A spatial clustering algorithm for constructing sparse local reduced-order bases in model order reduction problems

Tina White – Summer 2016

1. Introduction

This research is an extension of a previous implementation of k-means clustering to both the rows (spatial) and columns (temporal) of a snapshot matrix. Originally, reduced order models were constructed and run online using a fixed number of basis vectors corresponding to row-column clusters of Burger's equation solutions. The motivation behind the original row-column clustering method is to capture spatial discontinuities like shock waves. This research extends the previous method by building a matrix of row-column local basis vectors with descending singular values so that energy methods may be applied to choose the singular vectors to truncate. These extensions were made in order to prepare the row-clustering method for efficient implementation in codes where the entire matrix of local basis vectors is calculated offline, stored and reused during online computations. The updated implementation also takes advantage of the sparsity of the matrix and shows that runtime of the resulting ROM can be improved with the methods.

2. Methods

Instead of approximating the solution of interest in a fixed lower-dimensional subspace of local basis vectors constructed from the *columns-only* of a snapshot matrix, this method generates spatially and temporally local basis vectors from the *columns and rows* of a snapshot matrix. The solution space is partitioned into subregions in both space and time, and a local reduced-order basis is constructed and assigned to each subregion offline. While the singular value decomposition of a matrix will yield the best approximations for the entire matrix given data that clusters around a single solution vector or center, CFD datasets represent information that is far from linear in space and time. A singular value decomposition of the entire matrix does not take advantage of the non-linear structure and variation that can exists in the snapshot matrix.

While this clustering method has been applied previously [1-2] the logic was applied in the temporal domain and has not yet been applied to the spatial domain, or in a combination of both the spatial and temporal domain. In the case of column-only (temporal) clustering, the matrix of snapshots is clustered and split along its columns, and then a truncated SVD is performed to generate a set of matrices of basis vectors V that represent the local bases:

$$u = Vy = \{V_1 , V_2\}\, y$$

In the case of spatial row clustering, each basis vector V has also been clustered along its rows. This may be represented as a single submatrix by creating a diagonal matrix from the submatrices of V as shown. Alternatively, the individual matrices could be saved as a set, which will speed up the computation, but implementation would be more complex. The method is currently implemented in the following vectorized form:

$$u = Vy = \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

In the case of spatial and temporal clustering, the basis vector *V* is split first along the columns, and then each submatrix is split along its rows as shown:

$$u = Vy = \{V_1, V_2\}\, y$$

$$u = Vy = \left\{ \begin{bmatrix} V_{11} & 0 \\ 0 & V_{12} \end{bmatrix}, \begin{bmatrix} V_{21} & 0 \\ 0 & V_{22} \end{bmatrix} \right\} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Because the row clustering is arranged in this vectorized form, the matrices V_{11} and V_{12} are reconstructed into V_1 prior to the online update of the local ROB. This enables the method to be used with temporal-only clustering codes with little modification to the current setup of the ROM update. However, this implementation will not improve the computation time. This can only be expected if submatrices are handled separately or treated as sparse. Left in this form, the zero padding leads to a waste of energy. However, in python, the test code pymortestbed was modified to take advantage of the sparsity of the matrix by using the scipy.sparse libraries. The algorithms for construction of this set of local ROBs are outlined below. When applicable, the matrices are stored as sparse matrices, the dot products are defined as sparse, and the least-squares solution is computed for a large, sparse, linear system of equations using scipy.sparse.linalg.lsqr. Ideally the code would store the sub-matrices separately, and the ROM code would be re-written to handle the new method. But using sparse implementation while leaving the rest of the method unchanged is a suitable halfway point to an ideal implementation of the method in a larger code, which will illustrate potential runtime improvement.

<u>Original Algorithm: Construct local ROBs in time and space</u>

Input: Snapshot matrix S, number $N_R \geq 1$ *and* $N_C \geq 1$ of spatial and temporal clusters

Output: Local ROBs $\bar{V}_j$, where $j = 1, \ldots, N_R$

1. Cluster the snapshot matrix S into number N_C of pre-computed temporal clusters along the columns of *S* to create submatrices X_i and store their centroids c_i
2. For $i = 1, \ldots, N_C$
    a. Cluster the submatrix X_i into number N_R of pre-computed spatial clusters along the rows of X_i to create submatrices X_{ij}
    b. Perform the SVD of each matrix
        i. $X_{ij} = U_{ij}\Sigma_{ij}V_{ij}^T$
    c. Choose the dimension of the ij-th ROB, k_{ij}
    d. Truncate the first few left singular vectors
        i. $\bar{V}_{ij} = U_{ij}(:, 0{:}\, k_{ij})$
    e. Arrange local basis column vectors $\bar{V}_i$ in vectorized form
        i. $\bar{V}_i = \begin{bmatrix} V_{i1} & 0 \\ 0 & V_{i2} \end{bmatrix}$

f. Perform local ROM updates with column-only local basis setup

Updated Algorithm: Construct sparse local ROBs in time and space with descending singular values

Input: Snapshot matrix S, number $N_R \geq 1 \; and \; N_C \geq 1$ of spatial and temporal clusters

Output: Local ROBs $\bar{V}_j$, where $j = 1, \dots, N_R$

1. Cluster the snapshot matrix S into number N_C of pre-computed temporal clusters along the columns of *S* to create submatrices X_i and store their centroids c_i
2. For $i = 1, \dots, N_C$
    a. Cluster the submatrix X_i into number N_R of pre-computed spatial clusters along the rows of X_i to create submatrices X_{ij}
    b. Perform the SVD of each matrix and save both *V*, the singular vectors, and Σ, the singular values.
        i. $X_{ij} = U_{ij}\Sigma_{ij}V_{ij}^T$
    c. Arrange local basis column vectors $\bar{V}_i$ in vectorized form
        i. $\bar{U}_i = \begin{bmatrix} U_{i1} & 0 \\ 0 & U_{i2} \end{bmatrix}, \Sigma_i = \begin{bmatrix} \Sigma_{i1} \\ \Sigma_{i2} \end{bmatrix}$
    d. Sort Σ_i in descending order and save the indices in vector of indices, *ind*
    e. Sort $\bar{U}_i$ according to the indices, *ind*
    f. Choose the dimension of the i-th ROB, k_i
    g. Truncate the first few left singular vectors
        i. $\bar{V}_i = \bar{U}_i(:, 0: k_i)$
    h. Perform local ROM updates using scipy sparse libraries

3. Results for Original Algorithm

The implementation of row-column clustering was found to substantially improve the quality of the ROBs for the Burger's equation in the proximity of the shock wave discontinuity when using a fixed number of basis vectors for each for row cluster equal to the number of basis vectors saved in the column-only solution. Figure 2 shows the damping of the Gibbs' oscillations for the case of a 10 column clusters on the data with an increasing a number of row clusters from 1 (column-only clustering case) to 10 row clusters.

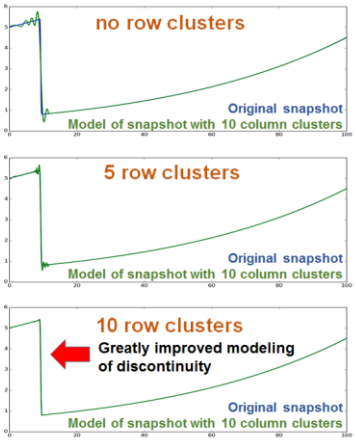
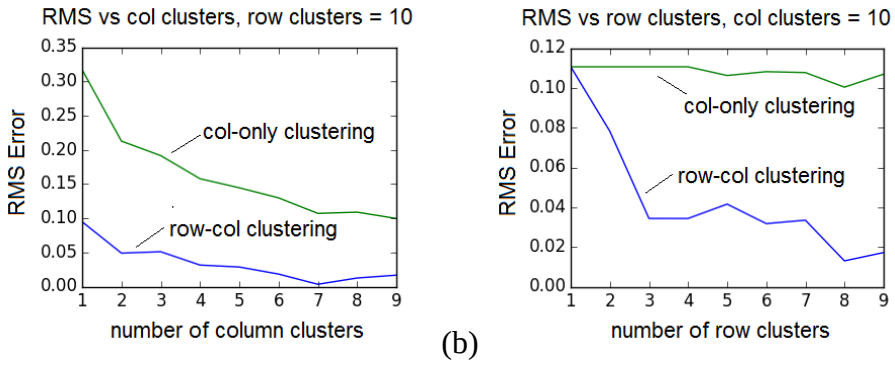


Figure 2: Reduction of Gibbs' oscillations due to row-column clustering

Additionally, Figure 3a illustrates the reduction in RMS error between the original snapshot and the projections as the number of column clusters are increased, given a fixed number of row clusters. Figure 3b illustrates the reduction in RMS error as the number of row clusters is increased, given a fixed number of column clusters. Row-column clustering is clearly an improvement for both conditions. As before, this method saves the same number of basis vectors for each row cluster that had been saved in the column-only clustering method.



(a) (b)

Figure 3: RMS error reduction with increasing spatial (row) or temporal (column) clusters given fixed number of clusters along the other dimension of the matrix

The figure below shows the change in modeling error for the Burger's equation test case after implementing temporal, spatial, and temporal/spatial clustering in the same online ROM code. Each figure its labeled with the number of spatial or temporal clusters used in implementation. For each dataset, the same number of left singular vectors, 10, was used from each singular value decomposition to construct the local bases.

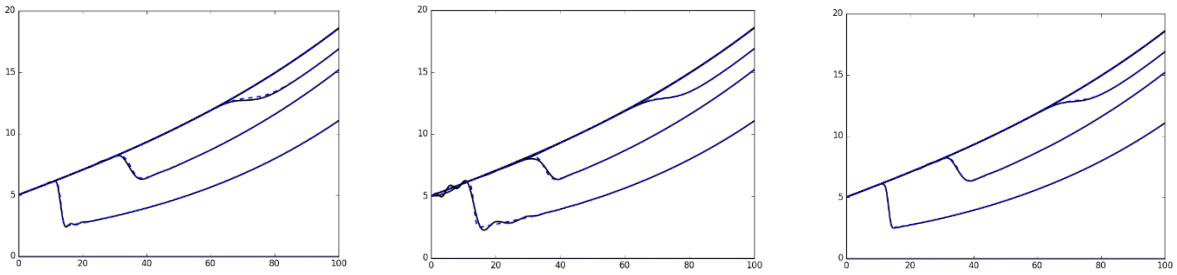


Figure 4: (a) 4 temporal clusters (b) 4 spatial clusters (c) 4 spatial and 4 temporal clusters

The combination of temporal and spatial clustering outperforms the other two clustering methods. The shockwave discontinuity is captured the best of the three methods. Also, the spatial-only clustering appears to perform far worse than the other two. However, in the spatial-only clustering, only 10 left singular vectors are saved across the entire matrix *S.* Therefore, this simulation is using much less data than the temporal-only and spatial-temporal simulations, which are saving 40 total left singular vectors over the entire matrix. This imbalance is handled with n the updated algorithm results to follow, which will show a back-to-back comparison of 10 vs. 10 total vectors saved.

At this point, the temporal clustering and spatial/temporal clustering are truly a back-to-back comparison using the same amount of saved information, only the implementation will be slower since the sparsity has not yet been handled appropriately for these cases. The spatial clustering-only in this case is also expected to be much worse because less information is being stored and utilized. These preliminary results confirm that not only is the projection error improved using spatial-temporal clustering, but the online model reduction works with the method, and is better able to capture the range of Burger's equation solutions while storing the same amount of data as temporal-only methods.

4. Results for Updated Algorithm

The previous results showed that improvement in the accuracy of the solution was possible, but it did not take advantage of matrix sparsity, and the method wasn't feasible for implementation in a code that calculated the SVD offline and truncated it online. So, in addition to these results, it was necessary to implement the method in a way that saved all singular vectors after performing the SVD, so that *V* could be truncated later.

The results show that very little change in the RMS error in observed in this implementation, in spite of the fact that the same number of *total* basis vectors are saved in this scenario. This is unexpected, since it was assumed that a singular vectors containing 30-98% zeros would not perform as well as the same-sized matrix of basis vectors with no sparsity, given the same number of total singular vectors saved. Therefore, the Figure 4 is a surprising and promising result for the usefulness and robustness of the method under different scenarios and specific implementations. Meanwhile, the figure also shows that runtime is considerably improved using this method, as expected.

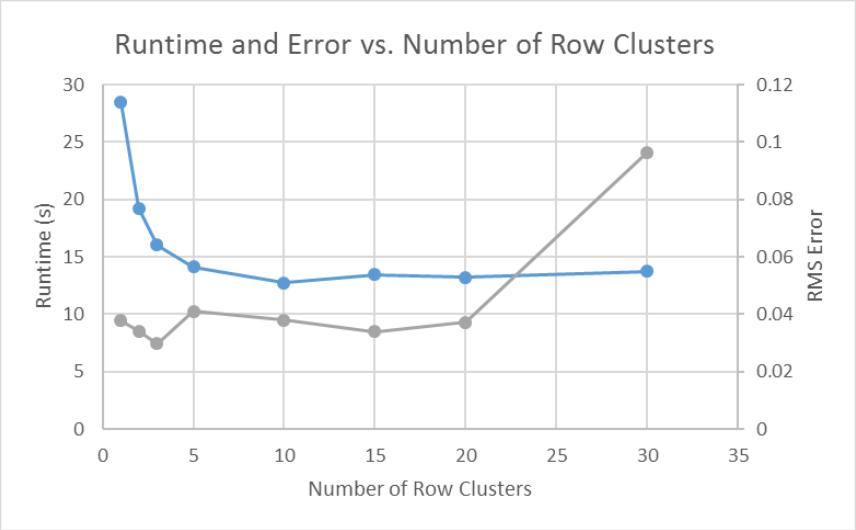


Figure 4: Runtime and RMS Error for python implementation that both takes advantage of matrix sparsity and truncates the matrix only during the online computation

5. Future work

The next step is to utilize the code as currently implemented in python to extend the method for use in the aero-f code. Many different variations are possible and choices need to be made for implementing the method efficiently in aero-f. The results so far show that the row-column clustering method will improve the code by improving the runtime, accuracy, or both, depending on the specific choices made in implementation.

References:

1. Amsallem, David, Matthew J. Zahr, and Charbel Farhat. "Nonlinear model order reduction based on local reduced- order bases." *International Journal for Numerical Methods in Engineering* 92.10 (2012): 891-916.
2. Washabaugh, Kyle, et al. "Nonlinear model reduction for CFD problems using local reduced-order bases." *42nd AIAA Fluid Dynamics Conference and Exhibit, Fluid Dynamics and Co-located Conferences, AIAA Paper*. Vol. 2686. 2012.