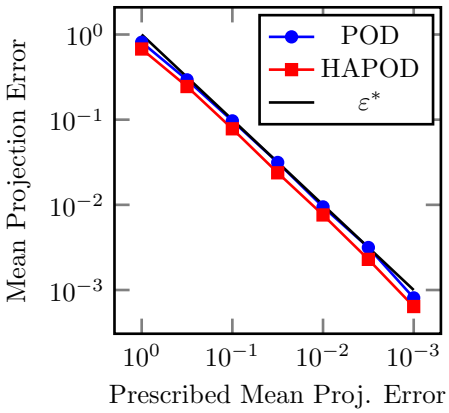
$$\dot{z}(t) = A(z(t)\circ z(t)) + Bu(t),$$

with $\circ$ denoting the elementwise Hadamard product. The experiment runs with constant temporal resolution $h = 10^{-4}$ resulting in 10^4 explicit Euler time steps. As a forcing term, a scaled Gaussian bell curve $b(x,t) = u(t)\exp(-\frac{1}{20}(x-\frac{1}{2})^2)$ is chosen with a time-dependent coefficient $u(t)$ which is 99.9% of all time steps zero, but at random instances over the whole time interval for 0.1% of all time steps it becomes a constant value sampled from the uniform random distribution in the interval $[0, \frac{1}{5}]$. The full order model evolution is visualized in Figure 3a.

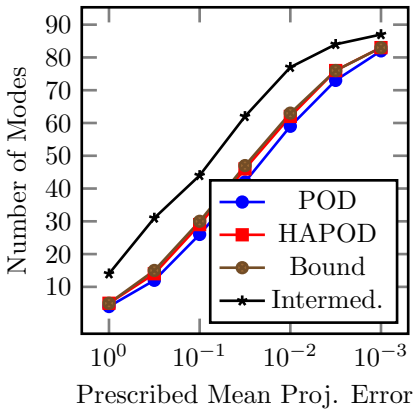
An incremental HAPOD is performed as described in section 3.2 to extract the dominant modes for different accuracies on a subdivision of the full time series into one-hundred uniform length blocks, of which results are compared to a POD over the whole time series. The local error tolerances ε are chosen according to Theorem 10 with $\omega = 0.75$. The computation is conducted on a Raspberry Pi[3] single board

[2]Internally, the HAPOD implementation uses the same POD method as the plain POD.

[3]Rasperry Pi Model 1B: ARMv6-CPU 700MHz, 512MB RAM; see additional information online at http://www.raspberrypi.org/products/model-b.



(a) Actual ℓ^2-mean projection error of POD and incremental HAPOD computation for prescribed errors ε^*.



(b) Number of resulting POD and HAPOD modes, bound (7) for the number of HAPOD modes at output node $\rho_{\mathcal{T}}$, and the maximum number of intermediate HAPOD output modes (8).

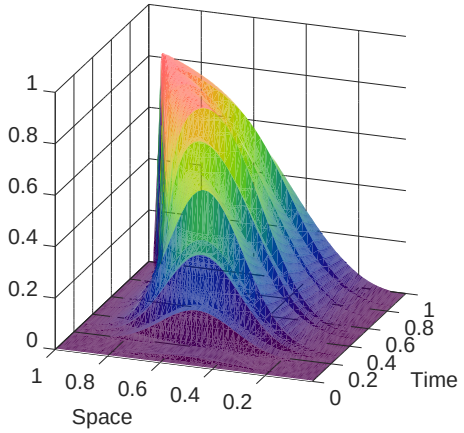
FIG. 2. *Approximation error and mode counts versus prescribed error tolerance for the data compression example with state-space dimension* $N = 500$ *(cf. section* 4.1*).*

computer, which is a memory limited device, comparable to embedded or power-aware environments.

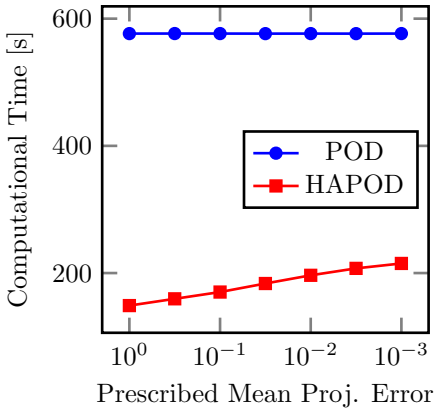
In Figure 2a, the ℓ^2-mean projection error (7) for the prescribed accuracies of $\varepsilon^* \in \{10^0, 10^{-1/2}, 10^{-1}, \ldots, 10^{-3}\}$ is depicted. Due to shock formation in the solution, a relatively large number of POD modes is required for accurate approximation. Thus, in view of the low spatial resolution, the prescribed errors are chosen in a manner to suppress effects of the discretization error in the results. The approximation error of the POD and the incremental HAPOD decay very similarly in rate and magnitude. In terms of the number of modes, Figure 2b shows that also the number of final HAPOD modes increases with the same rate as the classic POD. The HAPOD requires at most four additional modes, and the mode bound (7) overestimates the number of HAPOD modes by at most one. At most 15 additional output modes are generated at the intermediate HAPOD steps.

The time consumption is plotted in Figure 3b for the different ε^*. Since the used POD implementation fully factorizes the given input data, the required computational time for the POD is (almost) constant for different accuracies. The incremental HAPOD time requirements increase with higher accuracies, yet for all tested ε^* the HAPOD requires less time than the POD. Figure 4a shows the computational time for the POD and incremental HAPOD for varying state-space dimension $N = \{250, 500, 750, 1000, 1250, 1500, 1750, 2000\}$, but with fixed prescribed approximation error. For $N > 750$ the regular POD's memory requirements exceed the device capabilities, while the incremental HAPOD is still computable.

Furthermore, the dependence of the number of final HAPOD modes and intermediate modes together with the required computational time is compared for varying block sizes in Figure 4b. While the number of final modes stays almost constant, a smaller block size reduces the computational time at the expense of a slightly larger number of intermediate modes. This demonstrates the HAPOD's configurable trade-

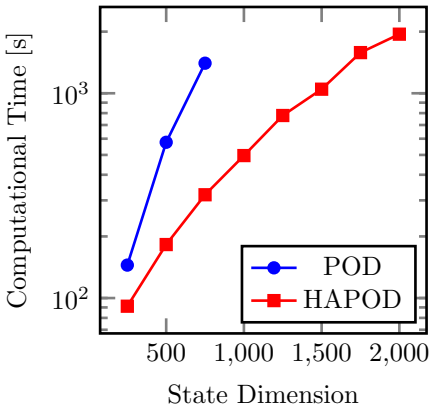


(a) Visualization of the temporal evolution of the Burgers equation example.

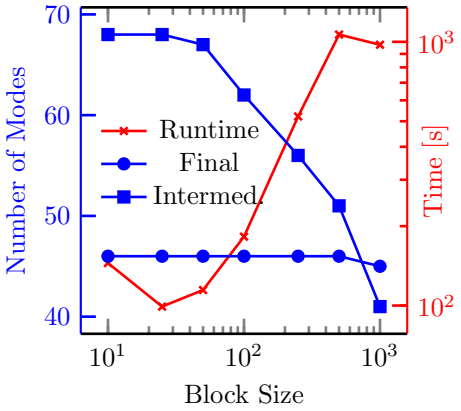


(b) Computational time for POD and incremental HAPOD with state-space dimension $N = 500$.

FIG. 3. *Solution visualization and computational time versus prescribed error for the data compression example (cf. section* 4.1*).*



(a) Computational time for POD and incremental HAPOD with a prescribed error $\varepsilon^* = 10^{-3/2}$ versus different state-space dimensions.



(b) Number of final HAPOD modes and the maximum number of intermediate modes as well as computational time for varying input data block sizes, $\varepsilon^* = 10^{-3/2}$ and $N = 500$.

FIG. 4. *Computational time and mode number versus state dimension and block size (the number of snapshots in a leaf node) for the data compression example (cf. section* 4.1*).*

off between memory and computation time: One can reduce the computational time by using smaller data partitions, but must take into account higher memory consumption for the intermediate modes; on the other hand, by enlarging the block partition size, less memory is consumed during the computation, yet the computational time is increased.

4.2. Distributed empirical cross Gramian. The second numerical experiment compares the POD with the distributed HAPOD computation (cf. section 3.2) in terms of the model reduction error resulting from the respective output modes.

Given a linear state-space control system with the same number of inputs and outputs $\dim(u(t)) = \dim(y(t))$,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{18}$$

the associated cross Gramian matrix [15] is defined as the composition of the system's controllability and observability operators:

$$W_X := \mathcal{CO} = \int_0^\infty \mathrm{e}^{At} BC\, \mathrm{e}^{At}\, \mathrm{d}t.$$

The modes U resulting from a POD of the cross Gramian constitute an approximate balancing transformation, which can be truncated based on the associated singular values:

$$W_X \overset{\text{SVD}}{=} UDV \to U = \begin{pmatrix} U_1 & U_2 \end{pmatrix}.$$

This truncated orthogonal projection induces a reduced order model for (18),

$$\begin{aligned} \dot{x}_r(t) &= (U_1^\intercal A U_1) x_r(t) + (U_1^\intercal B) u(t), \\ y_r(t) &= (CU_1) x_r(t). \end{aligned} \tag{19}$$

For further details, we refer the reader to [49]. Practically, the empirical cross Gramian [21] can be utilized for the computation of the cross Gramian:

$$\begin{aligned} \widehat{W}_X &:= \sum_{m=1}^{M} \int_0^\infty \Psi^m(t)\, \mathrm{d}t \in \mathbb{R}^{N \times N}, \\ \Psi_{ij}^m(t) &:= \langle x_i^m(t), y_m^j(t) \rangle, \end{aligned}$$

with $x^m(t)$ being the state trajectory for a perturbation of the mth component of an impulse input, and $y^j(t)$ the output trajectory for a perturbation of the jth initial state component. The empirical cross Gramian matrix may be assembled columnwise,

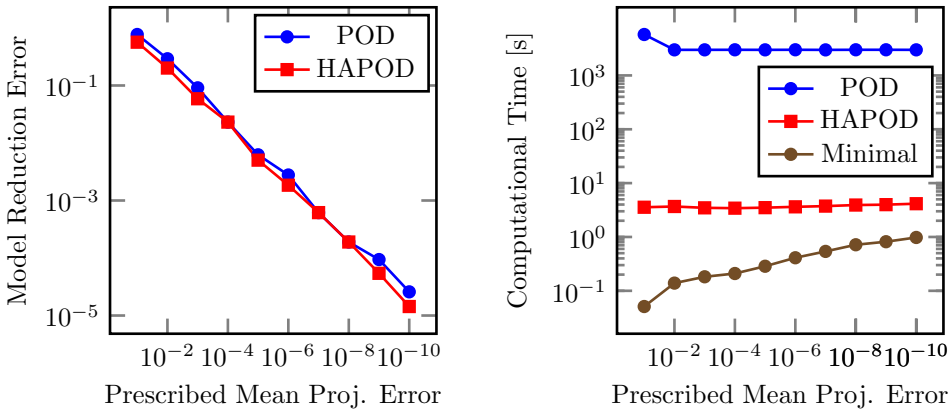
$$\begin{aligned} \widehat{W}_X &= \left[\sum_{m=1}^{M} \int_0^\infty \psi^{m1}(t)\, \mathrm{d}t, \ldots, \sum_{m=1}^{M} \int_0^\infty \psi^{mN}(t)\, \mathrm{d}t \right], \\ \psi_i^{mn}(t) &:= \langle x_i^m(t), y_m^n(t) \rangle, \end{aligned} \tag{20}$$

by sorting the $\Psi^m(t)$ into columns. This *distributed empirical cross Gramian* together with the distributed HAPOD computation then allows a fully parallel assembly of the cross-Gramian-based approximate balancing truncated projection U_1.

This experiment utilizes the procedural "Synthetic" benchmark model[4] from [36]. For $N = 10000$ a single-input-single-output system is generated, and we fix the parametrization to $\theta \equiv \frac{1}{10}$. The system is excited by an impulse input $u(t) = \delta(t)$ and evolves over a time span of $T = [0, 1]$ with a fixed time step width of $h = \frac{1}{100}$. An empirical cross Gramian $\widehat{W}_X$ is computed[5] using `emgr` (empirical Gramian framework) [20, 18, 19], for which a regular POD and a distributed HAPOD is used to determine the left-singular vectors. For the latter, the empirical cross Gramian

[4] See http://modelreduction.org/index.php/Synthetic_parametric_model.

[5] Computation on Intel Core i7-6700 (x86-64) CPU with 8GB RAM.



(a) Actual model reduction output ℓ^2-error of POD and distributed HAPOD for prescribed errors ε^*.

(b) Computational time for POD, distributed HAPOD time (sequential computation), and minimal required HAPOD time if full parallelization is assumed.

FIG. 5. *Comparison of model reduction error and computational time for the POD and distributed HAPOD computation for the distributed empirical cross Gramian example (cf. section* 4.2*) for varying prescribed projection error.*

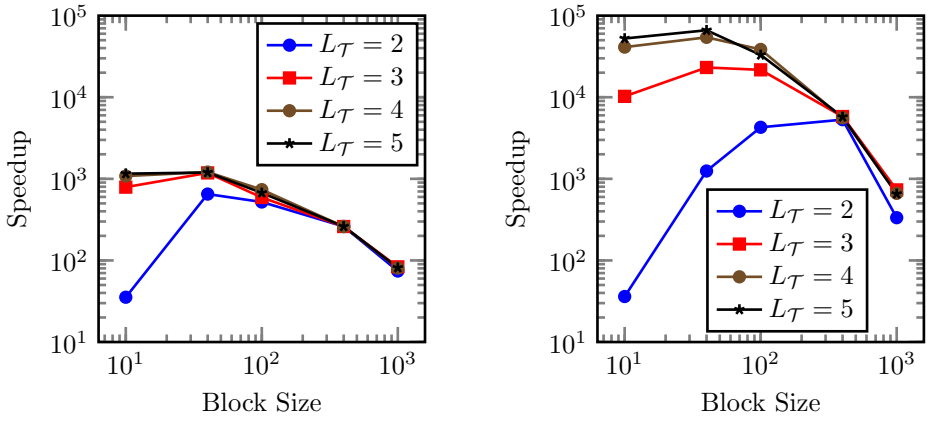
$\widehat{W}_X \in \mathbb{R}^{10000\times 10000}$ is partitioned columnwise into 100 blocks of size 10000×100, which are assigned to the leafs of the distributed HAPOD tree, and the local error tolerances chosen according to Theorem 10 with $\omega = 0.5$.

Figure 5a shows the error for the empirical cross Gramian-based state-space reduction comparing the original system's output and the reduced order model's output utilizing either the POD or the distributed variant of the HAPOD. For a varying prescribed projection error, the model reduction error resulting from the POD and HAPOD, i.e., the time-domain misfit between original system output and reduced-order system output measured in the ℓ^2-norm $\varepsilon_y = \|y - y_r\|_{\ell^2}$, decays with a similar rate as, and never exceeds the error resulting from, the classic POD.

Comparing the time consumption of the POD and HAPOD, the former, due to its constant complexity, requires a fixed amount of time for each prescribed error. The HAPOD assembly time is about three orders of magnitude smaller than for the POD and increases slowly for more accurate approximations, as shown in Figure 5b. Furthermore, if enough processor cores would be available for a full parallelization, meaning that all leaf sub-PODs could be evaluated concurrently, then for $\varepsilon^* \geq 10^{-6}$ the time requirements can be reduced again by up to one order of magnitude compared with the single worker setup used in the experiment. For smaller prescribed errors, the final POD starts to require a large part of the computational effort such that a balanced tree $\mathcal{T}$ with depth $L_\mathcal{T} = 3$ would be required to gain an additional speedup.

The next experiment tests the influence of the depth of the tree and the block size at the leafs on the runtime. To this end the 10^4 columns of the empirical cross Gramian are organized in partitions of 10×1000, 40×250, 100×100, 400×25, and 1000×10 columns. These partitions are each mapped to the leafs of balanced n-ary trees of depth $L_\mathcal{T} \in \{2, 3, 4, 5\}$. The number of children per node n is determined for each tree by the number of blocks s and the depth $L_\mathcal{T}$ of the tree via $n = \lceil s^{1/L_\mathcal{T}} \rceil$.

Figure 6 depicts the speedup of the HAPOD over a classic POD for varying tree depths and block sizes at the leafs. Specifically, Figure 6a shows the speedup for



(a) Sequential runtime of the HAPOD.

(b) Maximal speedup of the HAPOD assuming full parallelization.

FIG. 6. *Speedup of the HAPOD for balanced trees of different depth and block sizes (cf. section 4.2), $\varepsilon^* = 10^{-6}$, in comparison to the classic POD. The runtime for the classic POD is $2.98 \cdot 10^3$ seconds.*

a sequential execution of the HAPOD, while Figure 6b shows the maximal speedup assuming s processors by summing the maximum sub-POD runtimes for each level, as these sub-PODs could be processed in parallel.

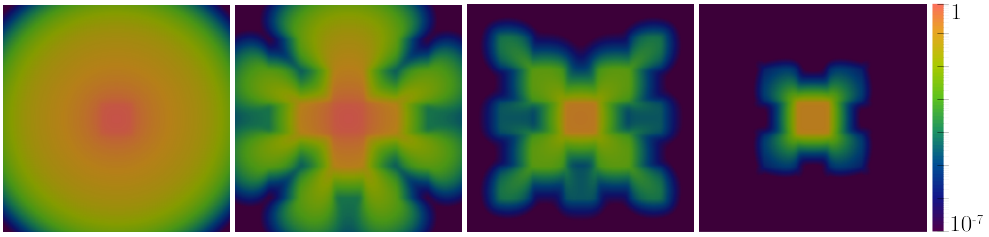
This test shows that (balanced) trees with smaller blocks are preferable in terms of runtime (Figure 6a). For highly parallel computations, trees with small block sizes and more levels (depth) perform better (Figure 6b). While the two-level tree with smallest block size performs worst in comparison, the larger the individual leaf block, the more similar the runtimes independent from tree depth.

4.3. Reduction of a large kinetic equation model. The third numerical experiment utilizes a kinetic equation model. In such models, the solution field does not only depend on time and space but also on velocity variables. Hence, directly solving a kinetic equation with standard numerical methods often causes a prohibitive amount of computational cost due to the curse of dimensionality. Moment closure models are one approach to overcome this difficulty by transferring the kinetic equation to a hyperbolic system of coupled equations which do not depend on the velocity variable anymore (see [1, 9, 45] and references therein). This significantly reduces the effort needed to solve the problem, especially in several space dimensions. However, the computational cost may still be too high to solve a parameter-dependent problem for a large set of parameters in a reasonable amount of time. In this case, a POD-based state-space Galerkin projection similar to (19) can be used to further reduce the model.

Our experiment is based on the checkerboard test case for the P_{15} moment closure approximation of the Boltzmann equation for neutron transport from [9]. The model equation in two dimensions is given by

$$\partial_t \mathbf{p}(t,\mathbf{x}) + \mathbf{A}_x \partial_x \mathbf{p}(t,\mathbf{x}) + \mathbf{A}_z \partial_z \mathbf{p}(t,\mathbf{x}) = \mathbf{s}(t,\mathbf{x}) + (\Sigma_s(\mathbf{x})\mathbf{Q} - \Sigma_t(\mathbf{x})\mathbf{I})\,\mathbf{p}(t,\mathbf{x}),$$

where $\mathbf{p}(t,\mathbf{x}) \in \mathbb{R}^{136}$ for fixed spatial coordinates $\mathbf{x} = (x,z)$ and time t, $\mathbf{I}$ is the identity matrix, and $\mathbf{Q}_{00} = 1$, $\mathbf{Q}_{ij} = 0$ otherwise. The positive coefficients Σ_s and $\Sigma_t = \Sigma_s + \Sigma_a$ describe scattering and total cross section, respectively, and $\mathbf{s}$ is a particle



(a) $\mu = (0, 0, 0)$ (b) $\mu = (0, 0, 6)$ (c) $\mu = (2, 6, 0)$ (d) $\mu = (8, 8, 4)$

FIG. 7. *Solutions to the Checkerboard test case for the kinetic Boltzmann equation (cf. section 4.3) for different parameters* $\mu = (\Sigma_{s,1}, \Sigma_{a,1}, \Sigma_{a,2})$. *Visualized is the first component of the solution at time* $T = 3.2$. *The color scale is logarithmic.*

source. The matrices $\mathbf{A}_x$, $\mathbf{A}_z \in \mathbb{R}^{136\times136}$ which describe the coupling between the moments are sparse with at most four and two entries per row, respectively. See [9, eqs. 8 and 9] for detailed definitions of the matrices.

The test case assumes a spatial domain $[0, 7] \times [0, 7]$ that is divided into 49 axis-parallel cubes with unit edge width and composed of two different materials (see Figure 11a) that are characterized by their scattering and absorption cross-section Σ_s and Σ_a, respectively. Initially, there are no neutrons in the domain. At time $t = 0$, a neutron source $\mathbf{s} = (1, 0, \dots, 0)^\intercal$ is turned on in the center region.

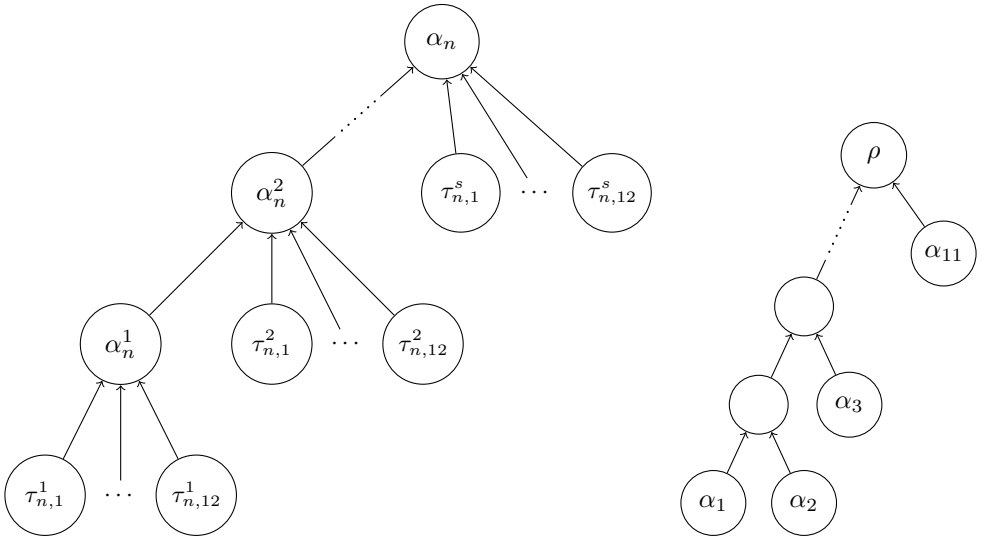
The parameter dependence for the scattering and absorption cross-sections $\Sigma_{s,1}$ and $\Sigma_{a,1}$ for the first material (red regions in Figure 11a) and the absorption cross-section $\Sigma_{a,2}$ for the second material (black regions in Figure 11a) is to be retained for the reduced order model, while the scattering cross-section of the second material is fixed to $\Sigma_{s,2} = 0$. The three parameters $\Sigma_{s,1}$, $\Sigma_{a,1}$, $\Sigma_{a,2}$ are each chosen in the range $[0, 8]$. For the POD, each parameter is uniformly sampled by the five values $\{0, 2, 4, 6, 8\}$ such that 125 solution trajectories have to be calculated.

The model is solved by a finite volume solver for systems of hyperbolic equations implemented in dune-gdt [29, 44] using a numerical Lax–Friedrichs flux and an explicit Euler fractional step time stepping scheme (see [31, Chap. 17.1]) to incorporate the right-hand side into the solution. Solutions for some exemplary parameter choices are visualized in Figure 7.

As the P_{15} model consists of 136 coupled equations with 136 unknowns and the finite volume scheme uses a uniform cube grid with k^2 elements, the discrete solution vector for the finite volume discretization at a fixed time contains $N = 136k^2$ entries. The test case is solved up to a time of $T = 3.2$, and the time step length is determined by a Courant–Friedrichs–Lewy number of 0.4, which leads to $n_t = \lceil \frac{T}{7/k \cdot 0.4} \rceil$ time steps per trajectory. To obtain an accurate reduced order model, the intermediate steps in the fractional step discretization have to be included in the snapshot set as well such that $2n_t$ discrete solution vectors have to be stored per trajectory. Thus, a total of approximately $250n_t$ snapshots has to be handled. This corresponds to roughly $250 \cdot \frac{T}{7/k \cdot 0.4} \cdot 136k^2 \approx 39000k^3$ double precision floating point numbers that have to be stored in memory. For a grid with $k = 40$, these would take about 20 gigabytes of memory, whereas for $k = 200$ about 2.5 terabytes of memory were needed.

The numerical experiments are performed on eleven compute nodes of a distributed memory computer cluster[6] utilizing 125 processor cores. In the case of the

[6]Each node encloses two Intel Xeon Westmere X5650 CPUs (2×6 cores) with 48GB RAM.



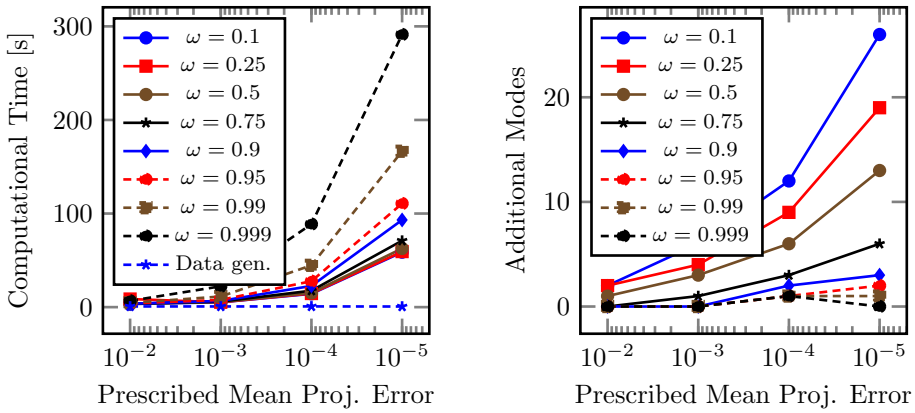
(a) HAPOD on compute node n. The time steps are split into s slices ($s = \lceil (2n_t + 1)/l \rceil$). Concurrently, each of the 12 processor cores calculates one chunk at a time, performs a POD, and sends the resulting modes to the main MPI rank on the processor. $\tau_{n,c}^t$: tth time slice on core c.

(b) An incremental HAPOD (cf. section 3.2) is performed on MPI rank 0 with the modes collected on each node. α_n: modes from node n.

FIG. 8. *HAPOD tree used for kinetic Boltzmann example (cf. section* 4.3*) on* 11 *compute nodes with* 12 *cores each.*

classical POD, each processor core calculates a solution trajectory for one parameter of the sample parameter set, after which the resulting discrete solution vectors are gathered on a single node where the POD is performed. For the HAPOD, the local PODs are calculated in parallel whenever possible. On each core a chunk of $l = 10$ time steps is calculated at a time, a POD is performed with this chunk per core, and the remaining modes are gathered per node and another POD is computed. Subsequently, the next solution chunk is calculated and compressed by a POD on each core. The resulting modes together with the modes from the first POD on node level serve as input to a second POD on node level. This is repeated until all time steps are calculated (cf. Figure 8a). The result is a set of modes on each node. Instead of gathering all modes on the main node at once, which would exceed the main node's memory, the modes are sequentially sent to the main node where additional PODs for each node are performed (cf. Figure 8b). The underlying POD algorithm is provided by `pyMOR` [34, 39], which is also used to compute and solve the resulting reduced order model. We use an optimized, incremental variant of the POD algorithm in Remark 3, which exploits the block structure of the Gramian, with the diagonal blocks being given by diagonal matrices containing the singular values of the PODs performed at the child nodes. For $k = 60$, $\omega = 0.95$, and $\epsilon^* = 10^{-4}$, this improved the overall HAPOD computation time compared to the unoptimized algorithm by 7.4% from 457 to 423 seconds.

In Figure 9, the computational time and number of HAPOD modes for different values of ω (see Theorem 10) are plotted against the prescribed ℓ^2-mean error tolerance. A 20×20 grid was used ($k = 20$, $N = 54400$). With decreasing ω, the



(a) HAPOD execution wall time for different values of ω. For all values of ω, the HAPOD is much faster than the POD, which took about 1600 seconds for each prescribed tolerance ε^*. Snapshot generation (Data gen.) took 0.8 seconds.

(b) Number of additional HAPOD modes (compared to POD) for different values of ω. The POD resulted in 2, 10, 35, and 94 modes for a prescribed error ε^* of 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5}, respectively.

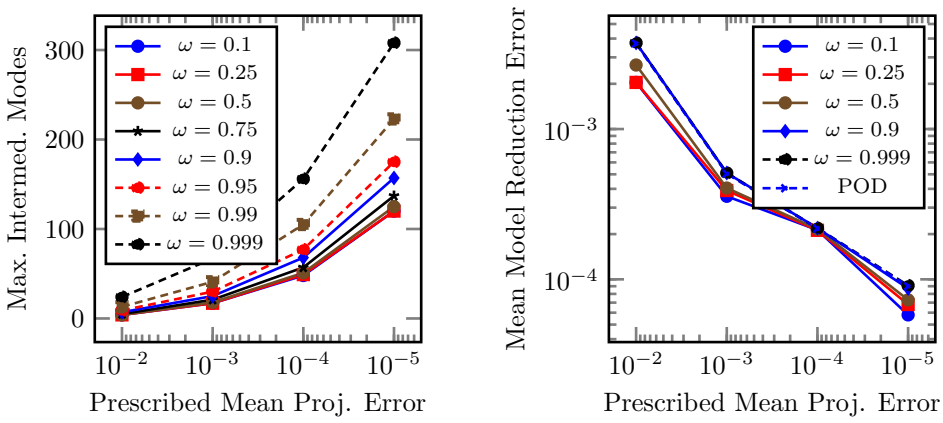
FIG. 9. *Influence of ω on HAPOD execution wall time and number of resulting modes for the kinetic Boltzmann equation example (cf. section 4.3) on a grid with $k^2 = 400$ elements ($N = 54400$ degrees of freedom).*

computational time for the HAPOD reduces but the number of final modes required to satisfy the error bound increases. Thus, choosing a larger value of ω means trading some time spent in the HAPOD for a more efficient reduced model.

Computing the classical POD takes about 1600 seconds for each tolerance. As for the previous numerical examples, the HAPOD is notably faster than the POD for all tested tolerances (see Figure 9a). Note that the HAPOD is about five times as fast as the POD, even for $\omega = 0.999$, where at most one additional final mode is obtained. The snapshot generation, i.e., the solution of the high-dimensional problem, takes only a few seconds for this grid size, so the overall computational time is dominated by the POD computation.

The maximal number of intermediate modes increases with ω (see Figure 10a). This may be important in terms of memory usage, especially if the intermediate modes are gathered in one node's memory at some time during the HAPOD. A smaller value of ω may thus be preferable if a shortage of memory is expected. Choosing $\omega = 0.95$, the number of final HAPOD modes is only slightly higher than the number of POD modes (at most two additional modes are needed), while the computation is, depending on the tolerance, at least one order of magnitude faster.

To get a measure for the model reduction error, the reduced model was solved for 1250 random combinations of $\Sigma_{s,1}$, $\Sigma_{a,1}$, $\Sigma_{a,2} \in [0, 8]$ and compared to the high-dimensional solution. For ω close to one, the resulting ℓ^2-mean error is almost equal for POD and HAPOD (see Figure 10b). For small values of ω, the model reduction error decreases slightly due to the larger number of HAPOD modes, which here result in slightly better approximation spaces than those backed by theory. Solving the reduced model takes about $5 \cdot 10^{-2}$ seconds independent of the grid size and is thus considerably faster than solving the full model, which takes up to 500 seconds on a 200×200 grid.



(a) Maximal number of intermediate HAPOD modes for different values of ω.

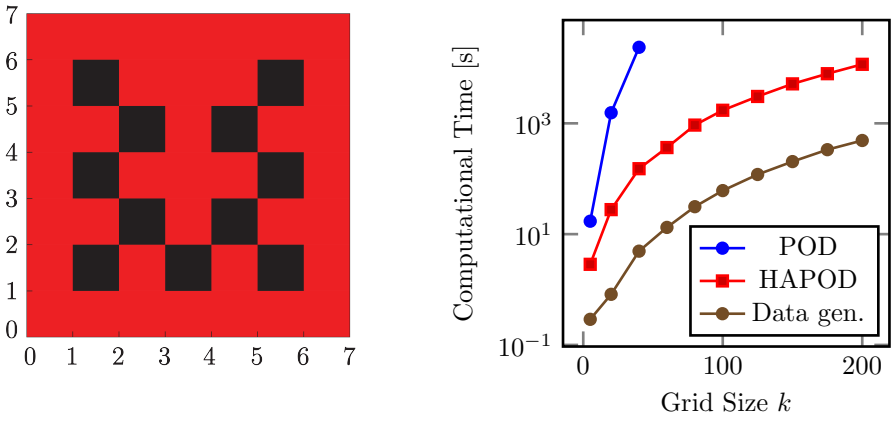
(b) ℓ^2-mean model reduction errors for 1250 random parameters, $k = 20$.

FIG. 10. *Number of local HAPOD modes and model reduction errors for the kinetic Boltzmann equation example (cf. section* 4.3*).*

The previous tests were performed on a coarse 20×20 grid. Since the memory consumption scales with k^3, refining the grid quickly leads to a situation where the snapshots do not fit in memory simultaneously such that a classical POD cannot be performed without access to mass storage. In Figure 11b, a performance comparison between POD and HAPOD ($\omega = 0.95$) for different grid sizes can be found. The HAPOD is up to two orders of magnitude faster than the POD for the coarse grids where the POD is still feasible. For $k \geq 60$, the POD fails to run due to memory limitations, while the HAPOD does not have this problem. Note that the HAPOD is twice as fast on the 200×200 grid than the classical POD on a 40×40 grid even though the amount of data that needs to be processed increases by a factor of 125 between $k = 40$ and $k = 200$. The time used for data generation plays a negligible role in the algorithm. Creating the snapshots for POD and HAPOD takes less than 10 seconds for $k = 40$ and about 500 seconds for $k = 200$. Using the HAPOD thus directly translates into a much faster overall reduced basis generation.

The final incremental PODs performed to collect the outputs of the individual compute nodes (Figure 8b) are not optimal in terms of parallelism, as all calculations are done on the main node. We thus tested another tree topology where a binary tree of nodes is built. Indeed this improved computational wall times of the HAPOD again, e.g., from 423 to 239 seconds (43% reduction) for $k = 60$, $\omega = 0.95$, $\epsilon^* = 10^{-4}$, while the memory requirements and the quality of the resulting HAPOD space were comparable.

5. Conclusion. With the HAPOD, this work introduces a general scheme for approximate POD computation that allows one to distribute the computational workload among arbitrary trees of workers, making it easily adaptable to different computing environments. Rigorous error and mode bounds are proven that ascertain the reliability and performance of the method. Specialized variants for incremental and distributed HAPOD computation are discussed, and numerical experiments underscore the applicability of the HAPOD, from small embedded devices to high performance computer clusters.



(a) Computational domain: red and black regions represent common materials.

(b) Computational wall time for POD and HAPOD ($\varepsilon^* = 10^{-4}$, $\omega = 0.95$).

FIG. 11. *Computational domain and required time for the kinetic Boltzmann equation example (cf. section 4.3). (Figure in color online.)*

Code availability. The source code used to compute the presented results is available under open source licenses and is included in the supplementary material for this publication.

REFERENCES

[1] G. W. Alldredge, C. D. Hauck, and A. L. Tits, *High-order entropy-based closures for linear transport in slab geometry* II: *A computational study of the optimization problem*, SIAM J. Sci. Comput., 34 (2012), pp. B361–B391, https://doi.org/10.1137/11084772X.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, 1999, https://doi.org/10.1137/1.9780898719604.

[3] C. Baker, K. Gallivan, and P. V. Dooren, *Low-rank incremental methods for computing dominant singular subspaces*, Linear Algebra Appl., 436 (2012), pp. 2866–2888, https://doi.org/10.1016/j.laa.2011.07.018.

[4] C. Beattie, J. Borggaard, S. Gugercin, and T. Iliescu, *A domain decomposition approach to POD*, in Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 6750–6756, https://doi.org/10.1109/CDC.2006.377642.

[5] M. Berry, D. Mezher, B. Philippe, and A. Sameh, *Parallel algorithms for the singular value decomposition*, in Handbook of Parallel Computing and Statistics, Chapman and Hall/CRC, 2005, pp. 117–164.

[6] M. Brand, *Fast online SVD revisions for lightweight recommender systems*, in Proceedings of the 2003 SIAM International Conference on Data Mining, SIAM, Philadelphia, 2003, pp. 37–46, https://doi.org/10.1137/1.9781611972733.4.

[7] M. Brand, *Fast low-rank modifications of the thin singular value decomposition*, Linear Algebra and Its Applications, 415 (2006), pp. 20–30, https://doi.org/10.1016/j.laa.2005.07.021.

[8] B. Brands, J. Mergheim, and P. Steinmann, *Reduced-order modelling for linear heat conduction with parametrised moving heat sources*, GAMM-Mitteilungen, 39 (2016), pp. 170–188, https://doi.org/10.1002/gamm.201610011.

[9] T. A. Brunner and J. P. Holloway, *Two-dimensional time dependent Riemann solvers for neutron transport*, J. Computat. Phys., 210 (2005), pp. 386–399, https://doi.org/10.1016/j.jcp.2005.04.011.

[10] J. Chen and Y. Saad, *Lanczos vectors versus singular vectors for effective dimension reduction*, IEEE Trans. Knowl. Data Eng., 21 (2009), pp. 1091–1103, https://doi.org/10.1109/TKDE.2008.228.

[11] P. Constantine and D. Gleich, *Tall and skinny QR factorizations in MapReduce architectures*, in MapReduce '11 Proceedings of the Second International Workshop on MapReduce and Its Applications, ACM, New York, 2011, pp. 43–50, https://doi.org/10.1145/1996092.1996103.

[12] P. G. Constantine, D. F. Gleich, Y. Hou, and J. Templeton, *Model reduction with MapReduce-enabled tall and skinny singular value decomposition*, SIAM J. Sci. Comput., 36 (2014), pp. S166–S199, https://doi.org/10.1137/130925219.

[13] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* II: *Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183, https://doi.org/10.1137/S0097539704442696.

[14] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU Octave Version* 4.2.1 *manual: A High-level Interactive Language for Numerical Computations*, available online at http://octave.org, 2017.

[15] K. Fernando and H. Nicholson, *On the structure of balanced and other principal representations of SISO systems*, IEEE Trans. Automat. Contr., 28 (1983), pp. 228–231, https://doi.org/10.1109/TAC.1983.1103195.

[16] M. Gubisch and S. Volkwein, *Chapter* 1*: Proper orthogonal decomposition for linear-quadratic optimal control*, in Model Reduction and Approximation: Theory and Algorithms, SIAM, Philadelphia, 2016, pp. 3–63, https://doi.org/10.1137/1.9781611974829.ch1.

[17] N. Halko, P. Martinsson, and J. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[18] C. Himpe, *emgr - The empirical Gramian framework*, Algorithms, 11 (2018), 91, https://doi.org/10.3390/a11070091.

[19] C. Himpe, *emgr - EMpirical GRamian framework (Version:* 5.1*)*, available online at http://gramian.de, 2017, https://doi.org/10.5281/zenodo.162135.

[20] C. Himpe and M. Ohlberger, *A unified software framework for empirical Gramians*, J. Math., 2013 (2013), 365909, https://doi.org/10.1155/2013/365909.

[21] C. Himpe and M. Ohlberger, *Cross-Gramian based combined state and parameter reduction for large-scale control systems*, Math. Probl. Eng., 2014 (2014), 843869, https://doi.org/10.1155/2014/843869.

[22] M. P. Holmes, J. Isbell, C. Lee, and A. G. Gray, *QUIC-SVD: Fast SVD using cosine trees*, in Advances in Neural Information Processing Systems 21, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds., Curran Associates, 2009, pp. 673–680, http://papers.nips.cc/paper/3473-quic-svd-fast-svd-using-cosine-trees.

[23] P. Holmes, J. Lumley, G. Berkooz, and C. Rowley, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge Monographs on Mechanics, Cambridge University Press, Cambridge, UK, 2012, https://doi.org/10.1017/CBO9780511919701.

[24] M. A. Iwen and B. W. Ong, *A distributed and incremental SVD algorithm for agglomerative data analysis on large networks*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1699–1718, https://doi.org/10.1137/16M1058467.

[25] H. Ji, W. Yu, and Y. Li, *A Rank Revealing Randomized Singular Value Decomposition (R3SVD) Algorithm for Low-Rank Matrix Approximations*, preprint, 2016, https://arxiv.org/abs/1605.08134, 2016.

[26] E. Jones, T. Oliphant, P. Peterson, et al., *SciPy: Open Source Scientific Tools for Python*, available online at http://www.scipy.org, 2017.

[27] K. Kunisch and S. Volkwein, *Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition*, J. Optim. Theory Appl., 102 (1999), pp. 345–371, https://doi.org/10.1023/A:1021732508059.

[28] K. Kunisch and S. Volkwein, *Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics*, SIAM J. Numer. Anal., 40 (2002), pp. 492–515, https://doi.org/10.1137/S0036142900382612.

[29] T. Leibner, *Numerical Methods for Kinetic Equations*, master's thesis, Westfälische Wilhelms-Universität Münster, Münster, Germany, 2015, https://doi.org/10.5281/zenodo.1406910.

[30] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, Basel, 1990, https://doi.org/10.1007/978-3-0348-5116-9.

[31] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge Texts Appl. Math. 31, Cambridge University Press, Cambridge, UK, 2002, https://doi.org/10.1017/CBO9780511791253.

[32] S. V. Macua, P. Belanovic, and S. Zazo, *Consensus-based distributed principal component analysis in wireless sensor networks*, in 2010 IEEE Eleventh International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2010, pp. 1–5, https:

//doi.org/10.1109/SPAWC.2010.5671089.
[33] Message Passing Interface Forum, *MPI: A Message-passing Interface Standard (version 3.1)*, http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf, 2015.
[34] R. Milk, S. Rave, and F. Schindler, *pyMOR – Generic algorithms and interfaces for model order reduction*, SIAM J. Sci. Comput., 38 (2016), pp. S194–S216, https://doi.org/10.1137/15M1026614.
[35] B. Moore, *Principal component analysis in nonlinear systems: Preliminary results*, in 18th IEEE Conference on Decision and Control Including the Symposium on Adaptive Processes, Vol. 2, 1979, pp. 1057–1060, https://doi.org/10.1109/CDC.1979.270114.
[36] MORwiki Community, *MORwiki - Model Order Reduction Wiki*, http://modelreduction.org, 2018.
[37] G. Oxberry, T. Kostova-Vassilevska, W. Arrighi, and K. Chand, *Limited-memory adaptive snapshot selection for proper orthogonal decomposition*, Internat. J. Numer. Methods Engrg., 109 (2017), pp. 198–217, https://doi.org/10.1002/nme.5283.
[38] A. Paul-Dubois-Taine and D. Amsallem, *An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models*, Internat. J. Numer. Methods Engrg., 102 (2015), pp. 1262–1292, https://doi.org/10.1002/nme.4759.
[39] pyMOR developers, *pyMOR - Model Order Reduction with Python*, https://pymor.org, 2013–2017.
[40] H. Qi, T.-W. Wang, and J. D. Birdwell, *Global principal component analysis for dimensionality reduction in distributed data mining*, in Statistical Data Mining and Knowledge Discovery, Chapman & Hall/CRC, Boca Raton, FL, 2004, pp. 327–342.
[41] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist, *Principal component analysis for dimension reduction in massive distributed data sets*, in Proceedings to the Second SIAM International Conference on Data Mining, SIAM, Philadelphia, 2002, pp. 1–12.
[42] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1100–1124, https://doi.org/10.1137/080736417.
[43] T. Sayadi, C. Hamman, and P. Schmid, *Parallel QR algorithm for data-driven decompositions*, in Center for Turbulence Research, Proceedings of the Summer Program 2014, 2014, pp. 335–343.
[44] F. Schindler, *dune-gdt*, http://github.com/dune-community/dune-gdt, 2016.
[45] F. Schneider, G. Alldredge, M. Frank, and A. Klar, *Higher order mixed moment approximations for the Fokker–Planck equation in one space dimension*, SIAM J. Appl. Math., 74 (2014), pp. 1087–1114, https://doi.org/10.1137/130934210.
[46] L. Sirovich, *Turbulence and the dynamics of coherent structures part* I: *Coherent structures*, Quart. Appl. Math., 45 (1987), pp. 561–571, http://www.jstor.org/stable/43637457.
[47] S. Solovyev and S. Tordeux, *Compute SVD of a very large matrix in the context of geological prospection*, in 6th EAGE Saint Petersburg International Conference and Exhibition, 2014, https://doi.org/10.3997/2214-4609.20140190.
[48] S. Solovyev and S. Tordeux, *Large SVD computations for analysis of inverse problems in geophysics*, in Proceedings of the WCCM XI - ECCM V - ECFD VI, 2014, pp. 2861–2869, http://congress.cimne.com/iacm-eccomas2014/admin/files/filePaper/p2861.pdf.
[49] D. Sorensen and A. Antoulas, *The Sylvester equation and approximate balanced reduction*, Linear Algebra Appl., 351–352 (2002), pp. 671–700, https://doi.org/10.1016/S0024-3795(02)00283-5.
[50] Z. Wang, B. McBee, and T. Iliescu, *Approximate partitioned method of snapshots for POD*, J. Comput. Appl. Math., 307 (2016), pp. 374–384, https://doi.org/10.1016/j.cam.2015.11.023.