Expedient and Parallelizable Sparse Coding Algorithm for Large Datasets

Zongxian Liang, [*] Rohit Deshmukh, [†] and Jack J. McNamara [‡]
Department of Mechanical and Aerospace Engineering
The Ohio State University, Columbus, OH, 43210, USA

Matt Wytock, [§] and J. Zico Kolter [¶]
School of Computer Science
Carnegie Mellon University, Pittsburgh, PA, 15213, USA

Recently, sparse coding has been identified as a promising alternative approach to Proper Orthogonal Decomposition for the basis identification task in Galerkin based fluid models. However, significant limitations of the approach are high memory requirements and increased computational cost relative to Proper Orthogonal Decomposition. This paper details the development of an improved sparse coding algorithm that enables the use of high performance parallel computing environments to distribute memory requirements and expedite the basis extraction process. The algorithm is demonstrated on three different datasets ranging from small (2-D lid-driven cavity) to medium (ship airwake, 3-D reversed flow over an airfoil) in size. Performance is assessed in terms of computational cost and quality of bases extracted from the data. The new algorithm is found to enable the extraction of sparse bases from at least terabyte scale datasets, a necessary component to bridging high fidelity flow simulation with systems level and multi-disciplinary analysis.

Nomenclature

$[a_i]$	= Galerkin matrix, constant terms
$[b_{ij}]$	= Galerkin matrix, linear terms
$[c_{ijk}]$	= Galerkin matrix, quadratic terms
$[d_{ij}]$	= correlation matrix of sparse modes
$a_{i,j}$	= entries of correlation matrix of snapshots
b_j^i	= the j^{th} element of the i^{th} eigenvector
$f_{obj}^{(k)}$	= objective function
A	= correlation matrix of snapshots
C	= correlation matrix of sparse modes and snapshots
C_{POD}	= Computational complexity of POD method
C_{SPC}	= Computational complexity of sparse coding method
G	= correlation matrix of sparse modes
L	= active (non-zero) number of modes in a snapshot matrix
m	= number of snapshots in a snapshot matrix
mem_{POD}	= minimum memory required for POD method

[*] Post-Doctoral Researcher, liang.593@osu.edu, AIAA Member.
[†] Ph.D. Student, deshmukh.17@osu.edu, AIAA Student Member.
[‡] Associate Professor, mcnamara.190@osu.edu, AIAA Associate Fellow.
[§] Ph.D. Student, mwytock@cs.cmu.edu.
[¶] Assistant Professor, zkolter@cs.cmu.edu.

mem_{SPC} = minimum memory required for sparse coding method
mem_{node} = physical memory of a computational node
N = number of modes in a dictionary
n_g = number of grid points
n_{dim} = 2 for two-dimensional flows, and 3 for three-dimensional flows.
n_{node} = number of nodes
n_{node}^{POD} = minimum number of nodes required for POD method
n_{node}^{SPC} = minimum number of nodes required for sparse coding method
n_{node} = number of nodes
n_p = number of processors
n_{pcol} = number of processors in the column of processor grid
n_{prow} = number of processors in the row of processor grid
n_{dim} = 2 for two-dimensional flows, and 3 for three-dimensional flows.
n = number of variables in a snapshot vector, equal to $n_g \times n_{dim}$
p = 4 for single precision floating number, and 8 for double precision
$\boldsymbol{Q}$ = snapshot matrix
$\boldsymbol{q}$ = fluctuating component of flows
r = rank of matrix
Re = Reynolds number
S = sparsity of matrix
$\boldsymbol{S}$ = coefficient matrix, $\in \mathbb{R}^{N \times m}$
$\boldsymbol{S'}$ = non-sparse coefficient matrix
s^i = i^{th} coefficient corresponding to the i^{th} mode at any time instant
$\boldsymbol{s_k}$ = k^{th} column of the coefficient matrix
t = non-dimensional time
$\bar{\boldsymbol{U}}$ = mean velocity component
$\tilde{\boldsymbol{u}}$ = prediction for the fluctuating component of the flow
$\boldsymbol{V}$ = eigenvectors
V_Q = data volume of datasets
$\boldsymbol{x}$ = position vector
$\boldsymbol{X}$ = modal matrix containing modes in its columns
$\boldsymbol{X}_i$ = i^{th} mode vector
$\boldsymbol{Y}$ = POD coefficient matrix, $\in \mathbb{R}^{r \times m}$
$\boldsymbol{y}_i$ = i^{th} column of POD coefficient matrix

β = sparsity coefficient
ϵ = convergence criterion
λ = eigenvalues
$\boldsymbol{\Lambda}$ = eigenvalue matrix
Ω = computational domain
$\boldsymbol{\Phi}$ = modal matrix containing modes in its columns
$\boldsymbol{\Phi}_i$ = i^{th} mode vector
$\boldsymbol{\Phi}^{POD}$ = POD modal matrix, $\in \mathbb{R}^{n \times r}$
$\boldsymbol{\Phi}^{SPC}$ = sparse modal matrix, $\in \mathbb{R}^{n \times N}$
∇ = the gradient operator
$\|\cdot\|$ = L2 norm, equal to the sum of squares of the entries in $(\cdot)$
$\|\cdot\|_0$ = L0 norm, equal to the number of non-zero entries in $(\cdot)$
$\|\cdot\|_{1,1}$ = L1 norm, equal to the sum of the absolute values of entries in $(\cdot)$
$\|\cdot\|_F$ = Frobenius norm for matrix, equal to the sum of squares of the entries in $(\cdot)$

Supscripts
T = Transpose of matrix
-1 = Moore-Penrose pseudo-inverse of matrix
(k) = k^{th} iteration

Subscripts
W = weights

I. Introduction and Problem Statement

The advancement of computational fluid dynamics (CFD), parallel computing algorithms, and computing hardware, has enabled unprecedented insight into complex flow physics through numerical analysis. However, the computational expense associated with such analysis tools has restricted their application to relatively small spatial and temporal scales on simplistic configurations. Thus, the advances in CFD methodologies have not yet enabled the broad consideration of nonlinear, multi-scale, unsteady flows, in systems level studies - e.g. fluid-structure interaction, flow control, aerodynamic design, structural design, etc. Yet the critical need for high fidelity flow modeling in such problems motivates the pursuit of tractable and robust reduced order models (ROMs).

A common ROM approach is to project the governing equations onto a reduced dimensional space comprised of characteristic bases.[1–6] These can be based on orthogonal (e.g., Galerkin) or non-orthogonal (e.g., Petrov-Galerkin) projections.[1–6] However, the accuracy of such approaches is intimately bound to the quality of the chosen bases. This is challenging for nonlinear problems, due to the need to carry out basis identification from post execution data of representative system dynamics. In the context of highly unsteady nonlinear flows, this is further complicated by the fact that generally: 1) the gathering of the data is computationally expensive, 2) the data is very large and high dimensional, and 3) the scope of the data is narrow. Thus, identifying a compact set of prominent and dynamically important flow features to fundamentally characterize the fluid dynamics is a non-trivial problem.

Proper Orthogonal Decomposition (POD), or Principal Components Analysis (PCA),[1,7–10] is a widely used and explored technique aimed at meeting this need. The approach is based on identifying and ordering principal components in observed data. The POD modes are optimal in terms of representing the energy of an observed flow response, thus a reduced dimensional basis of the system is often identified by truncating the modes based on energy contribution to the response.[1,7–10] However, there are several issues with this approach. First, the POD modes are only optimal in the sense of reconstructing the observed flow responses.[1,11,12] Thus, they may not generalize well for model predictions that deviate from observed conditions. From a fluid physics perspective, a truncated set of POD modes is biased towards the high-energy, large-scale, dominant structures and ignores the small-scale, low-energy structures.[1,11,13] The large-scale structures are formed as a result of disturbances in the flow- obtaining energy from the mean flow, and then subsequently breaking down into smaller scales.[14] The small-scale structures then cause energy dissipation and result in viscosity in the fluid-flow. Thus, POD based Galerkin ROMs do not account for sufficient energy dissipation, resulting in over-prediction of kinetic energy.[1,11,13] Moreover, the energy accumulation over a period of time may also cause the ROM to become unstable.[11,13,15] Related to this, it is now well-established that POD modes are ineffective in capturing the local dynamics (or transience) of full-order systems.[11,12] This is because the POD modes are always active, or have nonzero coefficients for all time windows. For these reasons it is clear that the optimality of POD modes, in terms of energy capture, is non-ideal for model reduction of highly unsteady, nonlinear flow fields.

Techniques such as Balanced Truncation,[16] Balanced POD (BPOD),[2] and Eigensystem Realization Algorithm (ERA)[17] have addressed some of the limitations identified earlier. However, balanced truncation is intractable for large data (for more than 10,000 degrees of freedom),[3] BPOD is only applicable to response data of linear systems as it requires adjoint system information,[2,17] and modes generated by ERA cannot be used for projection of non-linear dynamics.[17] In a recent study, a technique is developed to generate a stable Galerkin projection based ROM.[11] However, building the ROM is an iterative process, and requires multiple time-integrations until an energy-balance is achieved. These issues highlight the need to explore alternative basis identification techniques that not only generalize well to changing flow conditions, but also accurately capture essential multi-scale features.

In ongoing work by the authors,[18,19] a new approach for basis identification in Galerkin-based ROMs was examined, known as sparse coding. This approach was discussed in early work by Oshausen and Field,[20] who argued that most naturally occurring phenomena are not well represented by the PCA approach, but more appropriately by *sparse codes*; i.e., a large dictionary of bases where only a few are active

over any given time window. Compared to the POD approach - where the principal components of the observed data are identified, ordered and then truncated into a compact set - sparse coding is formulated as an optimization problem to identify a compact representation to approximately span the entire dataset. Hence, dimensionality reduction is carried out simultaneously to the factorization. This leads to a set of modes that are spatially multi-scale and that tend to capture the local transience of any given time window.[19] Comparisons of Galerkin ROMs generated using POD and sparse bases for turbulent flow in a lid driven cavity indicated that the sparse modes yield reasonably accurate predictions, with significantly fewer modes, for quantities such as turbulent kinetic energy and second order statistics of fluctuating velocities.[18,19] However, a drawback of the sparse coding approach is the significant increase in computational cost in the basis identification step. This issue limits the application, using the existing algorithm for sparse coding, to relatively simple, low-dimensional systems. Motivated by this, the present paper describes the development of an alternative approach that is parallelizable and able to handle very large datasets using high performance computing clusters.

The remainder of this paper is organized as follows. The POD and sparse coding approaches are presented in Section II. Results describing the application of POD and sparse modes to model the unsteady flow fields are presented in Section IV. Concluding remarks are provided in Section V.

II. Methods

The datasets analyzed represent snapshots of Navier-Stokes solutions for flow in a lid-driven cavity, reversed flow over an airfoil, and ship airwake. The lid-driven cavity data is computed using an in-house solver,[21] which directly simulates the non-dimensionalized Navier-Stokes equations with a second-order central difference spatial scheme and a second-order fractional-step method for time marching. The ship airwake is simulated using the commercial CFD code, ANSYS FLUENT, which solves the Navier-Stokes equations using the finite volume method with a turbulent model of Detached Eddy Simulation (DES). The reversed airfoil data is generated using the LES code FDL3DI which computes the compressible Navier-Stokes equations using a sixth-order compact difference method and an implicit second-order approximate-factorization method for time integration.[22,23]

II.A. Generation of Reduced Order Modes

The first step in the basis extraction process is to decompose the full order solution into mean ($\bar{\boldsymbol{U}}(\mathbf{x})$) and fluctuating components ($\boldsymbol{q}(\boldsymbol{x},t)$) of velocity. The mean velocity $\bar{\boldsymbol{U}}(\boldsymbol{x})$ is evaluated as time average of flow-field variables. The fluctuating component ($\boldsymbol{q}(\boldsymbol{x},t)$) is approximated as a linear combination of reduced order modes:

$$\boldsymbol{q}(\boldsymbol{x},t) \approx \sum_{i=1}^{N} s^i(t)\boldsymbol{\Phi}_i(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{\Phi_i}$ are modes, $s^i(t)$ are the temporal coefficient corresponding to $\boldsymbol{\Phi_i}$, and N is the number of modes. Eq. 1 can be written in a matrix form:

$$\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_m] \approx \boldsymbol{\Phi S}, \tag{2}$$

where $\boldsymbol{q}_i = \left[u_1^i, v_1^i, w_1^i, \cdots, u_{n_g}^i, v_{n_g}^i, w_{n_g}^i\right]^T$ represents the snapshots, $\boldsymbol{\Phi} = [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \cdots, \boldsymbol{\Phi}_N]$ is the matrix of modes, $\boldsymbol{S} \in \mathbb{R}^{N\times m}$ is the coefficient matrix and m is the number of snapshots for n_g grid points in the computational domain. The number of variables in one snapshot is $n = n_{dim} \times n_g$, where n_{dim} is 2 for two-dimensions, and 3 for three-dimensions.

As discussed, basis extraction from these datasets, for the purposes of constructing a Galerkin ROM, is carried out using POD and sparse coding. The POD modes are computed using the method of snapshots developed by Sirovich.[10] Sparse modes are evaluated using an algorithm developed in this work. These approaches are discussed below.

II.B. Basis Extraction Using Proper Orthogonal Decomposition

The method of snapshots[10] is used to extract a set of POD modes $\boldsymbol{\Phi}_i$ from the snapshot matrix $\boldsymbol{Q}$. It solves for the solution of the following eigenvalue problem:

$$\boldsymbol{A}\boldsymbol{V} = \lambda \boldsymbol{V} \tag{3}$$

where the entries of the correlation matrix $\boldsymbol{A}$ are $a_{i,j} = (\boldsymbol{q}_i, \boldsymbol{q}_j)$, and the rank of $\boldsymbol{A}$ is r. Here, $(\boldsymbol{f}, \boldsymbol{g})$ is an inner product for the space $L^2(\Omega)$ of square-integrable functions, defined as

$$(\boldsymbol{f}, \boldsymbol{g}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{g} d\Omega \tag{4}$$

The first r POD modes $\boldsymbol{\Phi}_i$ are given by:

$$\boldsymbol{\Phi}_i = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{m} b_j^i \boldsymbol{q}_j \tag{5}$$

where λ_i are the eigenvalues of the correlation matrix $\boldsymbol{A}$ in a descending order, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq 0$, and b_j^i is the jth element of the ith eigenvector $\boldsymbol{V}_i$. Note that $\sqrt{\lambda_i}$ are the eigenvalues of the snapshot matrix $\boldsymbol{Q}$.

The temporal coefficients $\boldsymbol{S}$ are given by

$$\boldsymbol{S}_i^j = (\boldsymbol{\Phi}_i, \boldsymbol{q}_j) \tag{6}$$

II.C. Basis Extraction Using Sparse Coding

In our previous work,[18,19] we solve the following minimization problem to obtain a set of sparse modes $\boldsymbol{\Phi}$:

$$\min_{\boldsymbol{S},\boldsymbol{\Phi}} \quad \frac{1}{2} \sum_{i=1}^{m} \left(\|\boldsymbol{q}_i - \boldsymbol{\Phi}\boldsymbol{s}_i\|_F^2 + \beta \|\boldsymbol{s}_i\|_1 \right), \quad \text{subject to} \quad \forall i, \|\boldsymbol{\Phi}_i\| \leq 1 \tag{7}$$

where $\boldsymbol{s}_i \in \mathbb{R}^N$ are the column vectors of the matrix $\boldsymbol{S}$, $\|\cdot\|_F$ is the Frobenius norm, and the L1 penalty term $\|\boldsymbol{s}_i\|_1$ induces sparsity in $\boldsymbol{s}_i$. An important factor in the generation of the sparse basis is the sparsity level of the coefficient matrix, where sparsity of a matrix is the fraction of zero elements over the total number of elements in the matrix:

$$S = \frac{\text{Number of zero elements in } \boldsymbol{S}}{N \times m} \times 100\% \tag{8}$$

A coefficient matrix with a low sparsity is associated with modes that do not capture the local dynamics (transience) of the full order system. In contrast, a highly sparse coefficient matrix may result in modes that provide a poor representation of structure in the observed data. Thus, an 'optimal' range of sparsity exists to capture both the structure and local dynamics in the observed data. In limited results carried out to date,[18,19] sparsity of 50-80% is observed to yield stable and reasonably accurate ROMs. Note that in Eq. 7, the sparsity of the basis is controlled using the sparsity parameter β. However, the exact specification of the sparsity is not possible using Eq. 7, rather the number of active bases in a factorization is determined through an iterative process of varying the sparsity parameter $0 < \beta < 0.5$.[24]

An alternative sparse coding approach does permit the direct specification of sparsity, and thus is adopted in this work:[25,26]

$$\min_{\boldsymbol{S},\boldsymbol{\Phi}} \quad \|\boldsymbol{Q} - \boldsymbol{\Phi}\boldsymbol{S}\|_F^2, \quad \text{subject to} \quad \forall i, \|\boldsymbol{s}_i\|_0 \leq L \tag{9}$$

Here, the L0 norm $\|.\|_0$ counts the number of non-zero entries in $\boldsymbol{s}_i$. By enforcing $\|\boldsymbol{s}_i\|_0 \leq L$ in Eq. 9, the number of non-zero entries in $\boldsymbol{s}_i$ is constrained to be less than or equal to a constant L. Sparsity in this context is determined as

$$S = \frac{(N - L) \times m}{N \times m} \times 100\% = (1 - \frac{L}{N}) \times 100\% \tag{10}$$

where L is the active number of basis vectors, out of N total, in any given snapshot. Thus using Eq. 9, the sparsity and dimension of the ROM is directly controlled by specifying L and N, respectively.

Solving Eq. 9 is challenging since its solution is non-convex.[24] Instead, following standard sparse coding procedures, we alternatively minimize over the matrices $\boldsymbol{S}$ and $\boldsymbol{\Phi}$:

$$\min_{\boldsymbol{s}_i} \quad \|\boldsymbol{q}_i - \boldsymbol{\Phi}\boldsymbol{s}_i\|_2^2, \quad \text{subject to} \quad \forall i, \|\boldsymbol{s}_i\|_0 \leq L \tag{11}$$

and

$$\min_{\boldsymbol{\Phi}} \quad \|\boldsymbol{Q} - \boldsymbol{\Phi}\boldsymbol{S}\|_F^2, \quad \text{subject to} \quad \forall i, \|\boldsymbol{\Phi}_i\| \leq 1 \tag{12}$$

Equation 11 determines a sparse representation of the each snapshot based on the current sparse modes. Greedy algorithms such as Matching Pursuit (MP),[27] Orthogonal Matching Pursuit (OMP),[28–30] and batch-OMP[26] algorithms can be used to learn $\boldsymbol{s}_i$. Equation 12 is an optimization over sparse modes across the whole snapshot matrix $\boldsymbol{Q}$ at once. It is a constrained least-squares problem that can be solved by using the Method of Optimal Directions (MOD) approach[31] and the K-SVD algorithm.[25]

While this alternative sparse coding procedure enables improved control over the properties of the sparse modes, it does not address the issue of high computational overhead for the sparse coding procedure of basis identification. To solve this issue, the dimensionality of the sparse coding problem is recast by first factorizing the snapshot matrix into POD modes and temporal coefficients. However, contrary to the standard POD implementation, all POD modes are retained so that all features in the snapshot matrix are retained. Next, sparse coding is performed on the temporal coefficient matrix, which is significantly lower dimension than the snapshot data. Finally, the sparse bases are constructed by multiplying the sparsified temporal coefficients by the full set of POD modes. The exact details, schematically depicted in Fig. 1, are described next:

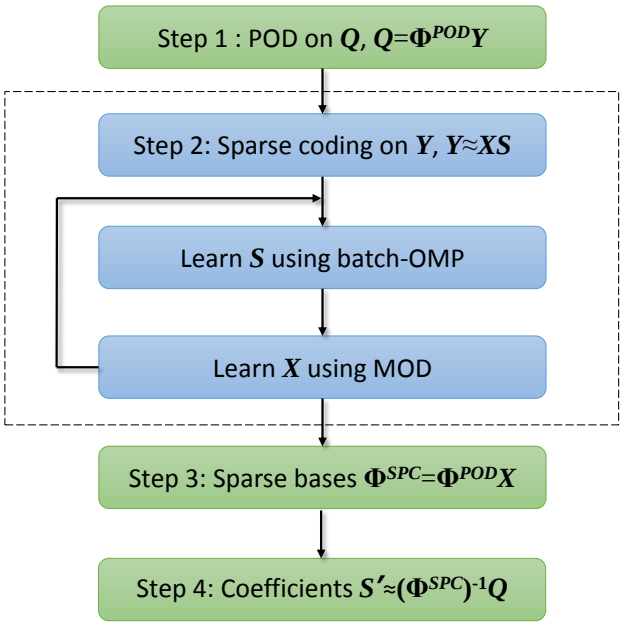


Figure 1. Flow chart of the algorithm

1. Compute the POD decomposition of $\boldsymbol{Q}$ using the method of snapshots.[10] This yields

$$\boldsymbol{Q} = \boldsymbol{\Phi}^{POD}\boldsymbol{Y} \tag{13}$$

where $\boldsymbol{Y} \in \mathbb{R}^{r\times m}$ is the POD coefficient matrix, and r denotes the rank of $\boldsymbol{Q}$ and $\boldsymbol{\Phi}^{POD}$. The rank r is less than the number of snapshots m if the snapshot matrix $\boldsymbol{Q}$ is rank-deficient. However, all features in the snapshot matrix are retained if r modes are used in sparse coding.

2. Given the POD coefficient matrix $\boldsymbol{Y}$, solve for a set of sparse modes $\boldsymbol{X} \in \mathbb{R}^{r\times N}$ using sparse coding. The modes $\boldsymbol{X}$ and corresponding coefficient matrix $\boldsymbol{S}$ are updated iteratively as explained earlier. The

sparse coding process is stopped once the modes are converged, i.e.,

$$\left| f_{obj}^{(k)} - f_{obj}^{(k-1)} \right| < \epsilon \tag{14}$$

where

$$f_{obj}^{(k)} = \left\| \boldsymbol{Y} - \boldsymbol{X}^{(k)} \boldsymbol{S}^{(k)} \right\|_F^2 \tag{15}$$

Here, the superscript k denotes the k^{th} iteration and ϵ is a pre-defined convergence criterion, set in this work as $\epsilon = 10^{-3}$. Due to the non-convexity of the overall minimization problem, numerical experimentation using random initial guesses for the sparse modes was found in a high percentage of cases to yield locally optimal solutions that produced poor results. In order to avoid this, the initial value of the sparse modes is computed as normalized column-vectors of the POD coefficient matrix:

$$\boldsymbol{x}_i = \frac{\boldsymbol{y}_j}{\left\| \boldsymbol{y}_j \right\|_2}, \quad i = 1, 2, ..., N, \quad j = \frac{m}{N} * (i-1) + 1 \tag{16}$$

where $\boldsymbol{\psi}_i$ is the initial value for the i^{th} sparse mode, $\boldsymbol{y}_j$ is the j^{th} column of the POD coefficient matrix. This step results in a sparse representation of $\boldsymbol{Y}$ as

$$\boldsymbol{Y} \approx \boldsymbol{X}\boldsymbol{S} \tag{17}$$

3. The sparse modes of the snapshot matrix are given by

$$\boldsymbol{\Phi}^{SPC} = \boldsymbol{\Phi}^{POD} \boldsymbol{X} \tag{18}$$

4. Compute the non-sparse representation of snapshots by projecting the snapshot matrix $\boldsymbol{Q}$ onto the sparse modal matrix $\boldsymbol{\Phi}^{SPC}$

$$\boldsymbol{S}' \approx (\boldsymbol{\Phi}^{SPC})^{-1} \boldsymbol{Q} \tag{19}$$

In this step, a QR factorization of $\boldsymbol{\Phi}^{SPC}$ is employed to find the least squares solution that satisfies

$$\min_{\boldsymbol{s}_i'} \quad \left\| \boldsymbol{q}_i - \boldsymbol{\Phi}^{SPC} \boldsymbol{s}_i' \right\|_2^2 \tag{20}$$

The non-sparse representation $\boldsymbol{S}'$ can be used to evaluate of reconstruction errors, or as initial values of sparse-ROMs.

The batch-OMP method and a modified MOD method are used in this study for solving sparse modes $\boldsymbol{X}$. The batch-OMP method is described in Ref. 26. The modified MOD method is to multiply $\boldsymbol{Y}$ with $\boldsymbol{S}^{-1}$, where $\boldsymbol{S}^{-1}$ is the Moore-Penrose pseudo-inverse of the matrix $\boldsymbol{S}$ and is computed using the SVD method. The sparse modes $\boldsymbol{X}_i$ are then normalized such that $\|\boldsymbol{X}_i\| = 1$. Note that this method does not directly solve the constrained least-squares problem. Instead, it solves a least-squares problem followed by enforcing the L2 constraint.

II.D. Parallel Implementation

Parallel computing is necessary for efficiently processing high-dimensional data since a large amount of memory is required in Steps 1, 3 and 4 of the algorithm. This also enables a reduction in computational cost.

Our parallel implementation is based on the linear algebra library ScaLAPACK.[32] All matrices involved in the computing are partitioned onto a two-dimensional, $n_{prow} \times n_{pcol}$ process grid using two-dimensional block-cyclic memory distribution.[32] Here, n_{prow} and n_{pcol} are the numbers of processors in row and column, respectively, and the total number of processors $n_p = n_{prow} \times n_{pcol}$. In practice, Step 1 is executed as one independent procedure, whereas Steps 2, 3 and 4 as another. This enables the reuse of POD modes generated in Step 1 by multiple sparse coding executions. Additionally, it provides more flexibility in allocating computational resources for different steps.

The parallelization of the POD method is described as follows. First, one process reads flow data and distributes the data to the other $n_p - 1$ processors. Next, the correlation matrix $\boldsymbol{A}$ is evaluated. One can simply compute the entries of the matrix $\boldsymbol{A}$ using weighted inner product of two snapshots, where the weights are the cell volume of grids. Alternatively, one can solve for

$$\boldsymbol{A} = \int_{\Omega} \boldsymbol{Q}^T \boldsymbol{Q} d\Omega = \boldsymbol{Q}_W^T \boldsymbol{Q} \tag{21}$$

using matrix-matrix multiplication, where the subscript W denotes weights. Compared to matrix-multiplication method, the inner-product method requires less memory, whereas the matrix-multiplication method uses faster algorithms and is more computationally efficient. In our numerical experiments, it is found that the matrix-multiplication method is approximately 12 times faster than the inner-product method for a snapshot matrix $\boldsymbol{Q}$ of $10^5 \times 10^3$ on a single core of an Intel Xeon X5650 @2.67GHz processor. The speed up is higher for larger matrices. Therefore, the matrix-multiplication method is used in the current study at a cost of relatively high memory.

Similarly, POD modes are generated by solving

$$\boldsymbol{\Phi} = \boldsymbol{Q}\boldsymbol{V} \tag{22}$$

and normalizing

$$\boldsymbol{\Phi}_i = \frac{\boldsymbol{\Phi}_i}{(\boldsymbol{\Phi}_{Wi}, \boldsymbol{\Phi}_i)} \tag{23}$$

The corresponding coefficients of POD modes are computed by

$$\boldsymbol{Y} = \boldsymbol{\Phi}_W^T \boldsymbol{Q}, \tag{24}$$

The batch-OMP and MOD methods are parallelized for sparse coding. The batch-OMP method can be naturally executed in parallel since it solves Eq. 11 with respect to different snapshots. Specifically, the dictionary matrix $\boldsymbol{X}$ in Eq. 17 is duplicated to n_p processors, whereas the POD coefficient matrix $\boldsymbol{Y}$ is re-distributed to n_p processors using one-dimensional block-cyclic distribution.[32] At each processor, the evaluation of column vectors $\boldsymbol{s}_i$ is conducted with respect to the corresponding $\boldsymbol{y}_i$ assigned to that processor. The sparse matrix $\boldsymbol{S}$ are formed by collecting $\boldsymbol{s}_i$ from different processors after the batch-OMP method completes. Parallelizing the MOD method is carried out using matrix-multiplication and SVD subroutines in ScaLAPACK.

II.E. Time Complexity and Memory Requirement

Time complexity of an algorithm provides a theoretical estimation of the execution time for an algorithm with different sizes of input data. Big O notation[33] is used in the current study to estimate the growth rate of run-time of an algorithm. It provides an upper bound on the growth rate of run-time, i.e. for an algorithm with $O(n)$ complexity, the run-time of the algorithm is less than $c \times f(n)$, where c is some arbitrary positive constant, n is an input size and f is the function expressing the growth rate of run-time.

The computational complexity of the equations used in the POD and sparse coding algorithms is listed in Table 1. The total complexity of POD is

$$C_{POD} = O(m^3 + nm + nm^2 + Nnm + Nn). \tag{25}$$

For high-dimensional fluid datasets, n is several orders of magnitude larger than m, m and r have the same order of magnitude, N is one order of magnitude smaller than m, L is two order of magnitude smaller than m. Thus, after omitting the quadratic terms and the cubic term m^3, the complexity of POD approximates to

$$C_{POD} \approx O(nm^2 + Nnm), \tag{26}$$

Similarly, the complexity of sparse coding is

$$C_{SPC} = O(rN^2 + (2rN + L^2N + 3LN + L^3)m + m^3 + nrN + nm^2) \approx O(nm^2 + Nnr), \tag{27}$$

since the last two terms are significantly larger than the other. Note that since m and r are of the same magnitude, the complexity of the new sparse coding algorithm is comparable to that of the POD method.

Note that performing sparse coding on the flow data without dimensionality reduction requires

$$O(nN^2 + (2nN + L^2N + 3LN + L^3)m + n^3 + nm^2) \approx O(n^3) \tag{28}$$

operations. Since n is several orders of magnitude larger than m, the computational cost without dimensionality reduction is significantly higher.

The memory usage of the POD and sparse coding algorithms is determined by the largest memory usage among the operations. Here, the data volume of the dataset V_Q is used as reference for the memory requirement. The data volume of the dataset V_Q is nmp bytes, where p equals 4 for single precision floating number and 8 for double precision. The minimum memory requirement for the POD process is

$$mem_{POD} = (2nm + m^2)p \text{ bytes} \approx 2V_Q \tag{29}$$

and it requires

$$n_{node}^{POD} = \frac{(2nm + m^2)p}{mem_{node}} \approx \frac{2V_Q}{mem_{node}} \tag{30}$$

computational nodes with mem_{node} bytes of memory per node.

Similarly, the sparse coding algorithm requires

$$mem_{SPC} = (2nm + 2nN + Nr)p \text{ bytes} \tag{31}$$

Depending on the number of sparse modes to be computed, the minimum memory of the sparse coding algorithm varies:

$$2V_Q < mem_{SPC} < 4V_Q \tag{32}$$

And the node requirement varies accordingly:

$$\frac{2V_Q}{mem_{node}} < n_{node}^{SPC} < \frac{4V_Q}{mem_{node}} \tag{33}$$

For a dataset with data volume of 10TB, computing POD modes requires a minimum of 417 nodes with 48GB memory per node. Furthermore, 417 to 834 nodes are required to compute sparse modes. Based on this estimation, datasets at O(10TB) could be processed by the new algorithm on clusters with thousands of nodes and memory larger than 48GB per node.

Table 1. Computational complexity order of the POD and sparse coding algorithms

	Operation	Order of Complexity[26,33]	Memory (bytes)
POD	$\boldsymbol{AV} = \lambda \boldsymbol{V}$	m^3	$3m^2p$
	$\boldsymbol{A} = \boldsymbol{Q}_W^T \boldsymbol{Q}$	$nm + nm^2$	$(2nm + m^2)p$
	$\boldsymbol{\Phi} = \boldsymbol{QV}$	Nnm	$(nm + mr + nr)p$
	$\boldsymbol{\Phi}_i = \frac{\boldsymbol{\Phi}_i}{(\boldsymbol{\Phi}_{Wi}, \boldsymbol{\Phi}_i)}$	Nn	$(n + nr)p$
	$\boldsymbol{Y} = \boldsymbol{\Phi}_W^T \boldsymbol{Q}$	$Nn + Nnm$	$(nm + mr + nr)p$
Sparse	$\min_{\boldsymbol{s}_i} \ \|\boldsymbol{y}_i - \boldsymbol{X}\boldsymbol{s}_i\|_2^2$	$rN^2 + (2rN + L^2N + 3LN + L^3)m$	$(rN + 2NL + N^2 + Nm + L^2)p$
	$\min_{\boldsymbol{X}} \ \|\boldsymbol{Y} - \boldsymbol{XS}\|_F^2$	m^3	$(rm + rN + N^2 + m^2 + mN)p$
	$\boldsymbol{\Phi}^{SPC} = \boldsymbol{\Phi}^{POD}\boldsymbol{X}$	nrN	$(nN + nr + rN)p$
	$\min_{\boldsymbol{s}'_i} \ \left\|\boldsymbol{q}_i - \boldsymbol{\Phi}^{SPC}\boldsymbol{s}'_i\right\|_2^2$	nm^2	$(2nm + 2nN + Nr)p$

II.F. Galerkin Projection

A reduced order solution to the unsteady, incompressible fluid system is obtained by computing the time histories of the modal weights (also called prediction coefficients) using a Galerkin projection framework,[1–4] in which the governing partial differential equations are projected onto the space spanned by a set of basis functions to yield a system of ordinary differential equations. Specifically, the mean and the expansion

in Eq. 1 are substituted into the governing Navier-Stokes equations. The residual term is assumed to be orthogonal to the space spanned by the modes:

$$\left(\boldsymbol{\Phi}_i, \frac{\partial(\bar{\boldsymbol{U}}+\boldsymbol{q})}{\partial t} + (\bar{\boldsymbol{U}}+\boldsymbol{q})\cdot\boldsymbol{\nabla})\,(\bar{\boldsymbol{U}}+\boldsymbol{q})\right) = \frac{1}{Re}\left(\boldsymbol{\Phi}_i, \boldsymbol{\nabla}^2(\bar{\boldsymbol{U}}+\boldsymbol{q})\right) \tag{34}$$

where $i = 1, 2, \cdots, N$, and $\boldsymbol{q}$ is expressed as a linear combination of the reduced order modes, as shown in Eq. 1. This yields the following set of evolution equations for the mode amplitudes $s^i(t)$:

$$\begin{bmatrix} \frac{ds^1(t)}{dt} \\ \vdots \\ \frac{ds^i(t)}{dt} \\ \vdots \\ \frac{ds^N(t)}{dt} \end{bmatrix} = [d_{ij}]^{-1} \begin{bmatrix} a_1 + \sum_{j=1}^N b_{1j}s^j(t) + \sum_{j=1}^N\sum_{k=1}^N c_{1jk}s^j(t)s^k(t) \\ \vdots \\ a_i + \sum_{j=1}^N b_{ij}s^j(t) + \sum_{j=1}^N\sum_{k=1}^N c_{ijk}s^j(t)s^k(t) \\ \vdots \\ a_N + \sum_{j=1}^N b_{Nj}s^j(t) + \sum_{j=1}^N\sum_{k=1}^N c_{Njk}s^j(t)s^k(t) \end{bmatrix} \tag{35}$$

where the coefficients are given as:

$$a_i = -\left(\boldsymbol{\Phi_i}, (\bar{\boldsymbol{U}}.\boldsymbol{\nabla})\,\bar{\boldsymbol{U}}\right) + \frac{1}{Re}\left(\boldsymbol{\Phi_i}, \boldsymbol{\nabla}^2\bar{\boldsymbol{U}}\right) \tag{36}$$

$$b_{ij} = -\left(\boldsymbol{\Phi_i}, (\boldsymbol{\Phi_j}.\boldsymbol{\nabla})\,\bar{\boldsymbol{U}}\right) - \left(\boldsymbol{\Phi_i}, (\bar{\boldsymbol{U}}.\boldsymbol{\nabla})\,\boldsymbol{\Phi_j}\right) + \frac{1}{Re}\left(\boldsymbol{\Phi_i}, \boldsymbol{\nabla}^2\boldsymbol{\Phi_j}\right) \tag{37}$$

$$c_{ijk} = -\left(\boldsymbol{\Phi_i}, (\boldsymbol{\Phi_j}.\boldsymbol{\nabla})\,\boldsymbol{\Phi_k}\right) \tag{38}$$

$$d_{ij} = (\boldsymbol{\Phi_i}, \boldsymbol{\Phi_j}) \tag{39}$$

The system of ordinary differential equations is then time marched using the fourth order Runge-Kutta scheme to obtain the prediction coefficients. Note that only ROMs using the lid-driven cavity data is studied in the current work to verify that the sparse bases computed using the new sparse coding approach are of equivalent quality to those computed using the past approach. For the lid-driven cavity flow, the projection of the pressure term is zero since the bases satisfy the homogeneous boundary condition at all boundaries.

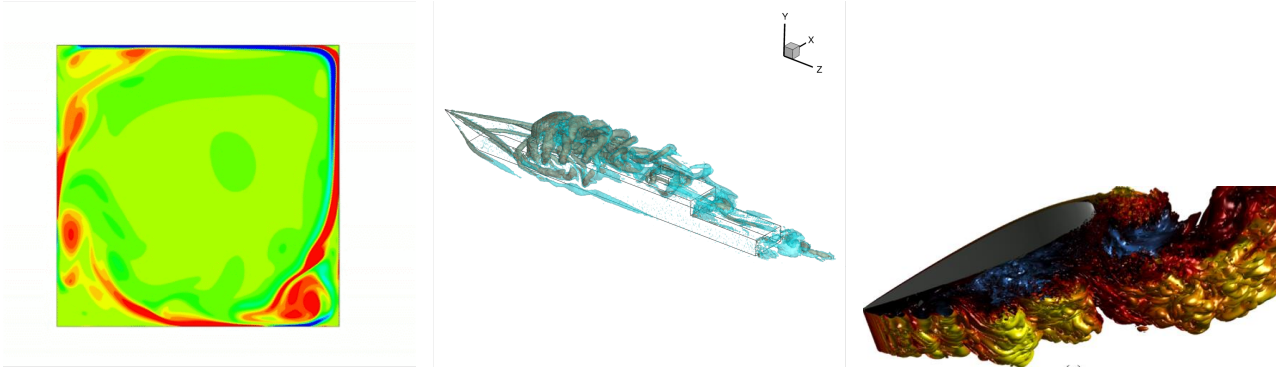
III. Fluid Flow Applications and Datasets

The parallel POD and sparse coding algorithms is applied to high resolution flow data from lid-driven cavity, ship airwake and reversed airfoil problems. The lid-driven cavity problem has been examined previously for initial study of sparse bases for Galerkin based ROMs.[18,19] The unsteady response of a fluid enclosed in a 2-D square cavity to rigid lid translating with constant velocity is studied, as shown in Fig. 2(a). The DNS solutions are obtained using a 512 $\times$ 512 uniform grid. A total of 10000 snapshots is obtained over 100 time units after the flow is statistically stationary, amounting to V_Q=42GB of data.

The wake behind a stationary ship is simulated using the commercial software ANSYS Fluent on a grid with 6-million tetrahedral elements. The simulation was carried out with a time step of 0.01 seconds. Snapshots were sampled for every 10 time steps and 3000 snapshots were obtained, resulting in V_Q=162GB of data. An instantaneous vortex formation near the ship is shown in Fig. 2(b).

The dataset of the flow past a reversed airfoil is from a 3-D LES simulation presented in Ref. 23 (see Fig. 2(c) for an instantaneous flow). The mesh is an O-type mesh, consisting of 643 $\times$ 395 $\times$ 75 grid points. Thus, it is about 57 million variables in a snapshot and V_Q=370GB of data for 810 snapshots over 8.1 non-dimensional time units.

These three configurations are chosen in order to examine the parallel sparse coding algorithm on characteristically different datasets, as highlighted by Table 2. Compared to the lid-driven cavity data, the airfoil data is approximately one order of magnitude smaller in the number of snapshots, two orders of magnitude larger in the size of a snapshot, and one order of magnitude larger in data volume. The ship airwake data is between the lid-driven cavity and airfoil in terms of these quantities. This diversity of the datasets enables broader understanding into the performance of the sparse coding algorithm.



(a) Lid-driven cavity (b) Ship airwake (c) Reversed airfoil. Courtesy Ref. 23

Figure 2. Instantaneous flow of lid-driven cavity, ship airwake, and reversed airfoil

Table 2. Parameters of datasets of lid-driven cavity, ship airwake and reversed airfoil

	Lid-driven cavity	Ship airwake	Reversed airfoil
# of snapshots, m	10000	3000	810
# of variables in a snapshot, n (million)	0.5	18	57
Volume of data, V_Q (GB)	42	162	370

IV. Results

In this section, computational performances of the parallel POD and sparse algorithm are studied with respect to parallel scalability, the effects of number of modes, sparsity, and size of the dataset. Computational cost of the new and 'original'[18,19] sparse coding algorithms are compared, and the quality of sparse modes is examined via reconstruction errors and turbulence kinetic energy prediction.

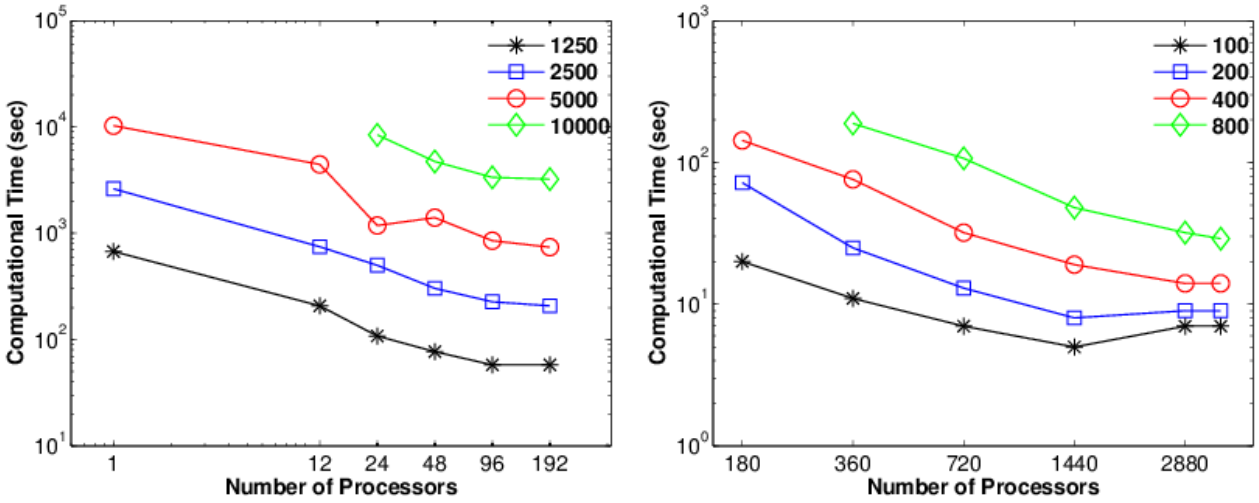
IV.A. Computational Performances of POD and Sparse Approaches

Computational performances of the POD and new sparse coding approaches are presented. First, the scalability of POD approach with number of processors is assessed. Next, the effects of number of modes, sparsity, and size of the dataset on computational time are studied. Finally, the new and 'original'[18,19] sparse coding algorithms are compared.

IV.A.1. Scalability of the Parallel POD Algorithm

A scale-out study of the parallel POD algorithm is performed by adding more processors into the computation. To better understand the scalability of the parallel algorithm, the datasets of the lid-driven cavity and the reversed airfoil, whose dataset parameters differ most, are used in this study. In the process grid configuration for ScaLAPACK, n_{pcol} is fixed to 4 and n_{prow} scales according to n_p. The computational time as a function of the number of processors for different numbers of snapshots is shown in Fig. 3. In each case, the number of POD modes generated is the same as the number of snapshots by assuming the snapshot matrix is full-rank.

For the lid-driven cavity data (Fig.2(a)), the speedup from the sequential algorithm to the parallel algorithm using 12 processors varies from 2.31x to 3.51x for different numbers of snapshots. The low speedup is mainly because only n_{prow} processors in a processor column are involved when normalizing the POD modes (Eq. 23). Overhead caused by inter-processor communications is another reason. The speedup from 12 to 24 processors is 1.93x for 1250 snapshots, which is close to the linear speedup of 2.0. However, this speedup decreases to 1.50x with the number of snapshots increasing. When more numbers of processors are used, the overhead caused by inter-processor communication becomes prominent because each node spends more time in communication than useful computing, and it finally dominates the computational time. The speedup from 96 to 192 processors drops to 1.00x for 1250 snapshots. Adding processors beyond



(a) Lid-driven cavity (b) Reversed airfoil

Figure 3. Log-log plot of time costs of generating POD modes against the number of processors for different sizes of snapshot matrix.

192 may increase rather than decrease the time costs for this dataset. Similar behaviors are observed for the airfoil data. The speedup is approximately linear from 180 to 720 processors, with the lowest being 1.57x. The effect of overhead becomes significant after $n_p > 720$ for 100 and 200-snapshot cases, and $n_p > 1440$ for the others.

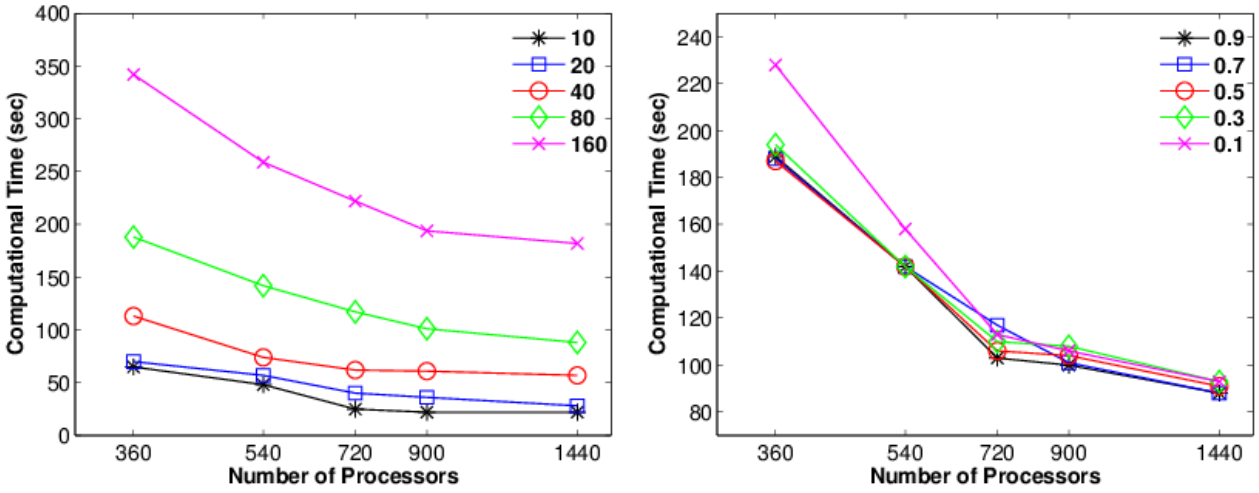
IV.A.2. Scalability of the Parallel Sparse Coding Algorithm

Next, the performance of parallel sparse coding algorithm is studied using the airfoil dataset. The computational time as a function of the number of processors for generating different numbers of sparse modes is shown in Fig. 4(a). Using 800 POD modes and 800 snapshots, a number of sparse modes from 10 to 160 were generated with sparsity fixed to 0.7. The speedup varies from 1.54x to 1.82x for different modes as the number of processors doubles from 360 to 720. Note that after the number of processors exceeds 800, some processors are not assigned to compute the sparse coefficient in the parallel batch-OMP algorithm since the number of snapshots is only 800. However, all processors contribute to the computation in Steps 3 and 4. Therefore, speedup larger than 1.0x is still obtained even the number of processors raises to 900. As the number of processors further increases to 1440, overhead becomes significant.

Furthermore, the computational time for different sparsity values is shown in Fig. 4(b). The number of sparse modes to be generated is 80, which is 10% of the number of snapshots. Although sparsity of 50-80% is found to yield stable and accurate ROMs in the previous studies,[18,19] here, a larger spectrum from 10-90% is studied for better understanding on the scalability of the algorithm with respect to sparsity. Except for the $S = 0.1$ cases, sparsity slightly affects the computational time, using the same number of processors. This is consistent with the complexity analysis of the sparse coding algorithm. As described in Section II.C, the sparse coding algorithm consists of the batch-OMP method, the MOD mode, sparse modes construction and sparse coefficients computation. It is found that sparsity significantly affects the run-time of the batch-OMP method. However, in all the airfoil cases, the run-time is less than 10 seconds, which are much smaller than the time of the other sparse coding operations. Thus, sparsity is an insignificant factor to the computational efficiency when the number of processors varies.

IV.A.3. Effect of Sparsity and Number of Modes

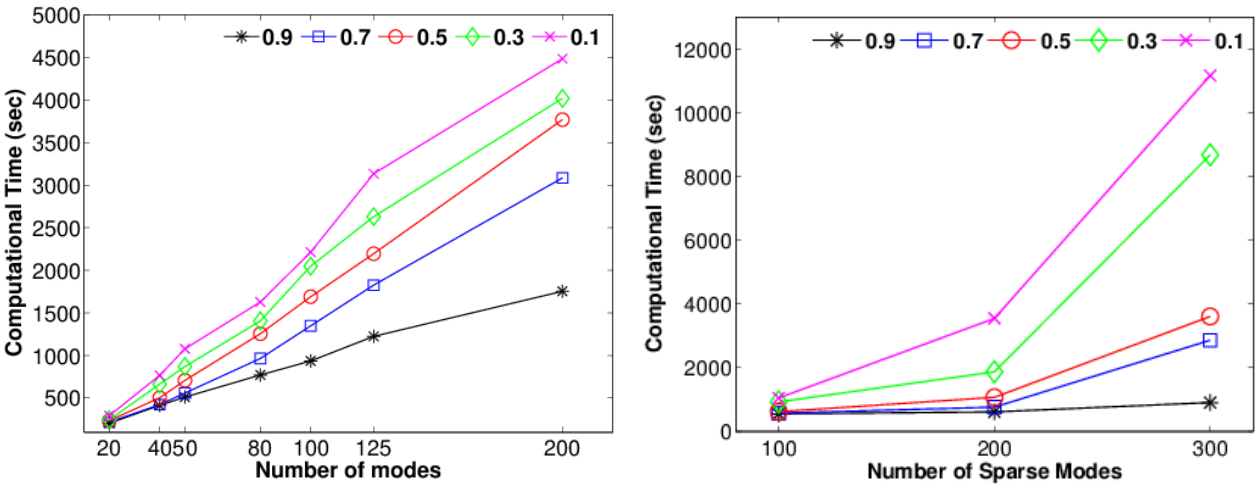
To gain further insight into the effect of number of sparse modes and sparsity on computational time, for a fixed number of processors, the costs of computing different number of modes at different sparsity level are studied. For this purpose, the lid-driven cavity dataset and the ship airwake dataset are used. The lid-driven cavity dataset consists of 1250 snapshot with a data volume of V_Q=5.2GB. The ship airwake data is



(a) For the numbers of modes (b) For sparsity

Figure 4. Semi-log plot of time costs against the number of processors

truncated to contain 1 million variables in each snapshot. It includes 3000 snapshots with a data volume of V_Q=24GB. The size of the datasets enables all cases to be executed in parallel using 24 Intel X5650 processors on two computational nodes with 48GB memory per node. Time costs as a function of the number of sparse modes for different sparsity values are shown in Fig. 5. In general, the time monotonically increases to generate more sparse modes. For the same number of modes, the time increases as sparsity decreases. Because the active number of modes L increases, the time cost for the batch-OMP method increases accordingly to find coefficients for the active modes. For both datasets, the time of executing the batch-OMP method is comparable or even larger than the time of executing the other parts of the sparse coding algorithm. Therefore, the time variation caused by sparsity significantly affect the total time when the number of processors is fixed. Increasing the number of processors to certain level may attenuate the effect of sparsity, as disscused in Sec. IV.A.2. Note that the increment of time cost for the batch-OMP method exhibits linear growth and quadratic growth for the lid-driven cavity dataset and the ship airwake dataset, respectively.



(a) Lid-driven cavity (b) Ship airwake

Figure 5. Time costs of generating sparse modes against the number of modes with respect to sparsity.

IV.A.4. Speedup Comparison Between the 'Original' and new Sparse Coding Algorithms

The computational efficiency is compared next for the 'original' algorithm used in Ref. 18, 19 and the new algorithm shown in Fig. 1. Both algorithms are applied to the datasets of lid-driven cavity and ship airwake. Since the 'original' sparse coding algorithm is a serial implementation, in order to execute it on a node with 48GB memory, only 40 snapshots is considered for the ship airwake dataset. The number of snapshots and the number of sparse modes are fixed to 40 for both datasets. Therefore, the only parameter that differs the datasets is the number of variables in a snapshot, n. Sparsity, S, is specified to 0.7 for both algorithms. Note that the β value was adjusted to produce the desired sparsity in the 'original' algorithm, whereas sparsity was directly assigned in the new algorithm using Eq. 10.

All cases were executed in serial on an Intel X5650 processor. To measure the speed of data processing using the sparse coding algorithms, Megabyte data processed Per Second (MPS) is used:

$$\text{MPS} = \frac{V_Q}{\text{second} \times 10^6} \tag{40}$$

The MPS value can be affected by multiple factors: the number of modes to be generated, the number of processors used, and the computational specification of the processors, etc. Here, since only the number of variables in a snapshot,n, is different for two datasets, it can provide an estimation about the speed of data processing as a function of n, for these two algorithms.

The time costs of two algorithms are listed in Table 3. It is found that the 'original' algorithm is advantageous to process snapshots with small size, as indicated by a higher MPS value of 3.4 MB/sec for the lid-driven cavity dataset. On the contrary, the new algorithm is more efficient at processing snapshots with large size, as indicated by a MPS value of 108.7 MB/sec for the ship airwake dataset. The time of computing POD modes (Step 1) is 1.3 and 45.8 seconds for the lid-driven cavity and airwake data, respectively; while the time of computing sparse modes is comparable (3.7 vs. 6.9 seconds). Overall, using the new algorithm, a speedup of 9.8 is achieved for the lid-driven cavity data, and it raises to 49.8 for the ship airwake data.

Table 3. Time costs of sparse coding algorithms applied to lid-driven cavity and ship airwake data

	Data size	Original[18,19]		New		Speedup
	V_Q (MB)	Time (sec)	MPS (MB/sec)	Time (sec)	MPS (MB/sec)	
Lid-driven cavity	167.8	49.0	3.4	5.0	33.6	9.8
Ship airwake	5760.0	2624.7	2.2	52.7	108.7	49.8

IV.B. Assessment of Quality of Extracted Modes

The quality of extracted modes using the POD, 'original'[18,19] and new sparse coding approaches is assessed next. Two metrics is used in the assessment: 1) ability to effectively extract information from a snapshot matrix, and 2) accuracy of predictions obtained from corresponding ROMs.

IV.B.1. Reconstruction Errors

The ability of the new sparse coding approach to carry out effective dimensionality reduction is compared to that of POD and sparse coding approach used in Ref. 18, 19. Recall that POD is optimal in terms of extracting modes that best reconstruct snapshot matrix; whereas sparse modes are sub-optimal due to the presence of L0 or L1 penalties in the sparse coding approaches. Thus, it is important to quantify the loss of information in capturing the snapshot matrix when using sparse coding, as compared to POD approach. Two different datasets are considered for this purpose: 1) flow inside a lid-driven cavity, and 2) flow past a reversed airfoil. The snapshot matrices are approximated using different sets of POD and sparse modes, and errors of reconstructions are compared. Note that the new sparse coding approach is compared to both the POD and 'original' sparse coding approaches[18,19] for the lid-driven cavity dataset, whereas the reverse airfoil dataset is only used for comparison between the new sparse coding and POD approaches. The qualities of bases extracted for the lid-driven cavity flow is discussed first.

The normalized L2 error in reconstruction is given as:

$$Error = \frac{\|\boldsymbol{Q} - \boldsymbol{\Phi}\boldsymbol{S}\|_F^2}{\|\boldsymbol{Q}\|_F^2} \times 100\%, \quad i = 1, 2, ..., m. \tag{41}$$

The errors in representation of the matrix containing 1250 snapshots of the lid-driven cavity flow using different numbers of modes are listed in Table 4. As noted, the errors corresponding to the POD sets are smaller than those of sparse sets, for same number of modes. However, the difference between the error in representation using POD and sparse sets are small, with less than 1.5% of maximum difference. Thus, the information within the snapshots is retained accurately. Moreover, the differences between the 'original' and new approaches are insignificant.

Table 4. Errors of reconstructing lid-driven cavity flow fields using POD and sparse modes. Sparsity, S, is specified to 0.7 for the new sparse algorithm. The value of β is adjusted to produce the equivalent sparsity effect for the sparse algorithm used in Ref. 18, 19.

Number of Modes	POD (%)	'Original' Sparse algorithm[18,19] (%)	New sparse algorithm (%)
10	43.48	44.27	44.79
20	28.32	28.98	29.51
50	9.21	10.16	10.33
80	3.56	3.78	3.77
100	2.06	2.12	2.10
200	0.21	0.21	0.22

The errors in reconstruction of flow past reverse airfoil using modes extracted from the new sparse coding and POD approaches are discussed next. The errors in reconstruction of matrix containing 810 snapshots are listed in Table 5. Similar to the observations made in previous set of results, the POD errors are only marginally smaller than that of new sparse approach. Hence, the new sparse coding approach is able to effectively capture the information contained within the snapshot matrix..

Table 5. Errors of reconstructing reversed airfoil flow fields using POD and sparse modes computed using the new algorithm. S=0.7 for sparse

Number of Modes	POD (%)	Sparse (%)
10	15.50	15.58
20	12.04	12.05
40	9.12	9.17
80	5.99	6.00

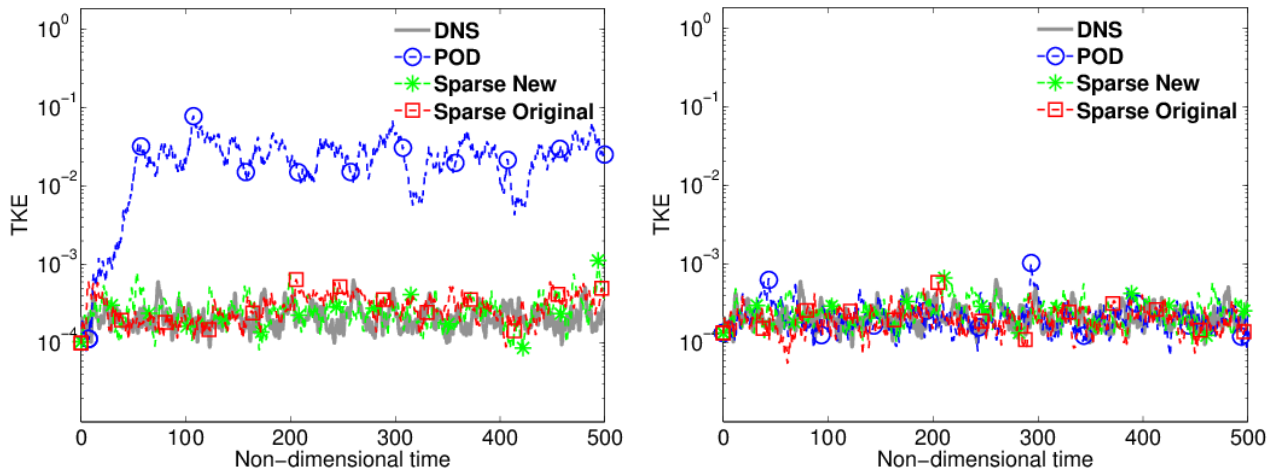
IV.B.2. Turbulence Kinetic Energy Prediction

Next, the POD and sparse ROMs prediction capabilities are assessed using instantaneous turbulence kinetic energy (TKE) metric. The TKEs of DNS solution for lid-driven cavity flow, and 20- and 80-mode ROMs solutions are compared in Fig. 6. The snapshot matrix spans the first 25 time units, and the ROMs are integrated for 500 time units. For the 20-modes ROMs, the averaged TKE predicted by the POD-ROM for 500 time units is two orders of magnitude larger than that of DNS, whereas all the sparse-ROMs predict the TKE with reasonable accuracy. Moreover, as many as 80 POD modes are required to accurately capture the TKE levels. These results are consistent with those developed using the original algorithm, thus verifying the quality of bases computed using the new approach.

V. Conclusion

A parallel sparse coding algorithm developed for efficient fluid data processing is described, with its computational complexity and memory requirement analyzed. The algorithm is applied to three datasets: a lid-driven cavity, ship airwake and reversed airfoil problems. Computational performance with respect to sparsity, the number of modes, snapshots and processors are studied. Reconstructions using POD and sparse modes are performed and the errors are compared. Galerkin projection based ROMs are developed and their performances in predicting the turbulent kinetic energy are compared. Some concluding remarks are:

1. Compared to the previous sparse coding method,[18,19] the new algorithm is faster, and the speedup increases as the volume of data increases.



(a) 20-mode ROMs (b) 80-mode ROMs

Figure 6. Time history of the instantaneous TKE of DNS, and as predicted by the POD and sparse-ROMs. Sparsity, S, is specified to 0.75 for the new sparse algorithm. The value of β is adjusted to produce the equivalent sparsity effect for the original sparse algorithm

2. The computational performance of the parallel POD algorithm scales non-linearly from 1.50x to 1.93x when doubling the number of processors.

3. The performance of the parallel sparse coding algorithm is affected by sparsity and the number of sparse modes to be generated. The time monotonically increases as the number of sparse modes increases. For the same number of modes, the time increases as sparsity decreases.

4. The performance of the parallel sparse coding algorithm scales non-linearly from 1.54x to 1.82x when doubling the number of processors. When the number of processor is more than the number of snapshots to be evaluated, the speedup drops to approximately 1.0x.

5. Overhead caused by inter-processor communication dominates the computational time in both POD and sparse coding procedures when a large amount of processors are used.

6. The new sparse coding method can generate high quality sparse modes for flow reconstructions and constructing sparse-ROMs.

Acknowledgments

The authors gratefully acknowledge the support of ONR grant N00014-14-1-0018, under the direction of Dr. Judah Milgram, an HPCMPO Frontier PETTT Project Grant, under the direction of David Bartoe, and an allocation of computing time from the Ohio Supercomputer Center. The authors thank Kelsey Shaler for providing the reversed airfoil data, and Dr. Benjamin Grier for providing techniqical support in the case study of reversed airfoil.

References

[1]Holmes, P., Lumley, J. L., Berkooz, G., and Rowley, C., *Turbulence, coherent structures, dynamical systems and symmetry*, Cambridge university press, New York, 2nd ed., 2012.

[2]Ilak, M., Bagheri, S., Brandt, L., Rowley, C. W., and Henningson, D. S., "Model reduction of the nonlinear complex Ginzburg-Landau equation," *SIAM Journal on Applied Dynamical Systems*, Vol. 9, No. 4, 2010, pp. 1284–1302.

[3]Kalashnikova, I., van Bloemen W. B., Arunajatesan, S., and Barone, M., "Stabilization of projection-based reduced order models for linear time-invariant systems via optimization-based eigenvalue reassignment," *Computer Methods in Applied Mechanics and Engineering*, Vol. 272, 2014, pp. 251–270.

[4]Rowley, C., "Model reduction for fluids, using balanced proper orthogonal decomposition," *International Journal of Bifurcation and Chaos*, Vol. 15, No. 03, 2005, pp. 997–1013.

[5]Noack, B. R., Afanasiev, K., Morzynski, M., Tadmor, G., and Thiele, F., "A hierarchy of low-dimensional models for the transient and post-transient cylinder wake," *Journal of Fluid Mechanics*, Vol. 497, 2003, pp. 335–363.

[6]Noack, B. R. and Eckelmann, H., "A low-dimensional Galerkin method for the three-dimensional flow around a circular cylinder," *Physics of Fluids (1994-present)*, Vol. 6, No. 1, 1994, pp. 124–143.

[7]Lucia, D. J., Beran, P. S., and Silva, W. A., "Reduced-order modeling: new approaches for computational physics," *Progress in Aerospace Sciences*, Vol. 40, No. 1, 2004, pp. 51–117.

[8]Berkooz, G., Holmes, P., and Lumley, J. L., "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual review of fluid mechanics*, Vol. 25, No. 1, 1993, pp. 539–575.

[9]Chatterjee, A., "An introduction to the proper orthogonal decomposition," *Current science*, Vol. 78, No. 7, 2000, pp. 808–817.

[10]Sirovich, L., "Turbulence and the dynamics of coherent structures. I-Coherent structures. II-Symmetries and transformations. III-Dynamics and scaling," *Quarterly of applied mathematics*, Vol. 45, 1987, pp. 561–571.

[11]Balajewicz, M. J., Dowell, E. H., and Noack, B. R., "Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier–Stokes equation," *Journal of Fluid Mechanics*, Vol. 729, 2013, pp. 285–308.

[12]Ilak, M. and Rowley, C. W., "Modeling of transitional channel flow using balanced proper orthogonal decomposition," *Physics of Fluids (1994-present)*, Vol. 20, No. 3, 2008, pp. 034103.

[13]Amsallem, D. and Farhat, C., "Stabilization of projection-based reduced-order models," *International Journal for Numerical Methods in Engineering*, Vol. 91, No. 4, 2012, pp. 358–377.

[14]Pope, S. B., *Turbulent Flows*, Cambridge University Press, New York, sixth ed., 2009.

[15]Amsallem, D., Zahr, M. J., and Farhat, C., "Nonlinear model order reduction based on local reduced-order bases," *International Journal for Numerical Methods in Engineering*, Vol. 92, No. 10, 2012, pp. 891–916.

[16]Zhou, K., Doyle, J. C., and Glover, K., *Robust and optimal control*, Prentice Hall, 1st ed., 1996.

[17]Ma, Z., Ahuja, S., and Rowley, C. W., "Reduced-order models for control of fluids using the eigensystem realization algorithm," *Theoretical and Computational Fluid Dynamics*, Vol. 25, No. 1-4, 2011, pp. 233–247.

[18]Deshmukh, R., Liang, Z., Gogulapati, A., McNamara, J. J., and Kolter, J. Z., "Basis Identification for Reduced Order Modeling of Unsteady Flows Using Sparse Coding," *AIAA Paper 2015-2053*, 2015.

[19]Deshmukh, R., Liang, Z., and McNamara, J. J., "Reduced Order Modeling of Turbulent Flows using Sparse Coding," *AIAA Paper 2016-0462*, 2016.

[20]Olshausen, B. and David, J., "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, Vol. 381, No. 6583, 1996, pp. 607–609.

[21]Mittal, R., Dong, H., Bozkurttas, M., Najjar, F., Vargas, A., and von Loebbecke, A., "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries," *Journal of computational physics*, Vol. 227, No. 10, 2008, pp. 4825–4852.

[22]Gaitonde, D. V. and Visbal, M. R., "High-order schemes for Navier-Stokes equations: algorithm and implementation into FDL3DI," Tech. rep., DTIC Document, 1998.

[23]Shaler, K. and Gaitonde, D. V., "Flow Control of a Retreating Airfoil via NS-DBD Actuators Using Large Eddy Simulations," *ASME 2014 4th Joint US-European Fluids Engineering Division Summer Meeting collocated with the ASME 2014 12th International Conference on Nanochannels, Microchannels, and Minichannels*, American Society of Mechanical Engineers, 2014, pp. V01AT09A002–V01AT09A002.

[24]Yang, J., Yu, K., Gong, Y., and Huang, T., "Linear spatial pyramid matching using sparse coding for image classification," *Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 1794–1801.

[25]Aharon, M., Elad, M., and Bruckstein, A., "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *Signal Processing, IEEE Transactions on*, Vol. 54, No. 11, 2006, pp. 4311–4322.

[26]Rubinstein, R., Zibulevsky, M., and Elad, M., "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," *CS Technion*, Vol. 40, No. 8, 2008, pp. 1–15.

[27]Mallat, S. G. and Zhang, Z., "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, Vol. 41, No. 12, 1993, pp. 3397–3415.

[28]Chen, S., Billings, S. A., and Luo, W., "Orthogonal least squares methods and their application to non-linear system identification," *International Journal of control*, Vol. 50, No. 5, 1989, pp. 1873–1896.

[29]Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P., "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, IEEE, 1993, pp. 40–44.

[30]Tropp, J. et al., "Greed is good: Algorithmic results for sparse approximation," *Information Theory, IEEE Transactions on*, Vol. 50, No. 10, 2004, pp. 2231–2242.

[31]Engan, K., Aase, S. O., and Hakon H., J., "Method of optimal directions for frame design," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 5, IEEE, 1999, pp. 2443–2446.

[32]Blackford, L. S., Choi, J., Cleary, A., D'Azeuedo, E., Demmel, J., Dhillon, I., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., and Whaley, R. C., *ScaLAPACK User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[33]Cormen, T. H., *Introduction to algorithms*, MIT press, 2009.