Clustering of State Vectors from CFD Simulations for Construction of Local Reduced-Order Models

Kyle Washabaugh*

Stanford University, Stanford, CA, 94305-3035, USA

A novel method for clustering computational fluid dynamics (CFD) solution vectors with the goal of building efficient local reduced-order models (ROMs) is proposed. In this method, the clustering process is accelerated through the introduction of triangle-inequality bounds to avoid unnecessary computation during K-Means, and the accuracy of the resulting ROM simulation is improved through the introduction of a semi-supervised clustering algorithm. The performance benefits of the proposed method are demonstrated using two data sets generated from CFD simulations. For the larger of the two datasets the proposed method was able to cluster 183 gigabytes of data (1252 state vectors) into five clusters in under two minutes using 512 CPUs.

I. Introduction

Discretizing the Navier Stokes equations in space and time can easily produce models with many millions of degrees of freedom, and has given Computational Fluid Dynamics (CFD) a reputation for requiring vast computational resources. Because of this inherent high-dimensionality, CFD problems are an intriguing application for Model Order Reduction (MOR) methods.

The goal of MOR is to greatly reduce the dimensionality a model by exploiting relationships between the degrees of freedom. This implies that a High-Dimensional Model (HDM) is a strong candidate for MOR if its trajectories are mostly confined to low-dimensional affine subspaces of the state space. When this is the case, it is possible to project the equations of the high-dimensional model (HDM) onto the low-dimensional subspace spanned by its trajectories, greatly reducing the number of degrees of freedom in the model while retaining the accuracy of the original HDM.[1–6]

This low-dimensional projection subspace is typically represented by a reduced-order basis (ROB), which is constructed from training snapshots of the fluid state that are precomputed using the HDM and then compressed using a procedure such as Proper Orthogonal Decomposition (POD).[7]

The resulting reduced-order model (ROM) can efficiently produce accurate responses within the projection subspace; however, a fundamental trade-off is that the ROM will never be able to explore portions of the state-space outside this subspace. Unfortunately, this trade-off greatly complicates the task of building ROMs for parameter-dependent problems. The complication arises because, although the trajectories of CFD models are typically confined to low-order subspaces, these subspaces can be strongly parameter-dependent. As a result, a ROM built for Mach=0.7 would not necessarily be able to capture the behavior at Mach=0.8.

The simplest solution to this problem is to construct a "global" ROB that includes the necessary projection subspaces for all operating points of interest. Typically, the HDM is evaluated at a representative set of operating points to gather snapshots, and the ROM is then queried for for predictive operating points. When a single ROB is used to reduce the dimension of an HDM, it must capture the dynamics of the HDM along the entire trajectory of interest. This can result in a very large ROB, and thus an inefficient reduced order model – an issue that is especially troublesome for design applications, where the ROB needs to capture the dynamics of the HDM at multiple design points.

The MOR method introduced in [8] alleviates this problem through the use of multiple *local* ROBs. In this approach, a local ROB is selected at each time step of the ROM simulation based on the current state of the system. Ideally, each local ROB captures only the local dynamics at a given point of the state-space, resulting in small and accurate models. Such a concept is particularly well-suited for the proper orthogonal decomposition (POD) method in which the ROB is built from snapshots of the system taken at various locations of the state-space.

This local ROB approach takes advantage of the fact that the state-space of a nonlinear dynamical system can often be partitioned into distinct characteristic regimes. In the context of CFD simulations, these state-space

*Graduate Student, Department of Aeronautics and Astronautics, William F. Durand Building, Room 028, Stanford University, Stanford, CA 94305-3035

partitions could distinguish, for example, between solutions that are dominated by laminar vs. turbulent flow, subsonic vs. supersonic flow, or transient vs. limit-cycle behavior. As the state of the system transitions from one regime to another during a ROM simulation, an appropriate local ROB can be chosen that best captures the physics of the current system, and omits information that is not immediately relevant.

This work focuses on method of partitioning the CFD state snapshots into natural clusterings using the k-means algorithms. Due to length constraints, this write-up will focus on the proposed clustering method and the mathematical framework for local MOR will unfortunately need to be left to the references (e.g. [8]). This write-up is organized as follows: section II describes a proposed approach to speed up k-means using the triangle inequality, section III describes a partially-supervised version of the k-means algorithm should prove useful for creating accurate ROMs, and the remaining two sections show numerical results for the proposed methods.

II. Accelerating K-Means Using the Triangle Inequality

In this work, it is proposed to accelerate the k-means clustering algorithm using the triangle inequality to skip unnecessary distance calculations. The idea is very simple: if the distances from a datum to the cluster centers were calculated exactly at some previous k-means iteration, and the uncertainty in these distances is incremented at each subsequent iteration, then it is often possible to rule out that the datum has changed clusters *without* recalculating any of the distances exactly. Since there are no approximations introduced, it is easy to show that the iterations of the accelerated algorithm are identical to those of the non-accelerated algorithm (just substantially cheaper). Pseudo-code for this method is included as Alg. 1.

Not surprisingly, I was not the first person to think of this approach – it appears to have first been introduced in 2003 by Elkan [9]. To the best of my knowledge, my algorithm is subtly different than those that have been proposed before due to the treatment of the bounds. In all methods that I have seen, the bounds are incremented whenever a cluster center moves, and eventually become very loose. In my algorithm I also increment the bounds whenever a cluster center moves, but before every k-means iteration I evaluate the bounds exactly to prevent them from becoming too loose. This is admittedly a minor detail, but numerical results show that the tighter bounds result in significantly fewer unnecessary distance calculations.

III. Semi-Supervised Clustering For ROMs

As mentioned in the introduction, some degree of robustness to parameter changes can be built into a ROM if it is constructed using training simulations from a variety of operating points (for example, different free-stream Mach number for a fluid simulation). The idea is that the ROBs will then have information corresponding to different parameters, and should be more parametrically robust as a result. This approach can be sabotaged by a poor clustering algorithm, however.

As will be shown in section V, it is often the case the snapshots from a single simulation are more similar to themselves than they are to the snapshots gathered at other operating points. The result is that clustering with the standard k-means algorithm tends to produce some clusters that contain snapshots from only a single training simulation, resulting in reduced order bases that are not likely to be parametrically robust.

In this work, I propose an algorithm that constrains the clusters to contain snapshots from every training simulation. This is performed by first clustering the data from each training simulation separately into the desired number of clusters, and then taking two datasets at a time and performing a (low-dimensional) combinatorial search to establish which clusters from the two data sets should be agglomerated. This process is repeated until each cluster contains snapshots from every training simulation. I have referred to this approach as "semi-supervised" clustering because I am introducing additional information into the algorithm to obtain clusters that are better suited for my application. Pseudo-code for this method is included as Alg. 2.

IV. Accelerated K-Means Application: Unsteady Ahmed Body

The first dataset was generated using a single unsteady simulation of an Ahmed body, which is a common CFD benchmark in the automotive industry for turbulent flow around automobiles. This simulation was run at low mach number with a Reynolds number of 4.29×10^6. I used a DES turbulence model for this simulation due to the massive flow separation that occurs on the aft body. The fluid mesh has 2,890,434 fluid nodes, resulting in 17,342,604 degrees of freedom in the HDM system. The dataset of 1252 solution vectors totals 183 gigabytes. A detail of the mesh is shown in Fig. 1(a), the drag history of the unsteady simulation is shown in Fig. 1(b), and a two dimensional representation of the clustered 17,342,604-dimensional data is shown in Fig. 1(c) (the color indicates the cluster).

Algorithm 1 Accelerated K-Means Using the Triangle Inequality

Input: State snapshots $\mathcal{S}$

Output: State clusters $\{\mathcal{W}_j\}_{j=1}^{N_{\text{Clusters}}}$; state cluster centroids $\{\overline{\mathbf{w}}_j\}_{j=1}^{N_{\text{Clusters}}}$

Initialize the cluster centroids randomly as $\{\overline{\mathbf{w}}_j^{(1)}\}_{j=1}^{N_{\text{Clusters}}}$
while (K-Means has not converged) **do**
  if (Using "Tight(er)" bounds) **then**
    Store cluster centroids from this KMeans iteration.
  end if
  if (This is the first iteration) **then**
    Calculate $d_{i,j}$, the distance from each snapshot $\mathbf{s}_i$ to each centroid $\overline{\mathbf{w}}_j^{(1)}$
    Initialize $u_{i,j}$, the uncertainties associated with $d_{i,j}$, to zero
    Assign snapshots to nearest cluster and calculate $\{\overline{\mathbf{w}}_j^{(2)}\}_{j=1}^{N_{\text{Clusters}}}$
  else
    for $i = 1, \cdots, N_{\text{snaps}}$ **do**
      for $j = 1, \cdots, N_{\text{Clusters}}$ **do**
        Set $\hat{c}$ as the current cluster of snapshot $\mathbf{s}_i$
        if $(0 < d_{i,\hat{c}} - d_{i,j} + u_{i,\hat{c}} + u_{i,j})$ **then**
          Update $d_{i,\hat{c}}$ and set $u_{i,\hat{c}} = 0$
          if $(0 < d_{i,\hat{c}} - d_{i,j} + u_{i,j})$ **then**
            Update $d_{i,j}$ and set $u_{i,j} = 0$
            if $(0 < d_{i,\hat{c}} - d_{i,j})$ **then**
              Assign snapshot i to cluster j
              Update cluster centers, increment the corresponding uncertainties for all snapshots $(u_{*,j}, u_{*,\hat{c}})$
            end if
          end if
        end if
      end for
    end for
  end if
  if (Using "Tight(er)" bounds) **then**
    Evaluate all $u_{i,j}$ exactly using the current cluster centroids and the stored cluster centroids
  end if
end while

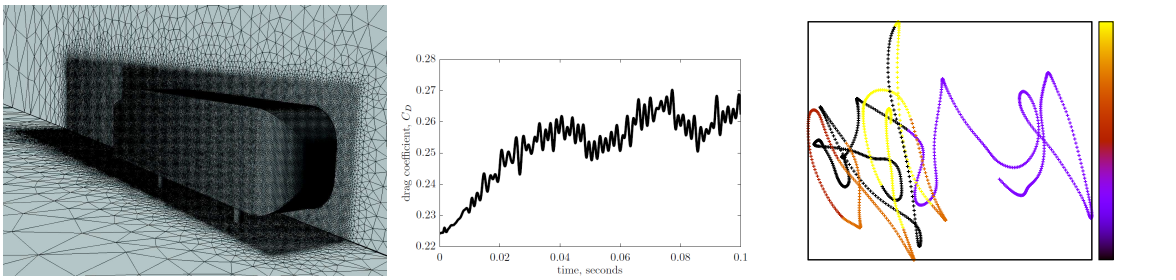
Algorithm 2 Semi-Supervised Clustering For ROMs

Input: Sets of state snapshots $\{\mathcal{S}_i\}_{i=1}^{N_{\text{Sets}}}$

Output: Agglomerated state clusters $\{\widehat{\mathcal{W}}_j\}_{j=1}^{N_{\text{Clusters}}}$ and centroids $\{\widehat{\mathbf{w}}_j\}_{j=1}^{N_{\text{Clusters}}}$

for (Each snapshot set index i) **do**
  Find $\{\mathcal{W}_{i,j}\}_{j=1}^{N_{\text{Clusters}}}$ and $\{\overline{\mathbf{w}}_{i,j}\}_{j=1}^{N_{\text{Clusters}}}$ using accelerated K-Means (algorithm 1)
end for
Set $\{\widehat{\mathcal{W}}_j\}_{j=1}^{N_{\text{Clusters}}} = \{\mathcal{W}_{1,j}\}_{j=1}^{N_{\text{Clusters}}}$ and $\{\widehat{\mathbf{w}}_j\}_{j=1}^{N_{\text{Clusters}}} = \{\overline{\mathbf{w}}_{1,j}\}_{j=1}^{N_{\text{Clusters}}}$
for (Each snapshot set index $i > 1$) **do**
  for (Each cluster index, j) **do**
    Calculate and store the distance from centroid $\widehat{\mathbf{w}}_j$ to every centroid $\{\overline{\mathbf{w}}_{i,k}\}_{k=1}^{N_{\text{Clusters}}}$
  end for
  Perform a (small) combinatorial search to find the set of indices k that minimizes $\sum_{j=1}^{N_{\text{Clusters}}} \|\widehat{\mathbf{w}}_j - \overline{\mathbf{w}}_{i,k}\|_2^2$
  Agglomerate the paired clusters and update $\{\widehat{\mathcal{W}}_j\}_{j=1}^{N_{\text{Clusters}}}$ and $\{\widehat{\mathbf{w}}_j\}_{j=1}^{N_{\text{Clusters}}}$
end for

The accelerated K-Means algorithms converged using roughly half the distance calculations as the standard K-Means algorithm, as reported in Table 1.



(a) Detail of Ahmed Body mesh (b) Drag history of simulation (c) 2D representation of clustered data

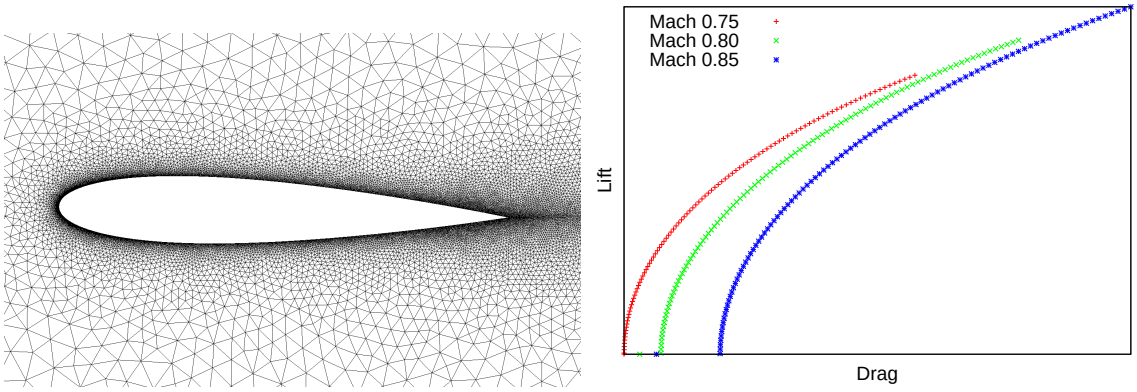
Figure 1. Ahmed body dataset

Table 1. Performance of Accelerated Clustering Algorithms for Ahmed Body Data

K-Means Algorithm	Standard	Accelerated (Loose Bounds)	Accelerated (Tighter Bounds)
K-Means Iterations	13	13	13
Distances computed	75,120	33,182	30,466
Distances skipped	0	43,942	46,405
Wall Clock Time (512 CPUs)	209.24 sec	107.61 sec	102.37 sec
Speedup	-	**1.944**	**2.044**

V. Semi-Supervised Clustering Application: Drag Polars at Variable Mach

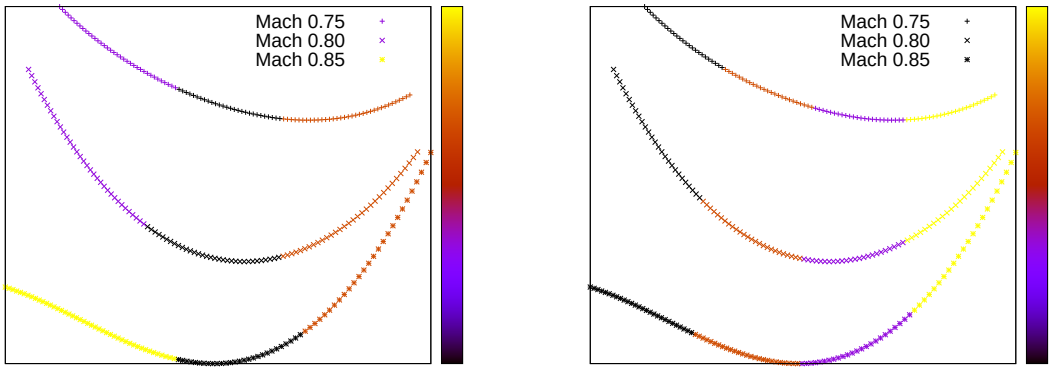
The second dataset was generated using three separate quasi-steady simulations where a NACA0015 airfoil at $M = 0.75, 0.8, 0.85$ and $Re = 6 \times 10^6$ was swept from an angle of attack of $\alpha = 0$ degrees to 10 degrees in increments of 0.1 degrees. As the angle of attack and Mach number increase, the shock on the airfoil strengthens considerably; there is no shock present for any of these Mach numbers at $\alpha = 0$ degrees, but a very strong shock present for $\alpha = 10$ degrees (especially for the $M = 0.85$ simulation). I used a Spalart-Allmaras turbulence model and a wall model for these simulations. The fluid mesh has 40,922 fluid nodes, resulting in 245,532 degrees of freedom in the HDM system. A detail of the mesh is shown in Fig. 2(a) and the lift and drag history of the three simulations is shown in Fig. 2(b).



(a) Detail of NACA0015 mesh (b) Lift vs. Drag for simulations

Figure 2. NACA0015 dataset

Figure 3(a) shows a two-dimensional representation of the data which was clustered using the standard k-means algorithm, whereas Fig. 3(b) shows the results for the proposed semi-supervised clustering method. Examining the results, it appears that the standard k-means algorithm has found clusters that describe the flow physics well, but that would be unlikely to lead to a parametrically robust ROM (the snapshots corresponding to Mach 0.85 at a high angle of attack are placed in their own cluster). The proposed semi-supervised k-means algorithm, however, finds clusters that contain snapshots from every simulation and would likely lead to a more useful ROM.



(a) Standard k-means

(b) Semi-supervised k-means

Figure 3. 2D representation of clustered data clustered for the NACA0015 dataset. The color indicates the cluster number. From top to bottom, these lines correspond to the Mach 0.75, Mach 0.80, and Mach 0.85 training simulations.

VI. Conclusions and Future Work

A method for clustering computational fluid dynamics (CFD) solution vectors with the goal of building efficient local reduced-order models (ROMs) was proposed. In the proposed method, the clustering process is accelerated through the introduction of triangle-inequality bounds to avoid unnecessary computation during K-Means, and the accuracy of the resulting ROM simulation is improved through the introduction of a semi-supervised clustering algorithm. The proposed method was implemented using C++/MPI to allow for efficient parallel computation. Applications to two datasets generated by CFD simulations demonstrate that the proposed method is substantially faster than the standard k-means algorithm (by at least a factor of two for all simulations that I ran), and is able to produce clusters that are better suited for constructing parametrically robust ROMs than the standard k-means algorithm.

References

[1]Rewienski, M. and White, J., "Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations," *Linear Algebra and its Applications*, Vol. 415, No. 2-3, 2006, pp. 426–454.

[2]Lieu, T. and Farhat, C., "Adaptation of aeroelastic reduced-order models and application to an F-16 configuration," *AIAA Journal*, Vol. 45, No. 6, 2007, pp. 1244–1257.

[3]Amsallem, D. and Farhat, C., "Interpolation method for adapting reduced-order models and application to aeroelasticity," *AIAA Journal*, Vol. 46, No. 7, 2008, pp. 1803–1813.

[4]Amsallem, D., Cortial, J., and Farhat, C., "Toward Real-Time Computational-Fluid-Dynamics-Based Aeroelastic Computations Using a Database of Reduced-Order Information," *AIAA Journal*, Vol. 48, No. 9, 2010, pp. 2029–2037.

[5]Chaturantabut, S. and Sorensen, D., "Nonlinear Model Reduction via Discrete Empirical Interpolation," *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[6]Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, 2011, pp. 155–181.

[7]Sirovich, L., "Turbulence and the dynamics of coherent structures. Part I: Coherent structures," *Quarterly of applied mathematics*, Vol. 45, No. 3, 1987, pp. 561–571.

[8]Amsallem, D., Zahr, M., and Farhat, C., "Nonlinear Model Order Reduction Based on Local Reduced-Order Bases," *International Journal for Numerical Methods in Engineering, in press*, 2012, pp. 1–31.

[9]Elkan, C., "Using the Triangle Inequality to Accelerate K-Means," *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, 2003, pp. 1–7.