Hindawi
International Journal of Aerospace Engineering
Volume 2019, Article ID 8291616, 15 pages
https://doi.org/10.1155/2019/8291616

Research Article

Using the Nonuniform Dynamic Mode Decomposition to Reduce the Storage Required for PDE Simulations

Brenton T. Hall,[1,2] **Ching-Shan Chou**,[1] **and Jen-Ping Chen**[1]

[1]*Ohio State University, Columbus, OH 43210, USA*
[2]*Riverside Research, Beavercreek, OH 45431, USA*

Correspondence should be addressed to Brenton T. Hall; bhallresearch@gmail.com

Received 27 February 2018; Accepted 14 September 2018; Published 3 January 2019

Academic Editor: Angel Velazquez

Partial Differential Equation simulations can produce large amounts of data. These datasets are very slow to transfer, for example, from an off-site supercomputer to a local research facility. There have been many model reduction techniques that have been proposed and utilized over the past three decades. Two of the most popular techniques are the Proper Orthogonal Decomposition and Dynamic Mode Decomposition. Nonuniform Dynamic Mode Decomposition (NU-DMD) is one of the newest techniques as it was introduced in 2015 by Guéniat et al. In this paper, the NU-DMD's mathematics are explained in detail, and three versions of the NU-DMD's algorithm are outlined. Furthermore, different numerical experiments were performed on the NU-DMD to ascertain its behavior with respect to errors, memory usage, and computational efficiency. It was shown that the NU-DMD could reduce an advection-diffusion simulation to 6.0075% of its original memory storage size. The NU-DMD was also applied to a computational fluid dynamics simulation of a NASA single-stage compressor rotor, which resulted in a reduced model of the simulation (using only three of the five simulation variables) that used only about 4.67% of the full simulation's storage with an overall average percent error of 8.90%. It was concluded that the NU-DMD, if used appropriately, could be used to possibly reduce a model that uses 400 GB of memory to a model that uses as little as 18.67 GB with less than 9% error. Further conclusions were made about how to best implement the NU-DMD.

1. Introduction

Partial Differential Equation (PDE) simulations can generate hundreds of gigabytes of data. However, it is very slow to transfer these large datasets. The purpose of this article was to find an algorithm to accurately reduce the storage size requirement of these types of models. This, in effect, will increase the transfer rate of these datasets from, for example, an off-site supercomputer to a local research facility. A decomposition of the model can be used to reduce the storage size requirement of these models. There have been many different model reduction techniques proposed over the past three decades. Two of the most popular are the Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD).

POD is a model reduction technique, which can capture the relevant dynamics of a fluid flow problem [1]. POD has been used frequently over the past few decades to extract coherent flow structures from flow-field time snapshots. This is done by ranking the dominant structures of the flow by their energy, which is achieved "by diagonalizing the temporal and spatial correlation matrix calculated from the time step sequence" [2].

DMD is another model reduction technique, which also extracts coherent structures from flow-field time snapshots. Reference [2] states that the DMD algorithm was developed by Schmid in 2010 [3] with previous development of the Koopman analysis of a dynamic system done by Mezic [4] and Rowley et al. [5]. Schmid also applied the DMD to experimental data in [6]. Although DMD is closely related to POD,

DMD ranks the modes by their growth rates or amplitudes and not by ranking the orthogonal modes by the amount of energy [7, 8]. Furthermore, DMD looks at the spectral (or temporal) and spatial orthogonalities, which results in frequency information as well as the structural information [2]. The DMD modes oscillate at one frequency while the POD modes oscillate at multiple frequencies. This means that an advantage of DMD is that it clearly separates each structure spatially and spectrally, while the POD structures can be contaminated by other uncorrelated structures [9].

The Nonuniform DMD (NU-DMD), introduced in 2015 by Guéniat et al., is one of the newest model reduction techniques, and its application to reducing the storage size requirement of PDE simulations is the main topic of this article. As its name implies, the NU-DMD allows the use of nonuniform or random time snapshots enabling the user more flexibility. However, it is also capable of using uniform time samples, which is how it is used in the remainder of this article [10].

The outline for the remainder of this paper is as follows: Section 2 explains and develops the mathematics and algorithm of the NU-DMD; Section 3 outlines three versions of the NU-DMD algorithm that will be utilized in the remainder of the paper; Section 4 shows numerous numerical experiments that were conducted to explain, quantify, and differentiate these three algorithms; Section 5 discusses different approaches to implement the NU-DMD on a computational fluid dynamics (CFD) simulation and presents a reduced model for the CFD simulation; and, finally, Section 6 presents the conclusions.

2. Mathematics of the NU-DMD Algorithm

The NU-DMD algorithm introduced by Guéniat et al. [10] gives the reduced-model approximation at an arbitrary time $t_n = n\Delta t$, $n \in \mathbb{N}$ as

$$\mathbf{u}_{t_n} \approx \lambda_1^{t_n}\phi_1 + \lambda_2^{t_n}\phi_2 + \cdots + \lambda_{N_{\mathrm{md}}}^{t_n}\phi_{N_{\mathrm{md}}}, \quad \forall t_n \in \mathbb{R}, \tag{1}$$

where the set $\{\phi_k\}_{k=1}^{N_{\mathrm{md}}} \in \mathbb{C}^{n_p}$ are spatial modes (in the form of column vectors) and the set $\{\lambda_k\}_{k=1}^{N_{\mathrm{md}}} \in \mathbb{C}$ are temporal coefficients. N_{md} is the number of modes retained in the approximation.

2.1. The NU-DMD Decomposition. In order to calculate the above approximation, we must first obtain a data matrix K, which is known as the Krylov matrix. This matrix contains observation vectors from either a numerical simulation or an experiment. The Krylov matrix is defined as $K \equiv (\mathbf{u}_1 \cdots \mathbf{u}_N) \in \mathbb{R}^{n_p \times N}$, where n_p is the number of spatial points and N is the number of time steps or snapshots. In words, each column of K is a time step and contains every spatial point. The ordering of the spatial points is arbitrary as long as it is consistent among every time step.

To find the spatial modes and temporal coefficients, we first express the Krylov matrix as

$$K = M\Lambda + R_{\mathrm{es}}, \tag{2}$$

where $M \equiv (\phi_1 \cdots \phi_{N_{md}}) \in \mathbb{C}^{n_p \times N_{\mathrm{md}}}$ and $R_{\mathrm{es}} \in \mathbb{C}^{n_p \times N}$ and Λ contains the temporal coefficients and is defined as

$$\Lambda \equiv \begin{pmatrix} \lambda_1^1 & \lambda_1^2 & \cdots & \lambda_1^N \\ \lambda_2^1 & \lambda_2^2 & \cdots & \lambda_2^N \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N_{\mathrm{md}}}^1 & \lambda_{N_{\mathrm{md}}}^2 & \cdots & \lambda_{N_{\mathrm{md}}}^N \end{pmatrix} \in \mathbb{C}^{N_{\mathrm{md}} \times N}. \tag{3}$$

Note that due to the nature of matrix multiplication and the dimensions of M and Λ, the number N_{md} is arbitrary, and therefore, we are able to select the number of spatial modes. This gives us the ability to choose just how much memory to save or how much accuracy to retain. That is, if we keep more spatial modes, it will produce a more accurate approximation after reconstruction, but we will save less memory and vice versa.

2.2. Computing the Temporal Coefficients and Spatial Modes. Now that we have established a form to decompose the Krylov matrix, we can now begin to solve for the temporal coefficients contained in Λ and then the spatial modes contained in M. We solve for Λ by minimizing the residual matrix R_{es}. To do this, we first must define R_{es} in terms of variables that we know or for which we can solve. Provided that the Frobenius norm of the R_{es} is small, we can accurately approximate M by

$$M \approx K\Lambda^+, \tag{4}$$

where Λ^+ is the Moore-Penrose pseudoinverse of Λ. Now, we can substitute (4) into (2) to obtain

$$K \approx K\Lambda^+\Lambda + R_{\mathrm{es}}. \tag{5}$$

Now solving for R_{es} and factoring out K to make the future computation faster, we have

$$R_{\mathrm{es}} \approx K(I_N - \Lambda^+\Lambda), \tag{6}$$

where I_N is the identity matrix of size $N \times N$.

Finally, we can state the minimization problem as

$$\lambda \in \underset{\widehat{\lambda} \in \mathbb{C}^{N_{\mathrm{md}}}}{\arg\min} \left\| K\left(I_N - \Lambda\left(\widehat{\lambda}\right)^+ \Lambda\left(\widehat{\lambda}\right)\right)\right\|_F. \tag{7}$$

Note that $\Lambda(\widehat{\lambda})^+$ and $\Lambda(\widehat{\lambda})$ are functions of $\widehat{\lambda}$. Thus, we need to find Λ such that R_{es} is minimized in the Frobenius sense. This can be easily solved by algorithms such as the derivative-free Nelder-Mead method or, e.g., the fminsearch function in MATLAB, which uses the Nelder-Mead simplex direct search method (see https://www.mathworks.com/help/matlab/ref/fminsearch.html). Lastly, we can easily solve for the spatial modes with simple matrix multiplication using the equation

$$M = K\Lambda^+. \tag{8}$$

2.3. Efficiency of the Minimization Problem. The minimization problem can be made more efficient when $N < n_p$ by evaluating the QR decomposition of K such that $K = QR$ with $Q \in \mathbb{R}^{n_p \times N}$ and $R \in \mathbb{R}^{N \times N}$. Therefore, the minimization problem reformulates as

$$\lambda \in \underset{\widehat{\lambda} \in \mathbb{C}^{N_{\text{md}}}}{\arg\min} \left\| R\left(I_N - \Lambda\left(\widehat{\lambda}\right)^{+} \Lambda\left(\widehat{\lambda}\right)\right) \right\|_F, \tag{9}$$

and it now only involves matrices of size $N \times N$ (notice the product of $\Lambda^{+}\Lambda$ is also $N \times N$), which will greatly improve efficiency because there are fewer floating point operations (flops) to perform per iteration in a minimization algorithm.

Now, the NU-DMD algorithm's efficiency can be further improved upon by reducing the number of spatial points n_p. As stated by Guéniat et al., "a space-decimated Krylov matrix may capture the temporal features of the flow." Therefore, we can select a small number of the spatial points $\tilde{n}_p$ and still obtain a good approximation. The new set of spatial points makes the observables $\tilde{\boldsymbol{u}}_n \in \mathbb{R}^{\tilde{n}_p \times 1}$ and the Krylov matrix $\tilde{K}_{\tilde{n}_p} \in \mathbb{R}^{\tilde{n}_p \times N}$. Now, the minimization problem is

$$\lambda \in \underset{\widehat{\lambda} \in \mathbb{C}^{N_{\text{md}}}}{\arg\min} \left\| \tilde{K}_{\tilde{n}_p} \left(I_N - \Lambda\left(\widehat{\lambda}\right)^{+} \Lambda\left(\widehat{\lambda}\right)\right) \right\|_F. \tag{10}$$

This minimization problem will be more efficient than what we originally had in (7) as there will be far fewer flops to perform for the matrix multiplication. However, we can still use the QR decomposition of $\tilde{K}_{\tilde{n}_p}$ when $N < \tilde{n}_p$. Therefore, we now have

$$\lambda \in \underset{\widehat{\lambda} \in \mathbb{C}^{N_{\text{md}}}}{\arg\min} \left\| \tilde{R}_{\tilde{n}_p} \left(I_N - \Lambda\left(\widehat{\lambda}\right)^{+} \Lambda\left(\widehat{\lambda}\right)\right) \right\|_F. \tag{11}$$

Although this minimization problem still only involves matrices of size $N \times N$, it will actually be more time efficient than (9). However, it can cause the accuracy of the reduced-model to decrease. These two artifacts are illustrated in Section 4.

2.4. Decreasing the Number of Spatial Points. The problem of reducing the number of spatial points can be formulated as finding the set of indices $\mathcal{J} = \{j_1, j_2, \ldots, j_{\tilde{n}_p}\}, 1 \le j_i \in \mathbb{N} \le n_p$ such that Λ calculated via (10) is a good approximation to Λ calculated via (7). To be able to solve this, we need to determine the dynamics of the flow, find spatial points where the dynamics of the flow are similar, and then only keep these spatial points.

To determine the dynamics of a single spatial point, we just need to calculate the Fourier time series of each spatial point. That is, if the observables are stated in the form

$$\boldsymbol{u}^{(i)}(t) = \left(u^{(i)}(t_1), u^{(i)}(t_2), \ldots, u^{(i)}(t_N)\right)^T \in \mathbb{R}^N, \tag{12}$$

then we need to find the Fourier time series in the form

$$u^{(i)}(t) \approx \sum_{k=1}^{N} a_k^{(i)} \exp\left(\sqrt{-1}\omega_k t\right), \quad \forall 1 \le i \le n_p. \tag{13}$$

To reiterate, each row of the Krylov matrix K corresponds to a single spatial point and each column of K represents a single point in time. Therefore, we need to find the Fourier time series of each row of K. Note that since these time series are not dependent on each other, they could be computed in parallel.

If we have data that is uniform in time, we can simply utilize the Discrete Fourier Transform (DFT) or, in practice, the Fast Fourier Transform (FFT) to obtain the needed Fourier time series of each spatial point. Then, to find spatial points with similar dynamics, we need to find the dominant Fourier spectrum of the obtained time series. To this aim, we simply need to cluster the collection of Fourier coefficients $\{\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, \ldots, \boldsymbol{a}^{(n_p)}\}$ (note that the FFT outputs a set of complex numbers, wherein we must then find the modulus of each complex number to find each corresponding coefficient). The standard K-means algorithm was used for this purpose in this work.

The K-means algorithm is a method of vector quantization. The algorithm "finds cluster centroids that minimize the distance between data points and the nearest centroid." Essentially, K-means finds a collection of k vectors such that a data vector can be reconstructed in such a way that the error is minimized [11].

Finally, we simply need to retain the spatial points that correspond to the coefficient vectors that are closest to their cluster centroid. Then, we can continue to find the spatial modes and the temporal coefficients with the space-decimated Krylov matrix $\tilde{K}_{\tilde{n}_p}$.

If the data is nonuniform in time, we cannot use the FFT. Although we do not present it in this article, [10] proposes a technique that approximates the set of the Fourier coefficients $\{\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, \ldots, \boldsymbol{a}^{(n_p)}\}$ and then continues with the above algorithm. Also according to [10], it is also possible to use a suboptimal relaxation technique for use with nonuniform in time data. But since PDE simulations yield a uniform in time dataset, the FFT is used to find the time series of each spatial point in this article.

3. Three Versions of the NU-DMD Algorithm

As discussed in Section 2, there are different methods of performing steps in the NU-DMD algorithm. This chapter will outline the three algorithms that are used throughout the remainder of this paper.

Algorithm 1 is the simplest version of the NU-DMD. However, it is also the most inefficient as shown in Section 4.1. As stated in step 1 below, the minimization problem uses the entire Krylov matrix K, which, as discussed earlier, uses more flops per iteration of the minimization method.

Algorithm 2 utilizes the QR decomposition. This increases the efficiency of the NU-DMD algorithm as the

Require: Observables contained in matrix K of size $n_p \times N$
1: $\mathrm{F}(\Lambda) \leftarrow \|K(I_N - \Lambda^+\Lambda)\|_F$ ▷Initialize function of norm of R_{es}
2: $g \leftarrow$ guess matrix of size $N_{\text{md}} \times N$ ▷Intial guess of Λ
3: $\Lambda \leftarrow$ minimize $\mathrm{F}(\Lambda)$ using g ▷Use minimization method to find Λ
4: $\mathrm{M} \leftarrow K\Lambda^+$ ▷Calculate the spatial modes

Algorithm 1: NU-DMD for uniform in time data (using just K).

Require: Observables contained in matrix K of size $n_p \times N$
1: $R \leftarrow \mathrm{qr}(K)$ ▷QR decomposition
2: $\mathrm{F}(\Lambda) \leftarrow \|R(I_N - \Lambda^+\Lambda)\|_F$ ▷Initialize function of norm of R_{es}
3: $g \leftarrow$ guess matrix of size $N_{\text{md}} \times N$ ▷Intial guess of Λ
4: $\Lambda \leftarrow$ minimize $\mathrm{F}(\Lambda)$ using g ▷Use minimization method to find Λ
5: $\mathrm{M} \leftarrow K\Lambda^+$ ▷Calculate the spatial modes

Algorithm 2: NU-DMD for uniform in time data (using QR on K).

minimization problem only involves matrices of size $N \times N$. And as shown in Section 4, the accuracy of Algorithm 1 is retained in Algorithm 2.

Algorithm 3 selects a subset of spatial points before continuing with Algorithm 2. And as shown in Section 4, this increases the speed at which a reduced model can be computed. However, it is also shown in Section 4 that this increased efficiency costs accuracy. It is also not known beforehand what specified distance, referred to in step 5 below, should be used to retain accuracy.

4. Numerical Experiments

The following numerical experiments were done on the data generated by the exact solution to the following unsteady 2D advection-diffusion problem:

$$\frac{\partial u}{\partial t} + p\frac{\partial u}{\partial x} + q\frac{\partial u}{\partial y} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial y^2}, \tag{14}$$

where a and b are diffusion coefficients and p and q are advection coefficients. The exact solution to the above PDE is

$$u(x, y, t) = \frac{1}{4t+1}\ \exp\left[-\frac{(x - pt - 0.5)^2}{a(4t+1)} - \frac{(y - qt - 0.5)^2}{b(4t+1)}\right]. \tag{15}$$

More details of this problem can be found in [12].

Notes: these numerical experiments were all done using MATLAB_R2015a on a MacBook Pro with a 2.7 GHz Intel i7 processor with access to 256 KB of L2 cache, 4 MB of L3 cache, and 8 GB of 1333 MHz DDR3 RAM. The coefficients were set at (unless otherwise stated) $a = b = 0.01$ and $p = q = 0.8$. The percent error for the entirety of this paper is defined as $(100 \times \|K - M\Lambda\|)/\|K\|$. The norm used in this equation will be labeled either F-norm (Frobenius norm), infinity norm, and 2-norm (L^2 norm).

4.1. Differences between Algorithms 1, 2, and 3. First, here are the results for a base run of the NU-DMD using Algorithm 1, which uses the entire matrix K. Note that the exact solution ran from time 0 to 1 second in 21 discrete steps.

As seen from Table 1, it took two minutes and eleven seconds to implement Algorithm 1 on the model of the advection-diffusion problem on a 40×40 grid with only 21 time steps. However, 12 spatial modes were used to obtain an average percent error of only 1.4179%. Algorithm 2 was used on the same grid and number of time steps while still choosing 12 spatial modes. Note that Algorithm 2 uses the QR decomposition of K.

Algorithm 2 was run on the same model with the results in Table 2. It ran for almost 45 seconds making it far more efficient than Algorithm 1. This is expected as the minimization problem depends greatly on the size of the matrices. Replacing K with R allows for far fewer flops per iteration of the minimization method. Furthermore, the replacement did not cause much difference in the percent errors of the two reduced models. Algorithm 3 was then used to see if the execution time could be further decreased.

Algorithm 3 requires the input of the number of clusters for the K-means algorithm and the distance between the cluster centroids and the points found by the K-means algorithm. For this run, one cluster and a specified distance (shown in Algorithm 3) of 0.06 was used to select points closest to that one centroid. This reduced the number n_p in K from 1600 to 196 ($\tilde{K}_{\tilde{n}_p}$ is 196×21) causing the QR decomposition to be computed more quickly and allowing the minimization problem to be solved faster.

As seen in Table 3, Algorithm 3 ran in just 22.15 seconds, which was about half the run time of Algorithm 2. However, it did come with some cost as now the average percent error is 8.8949%. Therefore, there are trade-offs. We could increase

Require: Observables contained in matrix K of size $n_p \times N$

1: $A \leftarrow |\mathrm{fft}(\mathrm{eachrowof K})|$ ▷Find Fourier coefficients
2: D $\leftarrow$ kmeans(A) ▷K-means clustering & save distances in D
3: ii $\leftarrow$ 1 ▷counter for next loop
4: **for** i $\leftarrow$ 1 to column length of D **do** ▷Obtain $\tilde{K}_{\tilde{n}_p}$
5: **if** i$^{\text{th}}$ row of D < specified distance **then**
6: ii$^{\text{th}}$ row of $\tilde{K}_{\tilde{n}_p} \leftarrow$ i$^{\text{th}}$ row of K
7: ii $\leftarrow$ ii+ 1
8: **end if**
9: **end for**
10: $\tilde{R}_{\tilde{n}_p} \leftarrow \mathrm{qr}(\tilde{K}_{\tilde{n}_p})$ ▷QR decomposition
11: $\mathrm{FF}(\Lambda) \leftarrow \|\tilde{R}_{\tilde{n}_p}(I_N - \Lambda^+\Lambda)\|_F$ ▷Initialize function of norm of R_{es}
12: $g \leftarrow$ guess matrix of size $N_{\mathrm{md}} \times N$ ▷Intial guess of Λ
13: $\Lambda \leftarrow$ minimize F(Λ) using g ▷Use minimization method to find Λ
14: M $\leftarrow K\Lambda^+$ ▷Calculate the spatial modes

Algorithm 3: NU-DMD for uniform in time data (using QR on $\tilde{K}_{\tilde{n}_p}$).

Table 1: Algorithm 1 results (40×40 grid, $N = 21$, $N_{\mathrm{md}} = 12$).

(a)

Elapsed time (sec)	Grid size n-by-n	Δt	# of snapshots	# of spatial modes (N_{md})
131.73	40×40	0.0513	21	12

(b)

Errors	Average percent error	F-norm percent error	Inf-norm percent error	2-norm percent error
	1.4179%	1.0874%	1.7779%	1.3884%

(c)

Reduced memory to % of original	KB to store M and Λ	KB to store original model	KB to store spatial Modes (M)	KB to store temporal coefficients (Λ)
57.8930%	151.969	262.5	150.0	1.969

Table 2: Algorithm 2 results (40×40 grid, $N = 21$, $N_{\mathrm{md}} = 12$).

(a)

Elapsed time (sec)	Grid size n-by-n	Δt	# of snapshots	# of spatial modes (Nmd)
44.98	40×40	0.0513	21	12

(b)

Errors	Average percent error	F-norm percent error	Inf-nor percent error	2-norm percent error
	1.4161%	0.99133%	1.8595%	1.3974%

(c)

Reduced memory to % of original	KB to store M and Λ	KB to store original model	KB to store spatial modes (M)	KB to store temporal coefficients (Λ)
57.8930%	151.969	262.5	150.0	1.969

the specified distance in Algorithm 3 (the distance between points and the cluster centroid), increasing the number n_p in $\tilde{K}_{\tilde{n}_p}$, which should improve the error but increase the run time. And, of course, we could do this the opposite way and get worse error but a faster execution time. Alternatively, Algorithm 1 or 2 could be used if more accuracy is needed and execution time is not as important.

It is important to note that all three algorithms result in the same memory size for the reduced model. The differences between the three algorithms only affect the execution time and accuracy and not the memory size of the reduced model. It is easy to conclude from Figure 1 that Algorithm 2 significantly reduces the computation time of Algorithm 1 but still retains the accuracy unlike Algorithm 3.

4.2. Spatial Modes and Memory. N_{md} can be chosen by the user to affect the error, run time, and/or memory usage in the desired manor. Figure 2 shows the linear relationship the memory size has with the number of spatial modes N_{md}. The percent storage is approximately $100 \times N_{\mathrm{md}}/N$ for $n_p >> N > N_{\mathrm{md}}$. That is, when the number of spatial points n_p is much greater than the number of time steps N, the size of M becomes the dominant factor over the size of Λ. Furthermore, since K has dimensions of $n_p \times N$ and M has dimensions of $n_p \times N_{\mathrm{md}}$, it is the ratio N_{md}/N that matters.

Table 3: Algorithm 3 results (40×40 grid, $N = 21$,$N_{md} = 12$).

(a)

Elapsed time (sec)	Grid size n-by-n	Δt	# of snapshots	# of spatial modes (Nmd)
22.15	40×40	0.0513	21	12

(b)

Errors	Average percent error	F-norm percent error	Inf-norm percent error	2-norm percent error
	8.8949%	04.8459%	14.5559%	7.2828%

(c)

Reduced memory to % of original	KB to store M and Λ	KB to store original model	KB to store spatial modes (M)	KB to store temporal coefficients (Λ)
57.8930%	151.969	262.5	150.0	1.969

4.3. The Effect of Spatial Modes on Run Time and Accuracy. Λ is a matrix of size $N_{md} \times N$. Therefore, when the number of spatial modes N_{md} is large, the minimization problem will take longer to solve and vice versa. However, increasing the N_{md} should increase the accuracy of the reduced model and vice versa. Therefore, when choosing the number of spatial modes N_{md}, it important for the user to be aware of these trade-offs. The following graphs in Figure 3 show the relationship between run time, accuracy, and the number of spatial modes. The collected data was found by using Algorithm 2, a 40×40 grid, $\Delta t = 0.0513$, and 21 snapshots.

4.4. Efficiently Using the NU-DMD. As has been stated before, the minimization problem in the NU-DMD algorithm depends mostly on the number of time snapshots N (provided the QR decomposition is used on K). Therefore, if the NU-DMD was run on data with a large number of time steps, it would take a very long time to solve the minimization problem. For example, Algorithm 2 was run on data generated by the same exact solution to the 2D advection-diffusion problem used previously from time 0 to 2 seconds split into 40 discrete time steps.

The run shown in Table 4 took almost 9 minutes to complete. Since the minimization greatly depends on N, the Krylov matrix can be split into submatrices, and then the NU-DMD can be used on these submatrices. This will greatly decrease the computational time as shown in Table 5.

As seen from the tables, the computational time improved drastically from using the full Krylov matrix to using two sub-Krylov matrices. In fact, it saved 489 seconds. However, using submatrices did come with some cost as the average percent error actually increased from 7.5848% to 20.0322%. However, this discrepancy is not as drastic when there are more snapshots to split up or when Δt is smaller, as will be discussed in the next subsection. This increase in error could also be the cause of NU-DMD not being able to fit the data across only a portion of the dominant frequency period. However, this may not be the issue as the NU-DMD algorithm is essentially a data-fitting algorithm and is not for calculating the physical frequencies. More research on this subject could be performed to answer this question, but that is beyond the scope of this article.

It also takes slightly less memory to store the reduced model, because it uses fewer temporal coefficients. That is, Λ is a matrix of size $N_{md} \times N$. So when both dimensions were cut in half, the size of Λ was cut in half, but now there are two of them. This means that when both dimensions are scaled down appropriately, the reduced model will require fewer overall temporal coefficients. Furthermore, notice that the memory required to save the spatial modes for both reduced models is the same because the N_{md} was cut in half when the snapshot matrix K was cut in half. Also, note that now there are multiple sub-Krylov matrices, which means each submodel can be reduced in parallel, further decreasing the computational time.

4.5. The Effect of Δt on Accuracy. As noted previously, the time step size Δt has a big effect on the accuracy of the NU-DMD algorithm. To illustrate this, the following graph (Figure 4) was made using Algorithm 2 on the same PDE as before, but with the following specifications: 200×200 grid, 200 snapshots split up into 20 submatrices with 10 snapshots each, 4 spatial modes per submatrix, and a varying final time (starting at zero) to keep the same number of 200 snapshots with a varying Δt.

Based on Figure 4, it is easy to conclude that if a Δt of 0.04 or less is used, the reduced model should have an average percent error of about 10% or less. However, this could be a disadvantage because the PDE simulation might need to have more time steps to be able to obtain a reduced model with good accuracy.

It is also important to note that the choice of Δt is related to the frequencies that constitutes the decomposition function. The smaller Δts could correlate with the dominant frequency of the function, which would explain the decrease in error as sampling frequency increases.

4.6. The Significance of the NU-DMD. To illustrate the power of the NU-DMD, it was run again on data generated from the exact solution of the advection-diffusion problem (with $a = 2$, $b = 0.01$, $p = 1$, and $q = 0.6$) with the results displayed in Table 6. One thousand time steps were used. These time steps were split up into 20 submatrices, each with fifty snapshots. A 200×200 grid was used with a Δt of 0.01. Time is from 0 to 9.99 seconds.

Table 6 shows that the reduced model is now only 6.0074% of the size of the original model. And, after it is reconstructed, it only has an average percent error of 7.2385%.

Since this model only uses megabytes of memory, this particular example is not that significant. However, it is clear that if this technique was applied to a large dataset that uses gigabytes or terabytes of data, the result would be quite meaningful, which will be shown in Section 5.

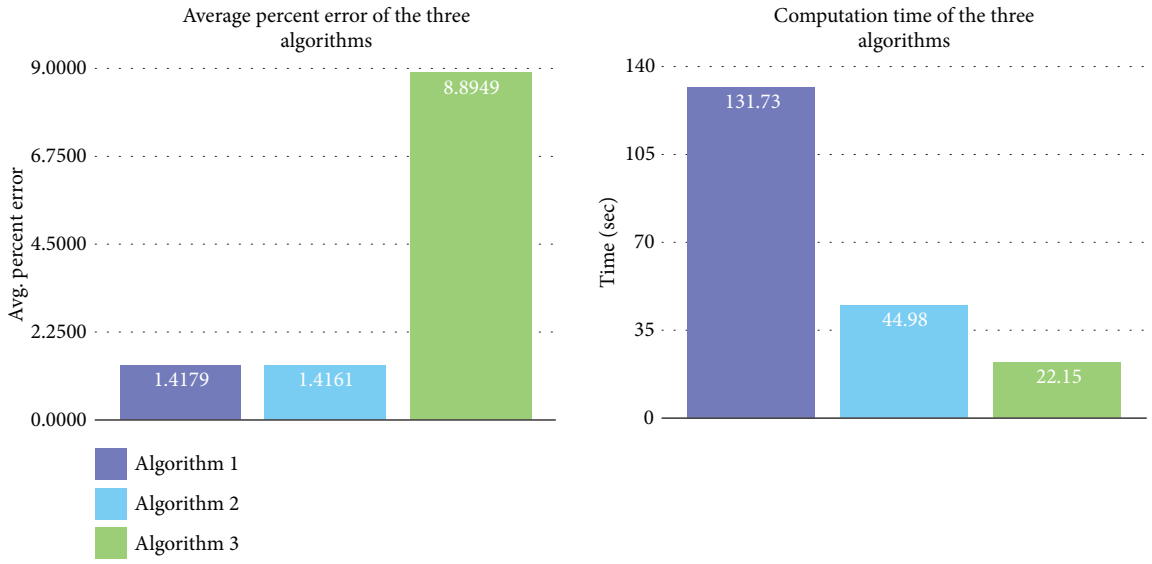

Figure 1: Trade-offs of the three versions of the NU-DMD.

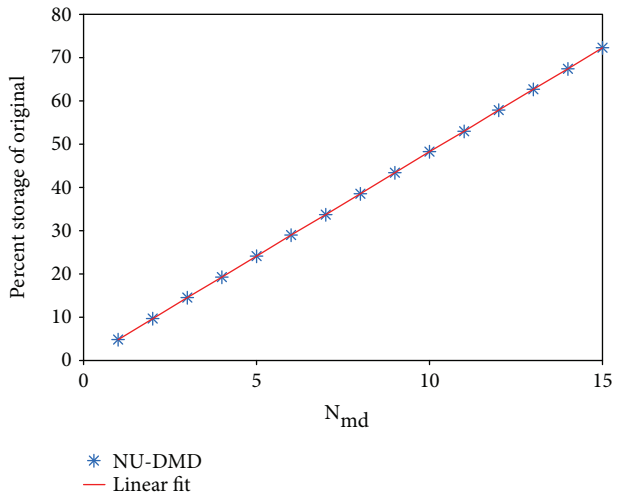

Figure 2: Percent memory vs. N_{md}.

5. Reducing the Storage Size of a Computational Fluid Dynamics Simulation

In this section, we will look at reducing the storage size needed for a particular computational fluid dynamics (CFD) simulation. The dataset from this CFD simulation was generated by Chen et al., as seen in [13]. They simulated a NASA single-stage compressor rotor that consists of 36 blade passages, shown in Figure 5. TURBO, which is "a physics-based simulation tool for multistage turbomachinery" [14], was used to solve the Navier-Stokes equations (details of which can be found in [15]) in a full-annulus model.

Each blade passage (as seen in Figure 5) was modeled using a curvilinear grid of dimensions $151 \times 71 \times 56$ (in general, $\vec{x} \times \vec{y} \times \vec{z}$). Therefore, there are 21.6 million grid points for the whole simulation of the rotor. In the simulation, each revolution of the rotor is divided into 3600 iterations. C.M. Chen et al. ran their simulations for at least four revolutions, generating at least 400 GB of data with 576 time steps. As done in their paper, we will be looking at time steps that are every 25 simulation iterations. This simulation yielded five variables for each spatial point: the density, the three components of velocity, and the stagnation energy. For more details about this simulation, see [13].

In the following reductions of the above model, the focus will be on one blade passage, which has 600,376 grid points, over 100 time steps. Each variable for one blade passage uses 0.4473150 GB (or about 458.0505 MB) per 100 time steps, since double precision was used. Algorithm 2 was used for these reductions. Algorithm 2 was chosen, because it is efficient, but it still obtains good accuracy as was seen in Section 4.

The Nelder-Mead method code was provided by Akiva Wernick of the Mechanical and Aerospace Engineering Department of The Ohio State University. Parameters for the Nelder-Mead method were set at the following: the reflection coefficient = 1.0, the expansion coefficient = 2.0, and the contraction coefficient = 0.5. More information about the Nelder-Mead method can be found in [16–18].

Algorithm 2 was written in Fortran 90 and was run on the Oakley cluster on the Ohio Super Computer [19] with access to sufficient memory. Appropriate subroutines were used from Intel's Math Kernel Library (see https://software.intel.com/en-us/intel-mkl).

5.1. Approaches to Reducing the Three-Dimensional CFD Model. The NU-DMD algorithm was first implemented on the density variable in all three dimensions over all 100 time steps.

Remarkably, the model can be reduced to only 5% of the original with an average percent error of 6.40% and only

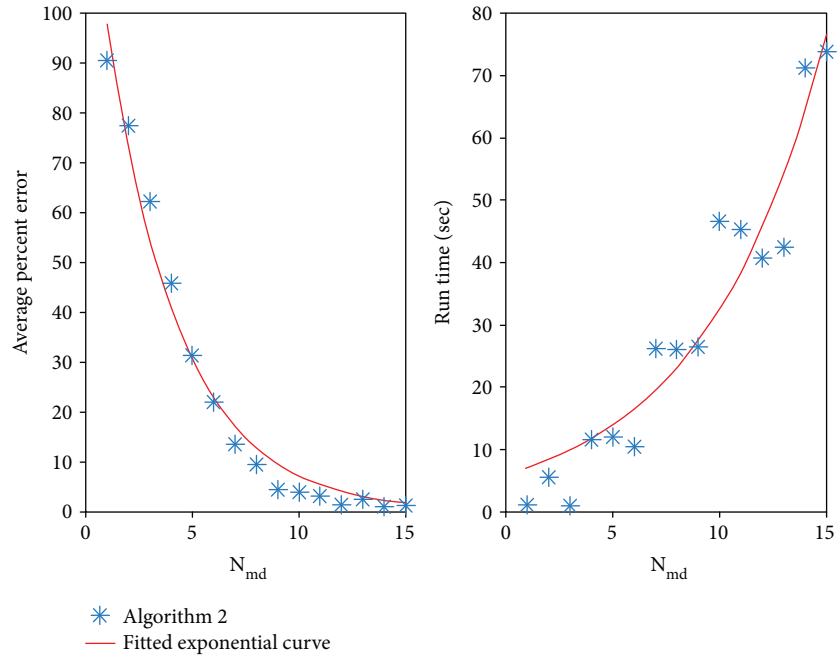

Figure 3: Relationship of spatial modes with percent error and run time.

Table 4: Algorithm 2 results (40×40 grid, $N = 40, N_{md} = 12$).

(a)

Elapsed time (sec)	Grid size n-by-n	Δt	# of snapshots	# of spatial modes (Nmd)
507.02	40×40	0.0513	40	12

(b)

Errors	Average percent error	F-norm percent error	Inf-norm percent error	2-norm percent error
	7.5848%	5.7018%	9.1534%	7.8992%

(c)

Reduced memory to % of original	KB to store M and Λ	KB to store original model	KB to store spatial modes (M)	KB to store temporal coefficients (Λ)
30.75%	151.75	500.0	150.0	3.750

Table 5: Algorithm 2 results (40×40 grid and 40 time steps).

(a)

Elapsed time (sec)	Grid size n-by-n	Δt	# of snapshots	# of spatial modes (Nmd)
18.07	40×40	0.0513	20	6

(b)

Errors	Average percent error	F-norm percent error	Inf-norm percent error	2-norm percent error
	20.0322%	14.0517%	26.2222%	19.8227%

(c)

Reduced memory to % of original	KB to store M and Λ	KB to store original model	KB to store spatial modes (M)	KB to store temporal coefficients (Λ)
30.75%	151.75	500.0	150.0	1.875

2.99% error with respect to the F-norm. Clearly, this feat would be even more significant on a larger dataset.

As shown in Table 7, using more spatial modes increases the accuracy until 6 or more spatial modes are used. Here, from 5 to 6 and 6 to 8 spatial modes, the error begins to increase slightly because the Λ matrix is becoming larger, and the Nelder-Mead method is unable to optimize the solution as well. This is caused by the Nelder-Mead method's sensitivity to the initial simplex [17]. If a better initial simplex for this problem is utilized, a higher number of spatial modes would yield more accurate results. Also, note that the efficiency decreases as more spatial modes are used, which is clearly expected.

Instead of using all three dimensions, the NU-DMD algorithm can be run on two-dimensional planes. The following reductions will look at 2D planes $\vec{y} \times \vec{z}$ for the i-th plane in the $\vec{x}$ direction. Therefore, there are only 3976 grid points. However, this has to be done for every i-th plane. So in this

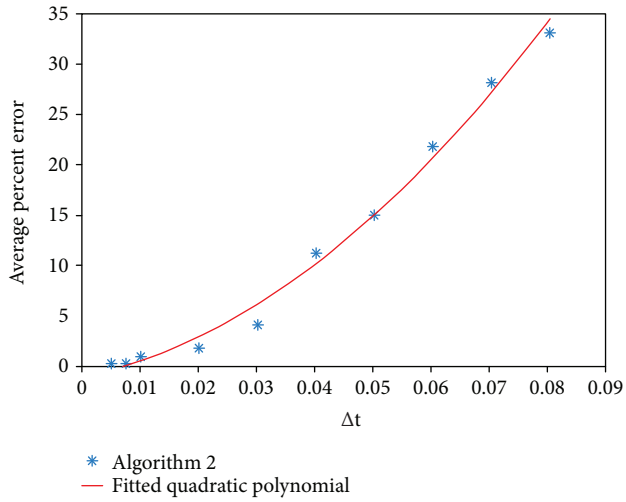

Figure 4: Average percent error vs. Δt.

Table 6: Algorithm 2 results (200×200 grid, 50 snapshots per submatrix, and 3 spatial modes per submatrix).

(a)

Elapsed time (sec)	Grid size n-by-n	Δt	# of snapshots	# of spatial modes (Nmd)
182.50	200×200	0.01	50	3

(b)

Errors	Average percent error	F-norm percent error	Inf-norm percent error	2-norm percent error
	7.2385%	5.8061%	8.0524%	7.8569%

(c)

Reduced memory to % of original	KB to store M and Λ	KB to store original model	KB to store spatial modes (M)	KB to store temporal coefficients (Λ)
6.0074%	18.333	305.176	18.311	0.02289

case, the NU-DMD would have to be done 151 times. However, this process could be done in parallel. The same 100 time steps were used for these reductions.

Tables 8–11 show that depending on which i-th plane's model is reduced, the error varied within 7% or less. For the $i = 2$ plane, an excellent average percent error of about 0.58% was obtained using five spatial modes. But for the $i = 50$ and $i = 150$ planes, the NU-DMD yielded reduced models of about 6.7% and 5.3% (using five spatial modes), respectively.

Table 12 looks at using the NU-DMD on each plane, as done previously, but now adding each plane's reduced model together to make a three-dimensional, reduced model that can be compared to the reduced model displayed in Table 7.

The reduced model referred to in Table 12 has a F-norm percent error of 3.08% and average percent error of 8.43% for two spatial modes. The compilation of 2D reduced models into a 3D reduced model actually uses slightly more memory than the reduced model referred to in Table 7 with corresponding number of spatial modes. This is because there are 151 Λ matrices of size $N_{\text{md}} \times N$ instead of just one Λ matrix. Furthermore, each Λ matrix has to be computed, which is why execution time took 52.6 times longer than when all three dimensions were used at once when using two spatial modes (see Table 7).

The reduced model that uses three spatial modes in Table 12 has slightly more error than the one with two spatial modes by about one percentage point, despite it taking more than three times to compute. As stated previously, the Nelder-Mead method is very sensitive to the initial simplex [17]. The same initial simplex was used to calculate the temporal coefficients for each plane, which appears to have caused a decrease in accuracy even though an increase in accuracy is expected going from two to three spatial modes.

The reduced model referred to in Table 7 with two spatial modes has lower accuracy than the one in Table 12 with two spatial modes. However, three spatial modes increased the accuracy of the model in Table 7 to an average percent error of 6.53% and an F-norm percent error of only 3.05%. Furthermore, it was computed in only 17.1 seconds.

The method of reducing models' 2D planes was also used for 2D planes of $\vec{x} \times \vec{y}$. This decreased execution time and memory usage because there are only 56 planes in the $\vec{z}$ direction. And, as shown in Table 13, the accuracy was also decreased. But, the models shown in Table 7 still take less time to compute, use less memory, and are more accurate.

From the comparisons made in this section, it is concluded that even though some 2D plane reduced models may have very desirable percent errors, using the NU-DMD on all three dimensions at once is considerably faster and more accurate than using it on each 2D plane individually and then compiling a 3D reduced model. It is also important to note that the differing errors that are obtained from using the NU-DMD on different planes of the dataset could be caused by the fact that there may be different dominant frequencies (though the physical frequencies will be in all planes just with different amplitudes) for each one of these planes of the dataset.

There are also two other approaches that can be taken to reduce the CFD model. One is that the NU-DMD can be run on all five variables in all three dimensions concurrently. However, this would only work if all five variables used the same initial guess of the simplex for the Nelder-Mead method. The other approach is to use the NU-DMD over fewer time steps, which will be discussed in a later section. First, the models for stagnation energy and velocity in the $\vec{x}$ direction are reduced using all three dimensions.

5.2. Reducing the Models of Velocity in the $\vec{x}$ Direction and Stagnation Energy. The NU-DMD was used to reduce the models of velocity in the $\vec{x}$ direction and the stagnation energy. These models were reduced over the same 100 time

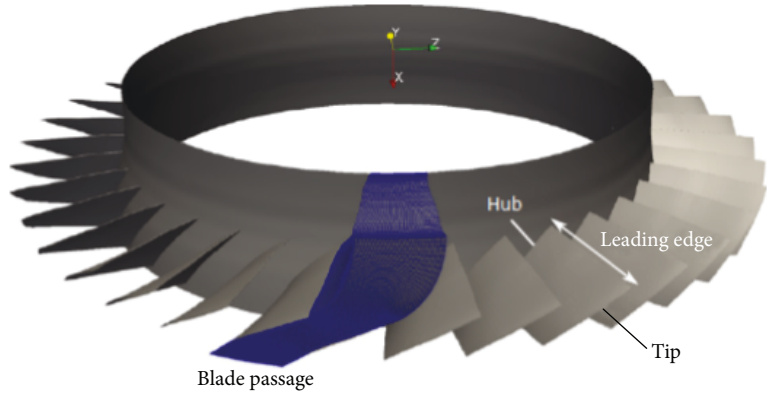

Figure 5: The simulated compressor rotor [13].

Table 7: 3D model of density over 100 time steps.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
1	7.59	64.7423	74.3623	55.1223	1.0001667
2	9.73	9.3271	7.1774	11.4767	2.000333
3	17.13	6.5317	3.0597	10.0037	3.000410
4	62.68	6.5031	3.0411	9.9651	4.000666
5	97.33	6.4022	2.9853	9.8192	5.000833
6	337.66	6.4282	3.0241	9.8324	6.000999
8	574.96	6.7636	3.5847	9.9425	8.001332

Table 8: Density in an $\overrightarrow{y} \times \overrightarrow{z}$ plane with $i = 2$.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	2.40	0.5929	0.4390	0.7468	2.050302
3	14.00	0.5901	0.4331	0.7471	3.075453
4	37.71	0.5930	0.4383	0.7476	4.100604
5	162.43	0.5830	0.4291	0.7369	5.125755

Table 9: Density in an $\overrightarrow{y} \times \overrightarrow{z}$ plane with $i = 50$.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	3.85	7.1248	4.3820	9.8675	2.050302
3	9.41	6.9378	4.3060	9.5695	3.075453
4	38.78	6.9824	4.3016	9.6632	4.100604
5	123.26	6.7262	4.2227	9.2297	5.125755

Table 10: Density in an $\overrightarrow{y} \times \overrightarrow{z}$ plane with $i = 100$.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	2.42	3.7632	2.4212	5.1054	2.050302
3	14.80	3.7570	2.3934	5.1206	3.075453
4	67.12	3.6847	2.3862	4.9832	4.100604
5	109.56	3.4531	2.3342	4.5721	5.125755

steps as in the previous reductions. All three dimensions and all 100 time steps were used concurrently to reduce the model. One variable's model was reduced at a time.

Table 14 shows that the reduced model for velocity in the $\overrightarrow{x}$ direction is not very accurate with an average percent error of about 17.83% and a F-norm percent error of about

Table 11: Density in an $\overrightarrow{y} \times \overrightarrow{z}$ plane with $i = 150$.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	2.23	5.3280	4.8502	5.8058	2.050302
3	20.59	5.3456	4.8469	5.8443	3.075453
4	66.56	5.3282	4.7881	5.8683	4.100604
5	89.71	5.3220	4.8125	5.8316	5.125755

Table 12: Densities of 3D reduced model built from all 151 2D reduced models.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	511.36	8.4327	3.0859	13.7795	2.050302
3	1666.46	9.4095	4.3802	14.4389	3.0754527

Table 13: Densities of 3D reduced model built from all $|z| = 56$ 2D reduced models.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	194.99	9.1723	6.6620	11.6826	2.018655
3	637.07	7.4423	3.0774	11.8072	3.027982

Table 14: Velocity in the $\overrightarrow{x}$ direction reduced 3D model over 100 time steps.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	13.16	18.6508	10.1081	27.1936	2.000333
3	30.12	18.5834	10.1261	27.0406	3.000450
4	85.30	18.5809	9.9811	27.1806	4.000666
5	150.11	17.9162	9.8789	25.9535	5.000833
8	484.50	17.8334	9.8407	25.8261	8.001332

Table 15: Stagnation energy reduced 3D model over 100 time steps.

Spatial modes	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
2	12.54	6.0028	6.5172	5.4884	2.000333
3	22.44	2.6860	1.4611	3.9109	3.000500
4	46.04	2.6675	1.4694	3.8655	4.000666
5	77.55	2.6752	1.4517	3.8987	5.000833
8	1268.21	2.6188	1.4110	3.8267	8.001332

9.84%, even with using eight spatial modes. However, this accuracy could be increased with the use of more spatial modes, but more memory and time will be needed. And, as expected, the accuracy and execution time increased with the increased number of spatial modes.

Unlike velocity in the $\overrightarrow{x}$ direction, the stagnation energy's reduced model has a very desirable accuracy as seen in Table 15, using as few as three spatial modes with average percent errors under 2.7% and F-norm percent errors under 1.5%. The general trend of the accuracy is that it increased with increased number of spatial modes. Also, as expected, the execution time increases as the number of spatial modes increases.

5.3. Splitting Up the Time Steps. There is one more approach to using the NU-DMD. As discussed in Section 4, the Krylov matrix can be split up into submatrices, and the NU-DMD can then be used on each one of the submatrices. This saves time as the minimization problem greatly depends on the number of time steps. As was shown, however, this efficiency came at the cost of some accuracy for the particular PDE dataset that was discussed in Section 4. In this section, using this approach on the CFD dataset that has been used throughout this section will be discussed. Furthermore, it will be assessed whether this method will be more efficient and/or accurate with this dataset.

The following reductions were done over the same 100 time steps of the CFD model, but they were split up into four sets of 25 time steps. Only three spatial modes were used for each set of 25 time steps, which means the model will not be reduced to about 2% of the original as in the previous reductions, but it will be reduced to about $100 \times 2/25 = 8\%$ of the

original. Therefore, these models should be compared to corresponding models that used eight spatial modes and 100 time steps.

For the density variable (see Table 16), this method yielded an average percent error of 6.7993% (only about 4.1% w.r.t the F-norm), which is slightly higher than the 6.7636% error shown in Table 7. Furthermore, the execution time was only 2.1336 seconds, which is obviously far better than 574.96 seconds (found in Table 7). Therefore, since these two models are using the same amount of memory, this model would be more advantageous to use because of the speed at which the model can be computed.

However, a lower average percent error of 6.5317% (and 3.0597% in the F-norm) was obtained using three spatial modes over 100 time steps as displayed in Table 7. Furthermore, this model only took 17.13 seconds to compute and only used 3% of the memory of the original. Therefore, using all 100 time steps concurrently to reduce the model for density was more advantageous than the approach used to obtain the reduced model referred to in Table 16.

For velocity in the $\vec{x}$ direction (see Table 17), this method obtained an average percent error of 18.3%, which is slightly higher than the 17.8% average percent error found in Table 14 (with the model that used 8% of the memory of the original). However, this approach only took 2.1 seconds to compute, while the model in Table 14 took 484.5 seconds to compute. But, similar accuracy was obtained using less memory than in the approach for Table 14.

For stagnation energy (see Table 18), this method actually yielded nearly the same percent error as its 100 time step counterpart (shown in Table 15 with eight spatial modes), only differing in the F-norm and Inf-norm percent error categories by less than 0.02 percentage points. Furthermore, splitting up the time steps for this variable increased the computation of the reduced model by 623 times. Therefore, it is clear that splitting up the time steps is more advantageous for stagnation energy when the same amount of memory is utilized. However, the other reduced models that use less memory in Table 15 have comparable percent errors and only use at most 78 seconds to compute the reduced model.

It is clear from this section that the approach of using all available time steps concurrently may or may not be more advantageous than splitting up the time steps. There are advantages and disadvantages to each one depending on the dataset that is to be reduced, as well as whether time, memory, and/or accuracy is more valuable to the user.

5.4. Best Results for the Blade Passage. As shown in the previous sections, it is better to reduce the models of each variable over all the time steps that are available, and in some cases, it may be better to reduce the model of a variable over smaller time intervals. The preference of either one largely depends on which result needs to minimized the most. This section will outline one of the best reduced models (in terms of accuracy, memory usage, and execution time) of the blade passage (for density, velocity in $\vec{x}$, and energy) based on the results in the previous sections (note that better results may be obtained by using more/less spatial modes, changing the number of time steps in a set, and/or using a more accurate guess of the initial simplex).

For density, the best overall result is shown in Table 7, where five spatial modes were used. This model used all 100 time steps at once to calculate the reduced model, but it took only 97.33 seconds to compute. It used only 5.000833% of the original 458.0505 MB and had an average percent error of 6.40% and 2.99% error in the F-norm.

The best overall result for velocity in the $\vec{x}$ direction is shown in Table 14, where five spatial modes were used. This model used all 100 time steps at once to calculate the reduced model. It took 85.3 seconds to compute, used only 5.000833% of the original 458.0505 MB, and had an average percent error of 17.92% with only 9.88% error in the F-norm.

Lastly, the best overall result for stagnation energy is shown in Table 15, where two spatial modes were used. This model used all 100 time steps at once to calculate the reduced model, took only 46.04 seconds to compute, used only 4.000666% of the original 458.0505 MB, and had an average percent error of only 2.67% and only 1.47% error with respect to the F-norm.

One of the best overall reduced models shown in Table 19, based on the results in Section 5, is only about 4.67% of the original. Furthermore, it was computed in under four minutes. The overall average percent error is 8.90%, and the average of the F-norm percent errors is only 4.78%.

If this model was extended to all 36 blade passages, the original model would be reduced from about 48.31 GB to only 1.34 GB, with error under 9%. Furthermore, if more time steps were added to the model and/or if the other two dimensions of velocity were included, the decrease in memory usage would be even more significant.

As stated at the beginning of this section, the simulations ran by C.M. Chen et al. generated at least 400 GB of data. Therefore, if the NU-DMD was used to compute a reduced model, the memory usage could possibly be reduced from at least 400 GB to as little as 18.67 GB.

6. Conclusion

This article explained and analyzed the Nonuniform Dynamic Mode Decomposition. Section 1 explained why model reduction techniques are important and gave reasons why the NU-DMD was the chosen technique. Section 2 explained and gave detailed information about the mathematics of the decomposition and algorithm. Section 3 outlined the three versions of the NU-DMD algorithm that were used in this paper and summarized the advantages and disadvantages of each algorithm. Section 4 analyzed different numerical experiments to ascertain the behavior of the NU-DMD's mathematics and algorithm. Finally, in Section 5, the NU-DMD was applied to a computational fluid dynamics simulation. I also outlined different approaches to using the NU-DMD on the five-variable, three-dimensional dataset.

The NU-DMD was shown to be a viable model reduction technique for PDE simulations. Furthermore, different

Table 16: 3D reduced model of density over four sets of 25 time steps.

Time steps	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
1–25	0.5529	6.9688	3.4445	10.4931	8.000333
26–50	0.5189	6.5938	6.1846	7.0030	8.000333
51–75	0.5289	6.2985	3.1240	9.4730	8.000333
76–100	0.5329	7.3359	3.6188	11.0528	8.000333
Total & averages	2.1336	6.7993	4.0930	9.5055	8.000333

Table 17: 3D reduced model of velocity in the $\vec{x}$ direction over four sets of 25 time steps.

Time steps	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
1–25	0.4869	17.6218	10.6049	24.6386	8.000333
26–50	0.6319	16.8684	8.2704	25.4663	8.000333
51–75	0.4899	18.4419	9.5824	27.3013	8.000333
76–100	0.5519	20.3665	11.2662	29.4668	8.000333
Total & averages	2.1606	18.3247	9.9310	26.7183	8.000333

Table 18: 3D reduced model of stagnation energy over four sets of 25 time steps.

Time steps	Execution time (sec)	Average % error	F-norm % error	Inf-norm % error	Reduced memory as % of original
1–25	0.5479	3.1926	1.9898	4.3953	8.000333
26–50	0.5509	2.9139	1.5442	4.2837	8.000333
51–75	0.5929	3.2361	1.7891	4.6831	8.000333
76–100	0.5659	4.7379	4.6580	4.8177	8.000333
Total & averages	2.0346	3.520125	1.429890	3.977997	8.000333

Table 19: One of the best reduced models based on results for the blade passage.

Variable	Execution time (sec)	Average % error	F-norm % error	Memory (GB) reduced/original	Memory as % of original
Density	97.33	6.40	2.99	0.02237/0.44731	5.000833
Velocity $\vec{x}$	85.30	17.92	9.88	0.02237/0.44731	5.000833
Energy	46.04	2.67	1.47	0.01790/0.44731	4.000666
Reduced model	228.67	8.90	4.78	0.06263/1.34193	4.667444

approaches to using the NU-DMD on a dataset, such as the CFD simulation in Section 5, were discussed. Running the NU-DMD on one variable over all available time steps concurrently (as seen in Sections 4.1, 5.1, 5.2, and 5.4) proved to be the most accurate and to use the least amount of memory. However, the method of splitting up the time steps also yielded accurate results and proved the most efficient approach, but more memory must be used to store the resulting reduced model. Calculating reduced models of each 2D plane and then compiling them into one 3D reduced model was shown to be less accurate and less computationally efficient. Finally, the approach of reducing the 3D model over all five variables concurrently was not attempted because each variable's minimization problem could need a completely different initial simplex such that the problem is solved accurately.

The three versions of the NU-DMD that were discussed in this paper all have their advantages and disadvantages. Algorithm 1 is the most basic version of the NU-DMD, but it is also the most computationally inefficient. Algorithm 2 greatly increased the computational efficiency while still retaining the accuracy of Algorithm 1. However, Algorithm 2 is still computationally expensive when applied to datasets with very large number of time steps. Algorithm 3 is the most computationally efficient version of the NU-DMD discussed in this paper, but it also yielded the highest errors. Because of its accuracy and computational efficiency, Algorithm 2 was chosen to be written in Fortran for the reduction of the computational fluid dynamics simulation discussed in Section 5.

A new approach to using the NU-DMD was explored in Sections 4.4 and 5.4. This method was to split the Krylov matrix up into several smaller Krylov matrices. This process

allowed the reduced model to be computed more quickly than its all-available time step counterpart (which used the same amount of memory); however, the accuracy was decreased. Still, for the stagnation energy variable discussed in Section 5, the accuracy only decreased by less than 0.02 percentage points (in the F-norm and Inf-norm percent error categories). It was also 623 times quicker to compute the reduced model, but it used slightly more memory by splitting up the times. The example in Section 4.4 showed that this approach actually increased the average percent error from about 7.6% to about 20%. Therefore, the conclusion whether or not to use this new approach of implementing the NU-DMD depends on the dataset that the NU-DMD is processing and whether storage, execution time, or error needs to be minimized.

The power of the NU-DMD is on display in Section 4.6 and throughout Section 5. The NU-DMD was able to take a 1000 time step model of the advection-diffusion problem discussed in Section 4 (which was modeled on a 200×200 grid) and reduce its storage size to 6.0075% of the original storage size. What is even more significant is that the NU-DMD was able to take a CFD simulation of a rotor blade passage and reduce its storage size to about 4.67% of the original storage size with an overall average percent error (over three variables: density, velocity in $\overrightarrow{x}$, and stagnation energy) of 8.90% and only 4.78% error with respect to the Frobenius norm. The average percent errors of this reduced model for each separate simulation variable (density, velocity in $\overrightarrow{x}$, and stagnation energy) were 6.40%, 17.92%, and 2.67%, respectively. With respect to the Frobenius norm, the percent errors were 2.99%, 9.88%, and 1.47%, respectively. This means that, if used appropriately, the NU-DMD could be used to possibly reduce a model that uses at least 400 GB of memory to a model that uses as little as 18.67 GB of memory with less than 9% error. Further analysis of the NU-DMD could yield a better understanding of how best to implement the NU-DMD, which then could produce reduced models with even less error.

The NU-DMD could yet be improved to increase efficiency, accuracy, and usability. Algorithm 3 requires a specified distance that is used to select points near clusters after the K-means algorithm step. However, the optimal distance is not known before the NU-DMD is employed. This could cause the NU-DMD to yield undesirable results. A method for identifying the optimal distance (in regard to efficiency, accuracy, and the dataset itself) still needs to be developed. A process to identify the optimal initial simplex for the Nelder-Mead method for a given dataset should be researched and developed. This will allow the NU-DMD to yield more accurate and consistent results. Furthermore, the Nelder-Mead method could be replaced with another minimization method such as a gradient descent algorithm. This also could result in the NU-DMD being more consistent and accurate as well as more computationally efficient.

Data Availability

The data that was used for this study was obtained from the authors of [13].

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This paper is based on my (Brenton T. Hall's) master's thesis, which was written in partial fulfillment of the requirements for the degree of Master of Mathematical Sciences (MMS) at the Ohio State University. I thank Gregory Heinlein for all his help with accessing the CFD simulation data. I thank my sister Jeanna Hall for helping me check the grammar in this paper. And last, but certainly not least, I thank Akiva Wernick for allowing me use his Nelder-Mead Fortran code.

References

[1] S. S. Ravindran, "A reduced-order approach for optimal control of fluids using proper orthogonal decomposition," *International Journal for Numerical Methods in Fluids*, vol. 34, no. 5, pp. 425–448, 2000.

[2] Q. Zhang, Y. Liu, and Y. Wang, "The identification of coherent structures using proper orthogonal decomposition and dynamic mode decomposition," *Journal of Fluids and Structures*, vol. 49, pp. 53–72, 2014.

[3] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[4] I. Mezić, "Spectral properties of dynamical systems, model reduction and decompositions," *Nonlinear Dynamics*, vol. 41, no. 1-3, pp. 309–325, 2005.

[5] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, vol. 641, p. 115, 2009.

[6] P. J. Schmid, "Application of the dynamic mode decomposition to experimental data," *Experiments in Fluids*, vol. 50, no. 4, pp. 1123–1130, 2011.

[7] D. Duke, J. Soria, and D. Honnery, "An error analysis of the dynamic mode decomposition," *Experiments in Fluids*, vol. 52, no. 2, pp. 529–542, 2012.

[8] M. Ali, A. Pandey, and J. Gregory, "Dynamic mode decomposition of fast pressure sensitive paint data," *Sensors*, vol. 16, no. 6, 2016.

[9] T. W. Muld, G. Efraimsson, and D. S. Henningson, "Flow structures around a high-speed train extracted using proper orthogonal decomposition and dynamic mode decomposition," *Computers & Fluids*, vol. 57, pp. 87–97, 2012.

[10] F. Guéniat, L. Mathelin, and L. R. Pastur, "A dynamic mode decomposition approach for large and arbitrarily sampled systems," *Physics of Fluids*, vol. 27, no. 2, article 025113, 2015.

[11] A. Coates and A. Y. Ng, *Learning Feature Representations with K-Means*, Springer LNCS, 2nd edition, 2012.

[12] Z. F. Tian and Y. B. Ge, "A fourth-order compact ADI method for solving two-dimensional unsteady convection-diffusion problems," *Journal of Computational and Applied Mathematics*, vol. 198, no. 1, pp. 268–286, 2007.

[13] C.-M. Chen, S. Dutta, X. Liu, G. Heinlein, H.-W. Shen, and J.-P. Chen, "Visualization and analysis of rotating stall for transonic jet engine simulation," *IEEE Transactions on*

Visualization and Computer Graphics, vol. 22, no. 1, pp. 847–856, 2016.

[14] J. P. Chen, M. D. Hathaway, and G. P. Herrick, "Prestall behavior of a transonic axial compressor stage via time-accurate numerical simulation," *Journal of Turbomachinery*, vol. 130, no. 4, article 041014, 2008.

[15] J. P. Chen and D. L. Whitfield, *Navier-Stokes Calculations for the Unsteady Flowfield of Turbomachinery*, AIAA, 1993.

[16] A. F. Ali and M. A. Tawhid, "A hybrid cuckoo search algorithm with Nelder Mead method for solving global optimization problems," *Springerplus*, vol. 5, no. 1, p. 473, 2016.

[17] M. Baudin, *Nelder-Mead User's Manual*, 2010.

[18] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[19] O. S. Center, "Ohio Supercomputer Center," 1987, http://osc.edu/ark:/19495/f5s1ph73.