Proceedings of Global Power and Propulsion Society
ISSN-Nr: 2504-4400
GPPS Hong Kong 2023
October 17-19, 2023
www.gpps.global

GPPS-TC-2023-0146

Improving an Algorithm for Assessing the Completion of Internal, Unsteady Flow Simulations using Spatial Downsampling

Amelia George
Turbomachinery and Unsteady Flows Research Group
georg11h@uwindsor.ca
Windsor, Ontario, Canada

Jeff Defoe
Turbomachinery and Unsteady Flows Research Group
jdefoe@uwindsor.ca
Windsor, Ontario, Canada

Abstract

In practical computational fluid dynamics (CFD), a reduction in flow time required to be solved is extremely beneficial in reducing the amount of time and computational resources required for the computation. The authors have previously introduced a convergence algorithm based on dynamic mode decomposition (DMD) that can determine the completion of an unsteady CFD flow, reducing the overall amount of flow time required to be solved. However, this algorithm itself can be expensive, and was only tested on cases with less than half a million cells. The original algorithm become too expensive on larger datasets. This paper introduces an updated algorithm along with a method for spatially downsampling the CFD data to make the execution of the convergence algorithm feasible on practical CFD, which could have up to hundreds of millions of cells. The updated algorithm and spatial downsampling method is applied to two CFD cases - a turbulent flow over a cylinder and a rotor in duct case - where the full solution is known. It is able to correctly determine the amount of flow time required for the spatio-temporal content stops changing meaningfully, requiring much fewer computational resources than the original version of the algorithm.

1 Introduction

With the ubiquity of time-resolved or unsteady computational fluid dynamics (CFD) in engineering applications, it is no surprise that there is also increasing demand in reducing their computational cost. These computations need to be highly accurate to resolve both deterministic and stochastic time-varying content, and can be very expensive. Each time step must be resolved before the next can be computed, and while the use of parallel computing can reduce run times, there is a limit to efficient parallelization. Thus, reducing the overall flow time required to be resolved is beneficial.

In a previous paper (George and Defoe, 2023), the authors introduced an algorithm based on dynamic mode decomposition (DMD) that could determine when a flow field's spatio-temporal content has stopped changing significantly, meaning that continuing the computation was no longer necessary. DMD is a model-free algorithm that extracts spatio-temporal patterns from any data. To apply it to CFD cases, DMD requires snapshots of data at evenly-spaced time intervals. For a 2D velocity field with n spatial points and $m+1$ time steps, the snapshots are constructed as follows (Krake et al., 2021):

$$\begin{bmatrix} | & | & & | \\ x_0 & x_1 & \cdots & x_m \\ | & | & & | \end{bmatrix} = \begin{bmatrix} u_1(t_0) & u_1(t_1) & & u_1(t_m) \\ v_1(t_0) & v_1(t_1) & \dots & v_1(t_m) \\ \vdots & \vdots & \ddots & \vdots \\ u_n(t_0) & u_n(t_1) & & u_n(t_m) \\ v_n(t_0) & v(t_1) & \dots & v(t_m) \end{bmatrix} \tag{1}$$

where x_i are the input to the DMD algorithm, and $u_n(t_m)$ and $v_n(t_m)$ are the velocity components at the mth time step and nth spatial point

The convergence algorithm is run successively as new flow time (snapshots) are computed, and determines that the flow has become periodic when the DMD outputs converge to within specified tolerances. In turbomachinery applications,

this algorithm is useful for simulations of stable operating points for turbomachines operating with flow distortion, requiring a full-wheel, unsteady approach. Even a single fan stage in this sort of computation can require a computational grid of up to 100 million cells and several rotor revolutions of flow-time (Gunn and Hall, 2014). Often a significant portion of that flow time is included "just in case," to be sure that the flow has become fully periodic. Such computations would benefit greatly from a shorter flow time requirement, if it could be determined that adding more time would not add new information. This is exactly what the convergence algorithm presented in George and Defoe (2023) does, and it was demonstrated to reduce the amount of flow time required for a developing periodic flow with separation by 8% compared to traditional methods in which measures of periodicity are manually assessed. However, while the convergence algorithm has been demonstrated on 2D CFD cases cell counts below half a million, as-is, it becomes prohibitively expensive in practical CFD applications, such as that mentioned above. Specifically, the compute time and memory requirement for the pseudo-inverse calculation step, step 4 in Algorithm 5 in Appendix A, is $O(n^3)$ in general (Courrieu, 2005), though some attempts are being made to make the computation more efficient (Stanojević et al., 2022). In the current Python-based implementation, this step also cannot be parallelized. In this paper, we do not focus on the mathematical and code details of these computations, but instead of higher-level concepts of what data is needed and how it should be used – how to reduce the size of the snapshot matrix. This enables the algorithm to be a feasible tool for typical CFD cases. The aim is thus to lower the computational cost of using the DMD-based convergence assessment algorithm regardless of how the details of the approach are implemented.

Several others have introduced methods to reduce the input size of data into DMD in an effort to make its use more tenable. These can be divided into two categories: reducing the temporal resolution, and reducing the spatial resolution. Li et al. (2022) conducted a study on the effects of temporal resolution and range on the stabilization of the observed modes. They showed that the optimal temporal resolution should be tailored according to the dynamics of interest. Effectively, the extent to which the temporal snapshots can be downsampled depends on the maximum frequency of interest;, this is discussed further in Section 2. Tu (2013) spatially downsampled their data (direct numerical simulation (DNS) of a 3D supersonic flow over a compression ramp) in the spanwise direction, while keeping the spatial resolution in the other 2 dimensions unchanged. This is effective but requires knowing that flow field variation in one direction is expected to be much less significant than in the other directions. Clainche et al (2015) took a different approach by taking a subdomain of their DNS computational domain to reduce computational costs. This approach is not mutually exclusive with spatial downsampling, as one could apply spatial downsampling to a subdomain. The common ground between all above-mentioned methods, however, is that they require the user to decide by how much or where their data should be downsampled. Thus, they must have at least some understanding of the spatio-temporal content of the flow they are studying a priori, since choices which exclude important flow content may lead to inaccurate results.

In contrast, the key contribution of this paper is a robust algorithm for a method of reducing the input data to the DMD, without affecting the algorithm's ability to accurately determine when an unsteady CFD computation can be terminated, and with minimal user input required.

The remainder of this paper is organized as follows. First, we present an updated convergence algorithm which can reduce the number of temporal snapshots included in each iteration of the algorithm. This is discussed in Section 2. Next, in Section 3, we introduce an approach to downsample the snapshots spatially while retaining the ability of the algorithm to determine when sufficient flow time has elapsed. Two test cases on which the improved algorithm and spatial downsampling approach are applied are introduced in Section 4. Section 5 presents the results of the algorithmic additions for the test cases.

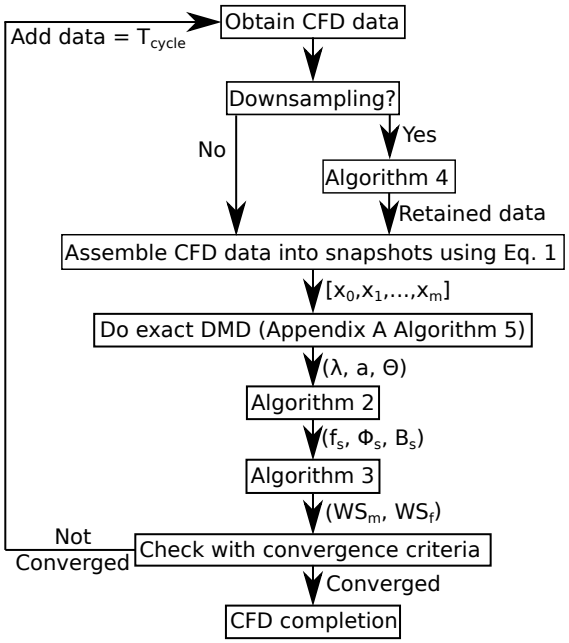
2 Improved DMD-Based Convergence Algorithm

The improved algorithm is shown in Algorithm 1. There is an optional step of including Algorithm 4 if downsampling is required, which will be discussed in more detail in Section 3. The details of Algorithms 2 and 3 are also included. Note that Algorithm 2 is identical to the beginning of the original convergence algorithm from George and Defoe (2023). Steps 3(b) and 4(c) from Algorithm 4 are also taken from the original algorithm, with an adaptation in step 3(b) where WS_m is calculated. A scaling factor q^* is added to generalize the calculation to work with spatial downsampling, as will be discussed in Section 3. In the case where no spatial downsampling is done, $q^* = 1$. Convergence is reached when $WS_f \leq 0.0001$ and $WS_m \leq 0.01$; this indicates that no further CFD flow time needs to be computed.

The improvements to the approach are in Algorithm 3. The first improvement is an updated method to determine the period of a fundamental cycle T_{cycle} of the flow field. The previous method of calculating the fundamental cycle length involved selecting the lowest non-zero frequency from the frequencies associated with the modes constituting 99% of the cumulative modal amplitude. However, if the initial execution of the algorithm occurs with snapshots spanning a time period significantly less than the real fundamental period, the DMD may identify non-physical low frequencies, leading to the determination of an artificially long cycle duration. This is mitigated by setting a lowest allowed frequency (Algorithm 3 step 2) based on the time span of data in the current iteration of Algorithm 1.

The next improvement is the automation of the determination of the number of snapshots added in the subsequent iteration. Increments of additional snapshots covering flow time corresponding to 0.5, 1, 1.5, and 2 times the fundamental period were assessed and it was determined that adding 1 period of snapshots provides the best balance between robustness and efficiency. This is shown in Algorithm 1.

Algorithm 1 Convergence Algorithm for Unsteady CFD



Third, the criterion for dropping earlier snapshots from the data considered is updated (Algorithm 3, step 3). Including more snapshots than necessary causes the DMD execution to become more expensive. Five cycles is found to be more than sufficient to resolve all spatio-temporal components of the flow (only two to three cycles are normally needed), and to take advantage of any information in an initial transient that may aid in earlier convergence. This method simply removes earlier snapshots as they become no longer useful. No temporal downsampling occurs in this process.

Determining what time interval to use between DMD snapshots (ΔT) depends on whether there is a priori knowledge of the expected flow field behavior. The highest frequency that can be resolved is

$$f_{max} = \frac{1}{2\Delta T} \tag{2}$$

so if the user has no a priori information about the flow, then no temporal downsampling should occur and ΔT should be equal or close to the time step size of the computation (Δt) being analyzed. However, if the highest relevant frequency is known or can be estimated, then the user can downsample their temporal data as required to reduce the computational requirements of the convergence algorithm.

Lastly, the convergence criteria for the weighted average of fractional change for the frequency and modes have been adjusted slightly (less strict for modes and more strict for frequency).

3 Spatial Downsampling

While the improved algorithm from Section 2 ensures the number of snapshots used is bounded, this does not help with the problem of very large snapshot matrices since the number of spatial points is normally much larger than the number of snapshots. Thus, a much more significant way to reduce the amount of data handled by the DMD is to reduce the number of spatial points analyzed. In this section we introduce a spatial downsampling approach which requires no a priori knowledge of the expected nature of the flow field. This is effective because, in many cases, the cell density required to resolve the flow field accurately in CFD is larger than that required by DMD, and also because the distribution of cells in a CFD computation is most often not optimal, especially when structured grids are used: many cells are located in regions where the flow field gradients are small.

The overall approach is to include spatial downsampling of CFD data. Referring back to Algorithm 1, this is done via the addition of a preprocessing step before snapshot assembly (Algorithm 4). The idea of Algorithm 4 is to use the root mean squared (RMS) variation in velocity over T_{cycle} to obtain an estimated field that is used to identify points more likely to be important for capturing temporal variations, and thus correctly capturing the DMD frequencies. Note that the algorithm is written for a 2D velocity field, but the generalization to 3D is trivial. Based on the cumulative probability function of the normal distribution, a probability of retention is applied to every point in the flow field using the RMS velocity. The range of RMS velocity is mapped linearly to ± 2 standard deviations from the mean. The choice of mapping to ± 2 standard deviations (rather than more or fewer standard deviations) is that the chosen number provides a good balance between

Algorithm 2 Identification of modes with $\geq 99\%$ of total scaled mode amplitude

1. Denote r as the number of eigenvalues with non-negative imaginary parts. Select those eigenvalues (λ_r), as well as their associated amplitudes and modes (a_r, Θ_r).
2. Compute the scaled modes matrix $A = \lambda_r^{m+1} a_r \Theta_r$. This gives $A = \begin{bmatrix} | & | & & | \\ A_0 & A_1 & \cdots & A_r \\ | & | & & | \end{bmatrix}$.
3. For each column A_i in A, compute the Frobenius norm where $\|A_i\|_F = \left[\sum_j \left|A_{i,j}\right|^2\right]^{1/2}$ to get the scaled modes vector of length r, where j denotes the indexing for the rows of A. Represent the scaled modes vector as Φ.
4. Define the normalized scaled mode vector $B = \frac{\Phi}{\sum_{i=0}^{r} \Phi_i}$, and define B^* is B sorted in descending order.
5. Define the cumulative sum of the elements of B^*, $B^*_{cs,i} = \sum_{i=0}^{k} B^*_i$. Identify the smallest number of modes k for which $B^*_{cs,k} \geq 0.99$ (99% of the total amplitude of all scaled modes). Denote this sorted list of selected scaled modes as $\Phi_s = \begin{bmatrix} \phi_0 & \phi_1 & \cdots & \phi_k \end{bmatrix}$ and their normalized value as B_s. Find the associated eigenvalues (λ_s).
6. For every $\lambda_{si} = a_i + b_i j$, find the frequencies $f_{sj} = \frac{\tan^{-1}\left(\frac{b_i}{a_i}\right)}{2\pi\Delta T}$.

Algorithm 3 Flow time inclusion determination and calculation of convergence criteria

1. Define
$$T_{cycle} = \begin{cases} n\Delta T & \text{for first execution of algorithm} \\ T_{cycle,old} & \text{for all other executions of algorithm} \end{cases}$$
2. Sort f_s in ascending order and remove all entries with $f_{sj} = 0$; define f_s^* as the sorted f_s with zero-frequency entries removed. Define
$$f_{low} = \begin{cases} f^*_{sj} & \text{for smallest } j \text{ for which} f^*_{sj} \geq \frac{0.99}{n\Delta T} \\ 1/T_{cycle} & \text{if all } f^*_{sj} < \frac{0.99}{n\Delta T} \end{cases}$$
3. Compare the number of selected modes from the last iteration to the current iteration and compute the weighted sum of fractional change for frequencies WS_f and modes WS_m:
   (a) If $f_{low} = 1/T_{cycle}$, or if the number of selected modes is different ($k_{new} \neq k_{old}$), $WS_f = 1$ and $WS_m = 1$.
   (b) If the number is the same, order all selected frequencies and their associated scaled modes in order of increasing frequency and denote as f_s^{*f} and Φ_s^{*f} respectively, along with associated sorted normalized scaled mode vector B_s^{*f}. Compute the weighted sum of fractional change for frequencies $WS_f = \sum_j \left(\frac{\left|f^{*f}_{sj,new} - fs^{*f}_{j,old}\right|}{f^{*f}_{sj,old}} \right) (B^{*f}_{sj,old})$ and modes $WS_m = \sum_j \left(\frac{\left|\left(\Phi^{*f}_{sj,new}/q^*_{new}\right) - \left(\Phi^{*f}_{sj,old}/q^*_{old}\right)\right|}{\left(\Phi^{*f}_{sj,old}/q^*_{old}\right)} \right) (B^{*f}_{sj,old})$.
4. If $f_{low} = 1/T_{cycle}$, retain all current snapshots for the next iteration. Otherwise, define $T_{cycle,new} = 1/f_{low}$. Evaluate this against the current data as follows:
   (a) If $n\Delta t < T_{cycle,new}$, retain all current snapshots for the next iteration.
   (b) If $n\Delta t > 5T_{cycle,new}$, remove initial snapshots until $5T_{cycle,new}/\Delta t$ snapshots remain for the next iteration.
   (c) If neither of the above apply, denote P as the set of eigenvalues with magnitudes less than or equal to 1: $P = \{\|\lambda_s\| \leqq 1\}$. Denote $Q = \{\|\lambda_s\| > 1\}$. If $n(P) \geqq 0.85 * n(P \cup Q)$ or $n(Q) \geqq 0.85 * n(P \cup Q)$, remove $T_{cycle,new}/\Delta t$ initial snapshots from the current dataset for the next iteration. Otherwise, retain all current snapshots for the next iteration. Note that in this instance, $n(X)$ is used to denote the number of elements in a set.

Algorithm 4 Spatial downsampling of a flow field for DMD

1. Calculate the time-averaged velocity components over the n snapshots being considered, $\bar{u} = n^{-1}\sum_{i=1}^{n} u_i$ and $\bar{v} = n^{-1}\sum_{i=1}^{n} v_i$.
2. Calculate the fluctuating velocity field, $u' = u - \bar{u}$ and $v' = v - \bar{v}$.
3. Calculate the RMS fluctuating values, $u_{rms} = \sqrt{n^{-1}\sum_{i=1}^{n} u_i'}$ and $v_{rms} = \sqrt{n^{-1}\sum_{i=1}^{n} v_i'}$. Compute the magnitude for each point with $\Delta U = \sqrt{(u_{rms})^2 + (v_{rms})^2}$.
4. Define $\Delta\hat{U} = \frac{S(\Delta U)}{max(\Delta U)} - S$ so that $-S \leq \Delta\hat{U} \leq S$. For this implementation of the algorithm, $S = 2$
5. Calculate the cumulative probability from the normal distribution value y for each $\Delta\hat{U}$; y is the probability that the associated point is retained.
6. Assign random, normally-distributed probabilities y_{rand} to each point.
7. Retain points where $y \geq y_{rand}$.

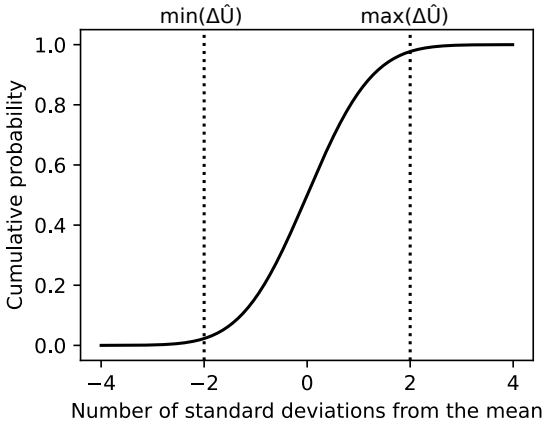


Figure 1 Cumulative probability function for the standardized normal distribution, showing cumulative probabilities for ± 2 standard deviations

capturing enough points to ensure flow information is sufficiently captured while maintaining computational resources savings. This was determined based on comparing the captured points using different number of standard deviations from the mean against the flow regions containing significant spatio-temporal content, which were determined by observing the mode shapes of the converged flow. Figure 1 shows the cumulative probability function for the normal distribution with the mapping of ± 2 standard deviation points to probabilities.

The output of Algorithm 4 is a list of spatial points, downsampled from the original data, that will be used in the subsequent DMD evaluation. It should be noted that the magnitude of a scaled mode scales linearly with the square root of the number of rows in the DMD snapshots, which is the number of spatial points n multiplied with the number of dimensions of the flow. Thus, unlike the analysis of the full data, the number of points in Algorithm 1 fluctuates from step to step, causing the weighted sum of fractional change of the captured modes (WS_m) to be higher than what it would be when comparing data chunks with an equal number of points. To remedy this, a normalization factor n^* can be found with $n^* = \sqrt{nd}$, where n is the number of points remaining in the flow field after spatial downsampling, and d is the number of dimensions of the flow field. The calculation of WS_m is modified as shown in Algorithm 3 step 4(b).

4 Test Cases

The updated algorithm and the spatial downsampling method will be demonstrated on two CFD test cases. These cases were first presented by the authors (George and Defoe, 2023).

The first is a 2D fully turbulent flow over a cylinder, with $Re = 10^7$, to generate a flow with large areas of separation and increase the complexity of the DMD modes. The computational domain is shown in Fig. 2(a). The top and bottom edges of the domain are periodic boundaries to represent a row of cylindrical "blades" with diameter $D = 2$ m and solidity $1/3$. The cylinder wall has a no-slip boundary condition, and uses the velocity-based Spalding wall function (Popovac

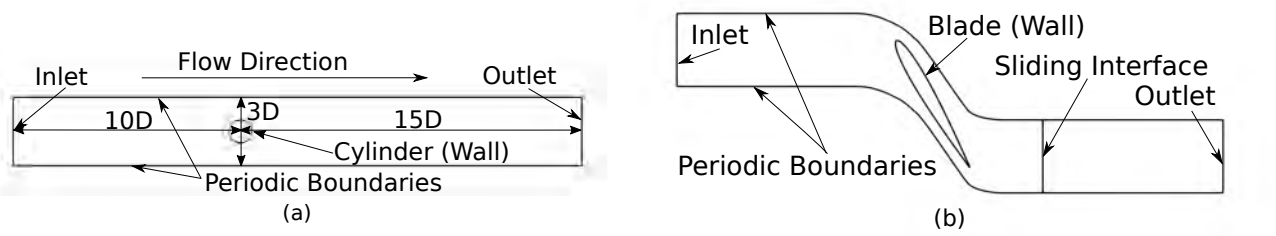


Figure 2 (a) Flow over cylinder case geometry and boundary conditions, and (b) Rotor in duct case geometry and boundary conditions

and Hanjalic, 2007) to calculate the turbulent kinematic viscosity υ_t. The inlet velocity $\vec{V}$ is 1 m/s to the right. This is an incompressible, unsteady Reynolds-averaged Navier-Stokes (URANS) computation, and is carried out in OpenFOAM v2106, using the steady state solver simpleFoam to provide the initial conditions for the unsteady solver pimpleFoam. It has fully second order discretization schemes in space and time, and uses Menter's shear stress transport (SST) model (Menter, 1994). Grid independence was assessed in (George and Defoe, 2023) and the results presented are for a grid with 431360 cells. The CFD time step is $\Delta t = 0.00085\tau$, where τ is one cycle of the shedding frequency from the cylinder. Further details on the case setup and mesh study can be found in (George and Defoe, 2023).

Data is generated up to 85τ, and the snapshot interval used for DMD is $\Delta T/\tau = 0.0425$. Using the original convergence algorithm from (George and Defoe, 2023), it is found that the solution can be terminated at $t = 42.5\tau$, with a set interval of 1.7τ of data (40 snapshots) added at each iteration of the algorithm.

The second test case is a 2D rotor problem, with a single NACA 0012 blade translating (with a moving mesh) followed by a stationary duct downstream, with axial inflow with zero incidence and $Re = 4.19 \times 10^5$. The computational domain is shown in Fig. 2(b). The solidity is 2.0, and the sliding interface is one pitch downstream of the trailing edge of the blade. Similar to the flow over cylinder case, this is an incompressible URANS computation carried out in OpenFOAM v2012, using simpleFoam for initialization and pimpleFoam for generating flow snapshots for use in the convergence algorithm. The inflow is axial with a flow coefficient of 0.577 at zero incidence. Again, further details related to the case setup can be found in (George and Defoe, 2023).

Data is generated up to 13.18 blade passing periods (β), with a snapshot interval of 0.01β. The original convergence algorithm is used on snapshots containing the absolute velocity for both the moving and stationary mesh regions, and finds convergence after one blade passing period, adding 0.5β of data at each iteration of the algorithm. It should be noted that the same result is found when the original algorithm is applied to each mesh region separately, or when applied to the relative velocity of the moving region together with the absolute velocity of the stationary region.

5 Results and Discussion

The updated algorithm with and without spatial downsampling is first demonstrated for the flow over a cylinder test case. The top half of Fig. 3 shows the field of $\Delta\hat{U}$ and the bottom half shows the full set of points (gray) and those that are retained (black) after spatial downsampling. The data used to generate this image is a converged portion of the flow (8.5τ) as determined by the original convergence algorithm in (George and Defoe, 2023). The $\Delta\hat{U}$ field is symmetric top-to-bottom and the points retained are nearly symmetric so this combined view permits a direct comparison of the relationship between $\Delta\hat{U}$ and points retained (in this case approximately 20%). The correlation between areas of high $\Delta\hat{U}$ and more points being retained is clear.

To visualize any impact of spatial downsampling on the mode shapes, we compare the three modes calculated from the same data portion as Fig. 3 which have the largest scaled modal amplitudes, with frequencies of 0 Hz (steady mode), 0.17

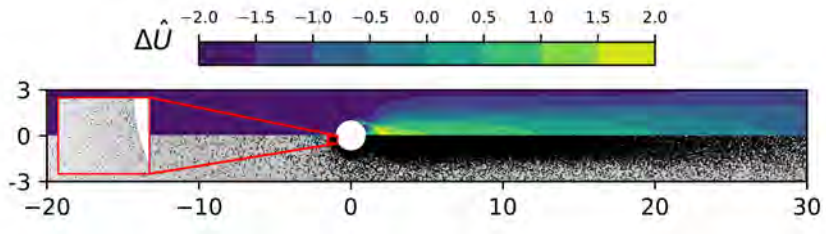


Figure 3 Flow over cylinder case. Top: normalized velocity fluctuation $\Delta\hat{U}$; bottom: grid points (gray: all, black: retained after downsampling). Inset shows detail of original and downsampled spatial points near the cylinder.

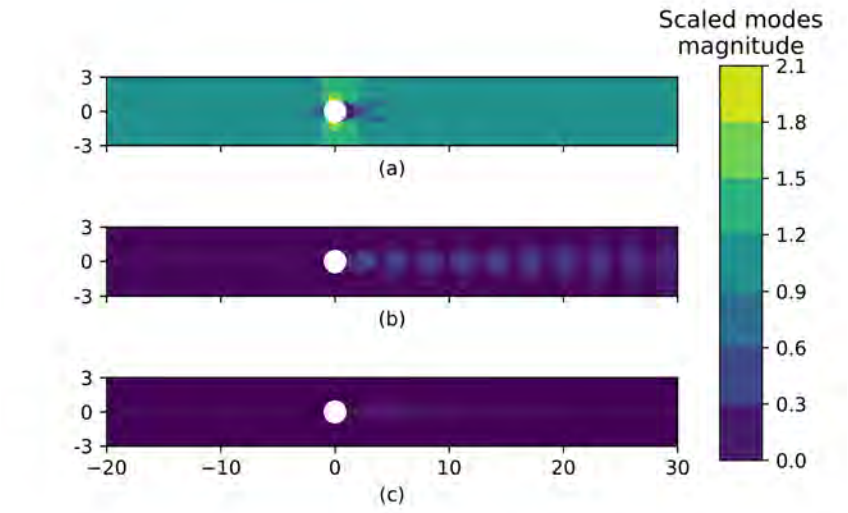


Figure 4 Mode shape comparison for flow over cylinder case between full (top half of each plot) and downsampled (bottom half of each plot) data. (a) 0 Hz mode, (b) 0.17 Hz mode, and (c) 0.50 Hz mode

Hz ($1/\tau$), and 0.5 Hz ($1/0.34\tau$), in Fig. 4 (a), (b), and (c), respectively. The modes are also symmetric top-to-bottom, so the top half of each plot contains the results with no downsampling, and the bottom half contains the downsampled results. These mode shapes indicate zones of interest in the wake and surrounding the cylinder. While 99% of the cumulative modal amplitude in the converged flow over cylinder case captures eight modes in total, the remaining five modes merely indicate similar regions in the wake; the three in Fig. 4 are representative. For all modes, it is clear that the mode shape is not affected by the downsampling. For the three modes shown in Fig. 3, the normalized RMS errors of the mode shapes between the full and downsampled data are 0.8%, 0.52%, and 0.62% respectively.

Figure 5 shows the comparison between the weighted sum of the fractional change for the frequencies and modes for this test case with and without the use of downsampling. The algorithm uses five snapshots as its first interval length. The markers indicate flow times at which the algorithm is executed. The closer the successive markers are along the horizontal axis, the smaller the interval is taken by the algorithm for the next iteration. The algorithm using the full data converges at 37.2τ, whereas when using the downsampled data, convergence occurs at 36τ. For comparison, convergence occurs at 42.5τ when using the original algorithm (George and Defoe, 2023). Table 1 compares the resources used by the original algorithm, the updated algorithm, and the updated algorithm with spatial downsampling. The computational

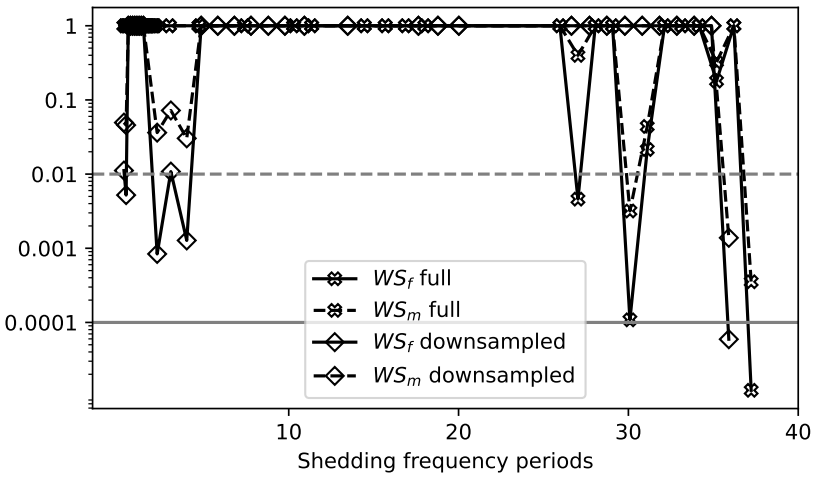


Figure 5 Weighted average of fractional change for frequencies (WS_f, solid lines) and modes (WS_m,dashed lines) for full (X) and downsampled (diamonds) data for flow over cylinder case. Gray horizontal lines are convergence criteria for frequencies (solid) and modes (dashed)

Table 1 Computational resource comparison convergence assessment for flow over cylinder case

	Core-hours	Memory (GB)
Original convergence algorithm	5.5	50
Updated convergence algorithm	1.6	18
Updated convergence algorithm with spatial downsampling	1.4	8.2

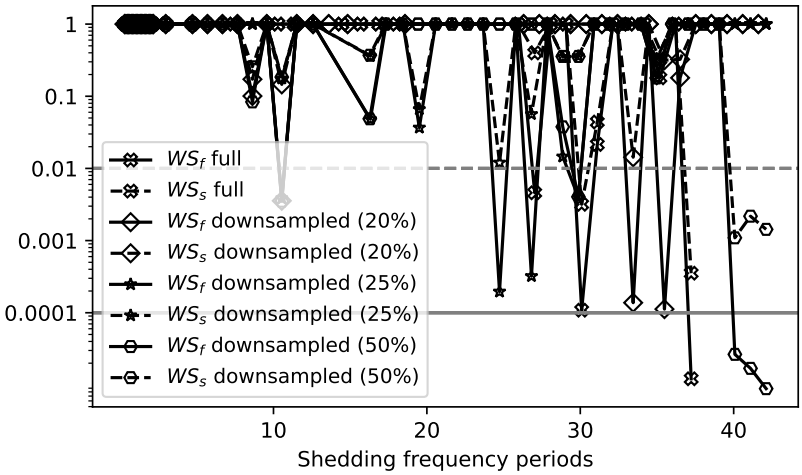


Figure 6 Weighted average of fractional change for frequencies (WS_f, solid lines) and modes (WS_m,dashed lines) for full (X), 20% (diamonds), 25% (stars), and 50% (hexagons) of data points for flow over cylinder case. Gray horizontal lines are convergence criteria for frequencies (solid) and modes (dashed)

cost is cumulative for all required iterative executions of the algorithm. Even without spatial downsampling, the updated algorithm requires approximately 30% of the computational resources the original algorithm required. In addition, the spatial downsampling allows for a further reduction in memory use of more than 50% and a further small decrease in computation time. These results were obtained with the algorithm implemented in Python, so the cost cannot be directly compared to the cost of the CFD itself since the code is far from optimized and not yet implemented to run in parallel. We can get a more specific view on the resource savings by focusing on just the DMD portion, which is the most resource-intensive step and is difficult to fully parallelize. DMD on the final iteration of the full dataset took 292 seconds, while on the downsampled data DMD only took 52 seconds to complete - a savings of 82%.

Analysis of the mode shapes of the full field data at both 36τ and 37.2τ show that there is little difference in flow information. Thus, while the downsampled data converges slightly earlier, it is only by 3.2%, and by 1.2τ, or one algorithm iteration and this does not make a significant difference to the DMD modes obtained: there is no significant loss of accuracy in the final period flow state determined by DMD using the downsampled data for convergence assessment.

As mentioned in the introduction, other methods of data downsampling have been employed in the literature, including downsampling in the spanwise direction. While this method is not applicable for the 2D test cases shown here, practical 3D turbomachinery cases often have low gradients in the spanwise direction (except for at endwalls) compared to in the axial and pitchwise directions. Thus, Algorithm 4 would be likely to automatically decrease spatial resolution in the spanwise direction in such cases. For the flow over cylinder case, a more general method of spatial downsampling was tested. Figure 6 shows the weighted sums for analysis on the full data (100% points), compared with results from using 20%, 25%, and 50% of spatial points, selected randomly at each iteration. Note here that for this case, the downsampling method using the root mean squared variation in velocity retained around 20% of points. When 20% or 25% of points are retained randomly, the solution does not converge even when running the algorithm up to 42.5τ, which is where the original algorithm (George and Defoe, 2023) determined convergence. Thus the behavior of the algorithm was not tested beyond this point. When 50% of spatial points are retained, convergence is reached at 40τ. This shows that randomly selecting points to retain requires a larger number of points retained compared to the calculated method of spatial downsampling to determine a convergence point close to that determined from analyzing the full data. Again, this method requires the user to decide how many points they wish to retain, likely resulting in an arbitrary choice that could become very case dependent.

To justify the choice of ± 2 standard deviations for the mapping range of RMS velocity, in Figure 7 results are shown for ± 1, ± 2, and ± 3 standard deviations for the flow over the cylinder case. Other than the full data, only the downsampled data that was mapped to 2 standard deviations converges before 42.5τ. Smaller and larger ranges are not optimal for

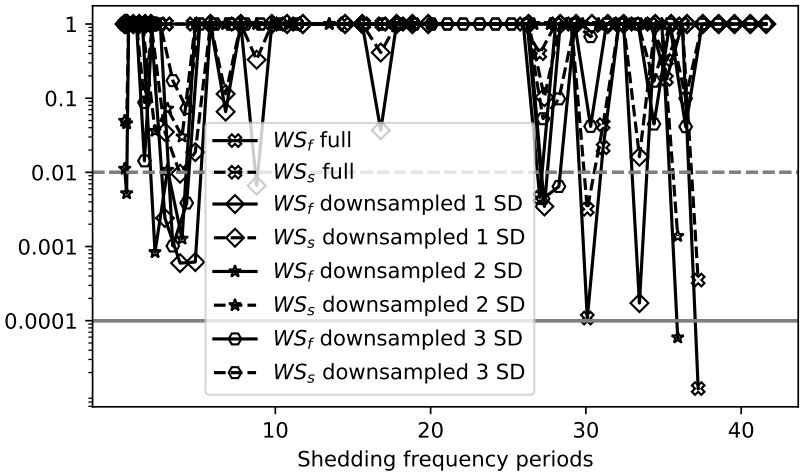


Figure 7 Weighted average of fractional change for frequencies (WS_f, solid lines) and modes (WS_m,dashed lines) for full (X), downsampled using mapping to 1 standard deviation (diamonds), downsampled using mapping to 2 standard deviation (stars), and downsampled using mapping to 3 standard deviation (hexagons) data for flow over cylinder case. Gray horizontal lines are convergence criteria for frequencies (solid) and modes (dashed)

downsampling and reducing total flow time required.

Now we apply the improved algorithm and spatial downsampling function to the second case study - the moving rotor problem. The bottom part of Fig. 8 shows that when the spatial reduction function is applied on a converged portion of the flow using the stationary frame velocity, the majority of the points retained are in the downstream zone, and aside from a cluster around the trailing edge of the blade, very few points remain in the upstream moving zone. This is to be expected, as the moving zone has steady, attached flow while wakes move through the downstream zone. Referring to the top part of Fig. 8, the $\Delta\hat{U}$ values are highest in the downstream zone as well, causing more points to be retained. The $\Delta\hat{U}$ distribution matches the expected flow field behavior in the downstream zone: there is no variation in the vertical direction since wakes move along that axis and the flow field is spatially and temporally periodic, but the magnitude of $\Delta\hat{U}$ decreases moving downstream as the wakes mix out. The discontinuity at the sliding interface is due to the fact that in the upstream zone, the grid is moving with the blade so that the velocity is nearly time-invariant, while in the stationary downstream zone, the wake location changes with time, yielding non-zero $\Delta\hat{U}$.

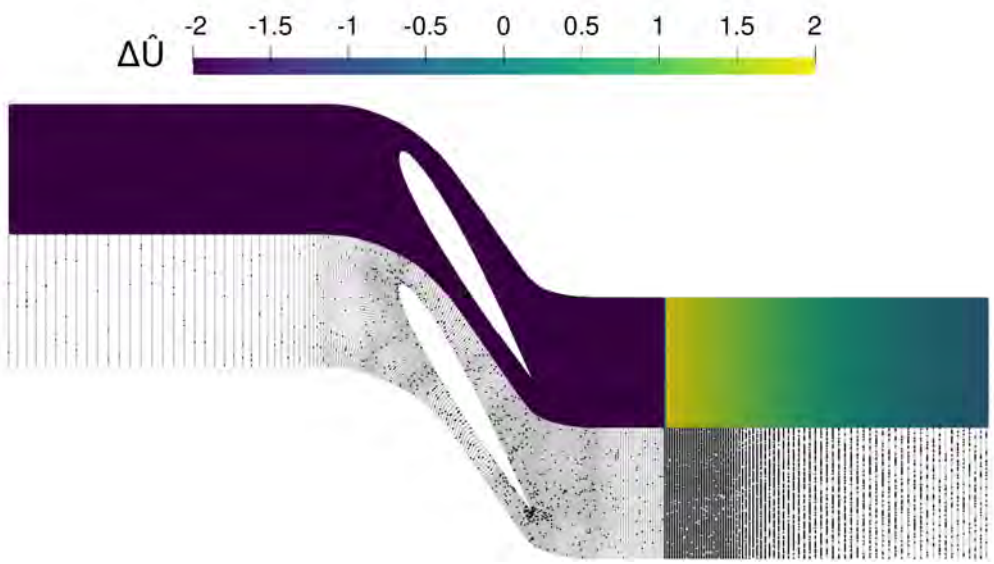


Figure 8 Rotor in duct case. Top: mapped fluctuating velocity magnitude $\Delta\hat{U}$; bottom: grid points (gray: all, black: retained after downsampling)

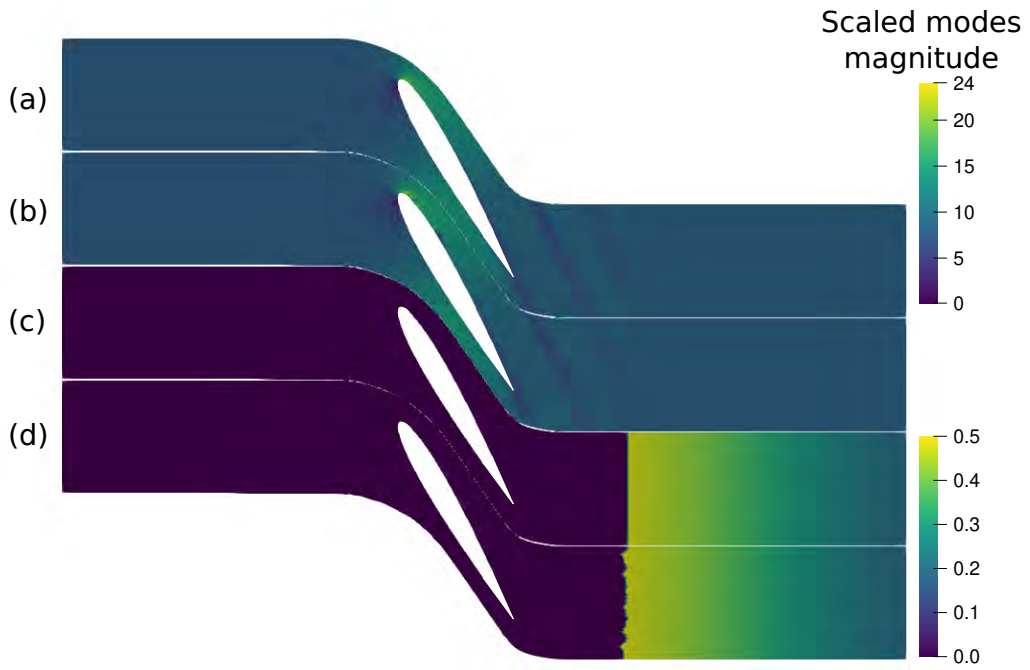


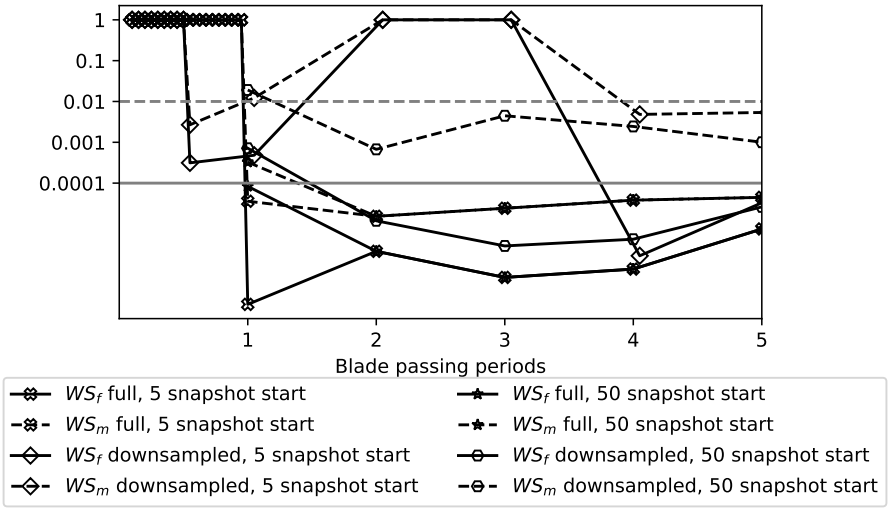
Figure 9 Mode shapes for rotor in duct case (a) 0 Hz mode with full data, (b) 0 Hz mode with downsampling, (c) 26 Hz mode with full data, and (d) 26 Hz mode with downsampling

Figures 9 (a) and (c) show the (only) 2 captured modes for the full data: 0Hz and 26Hz (blade passing frequency). Figures 9 (b) and (d) show the same 2 modes for the downsampled data. Again, the mode shapes are not affected by spatial downsampling. The normalized RMS error of the scaled modes magnitude fields between the full and downsampled data are 3.2% and 5.7% for the 0Hz and 26Hz mode, respectively.

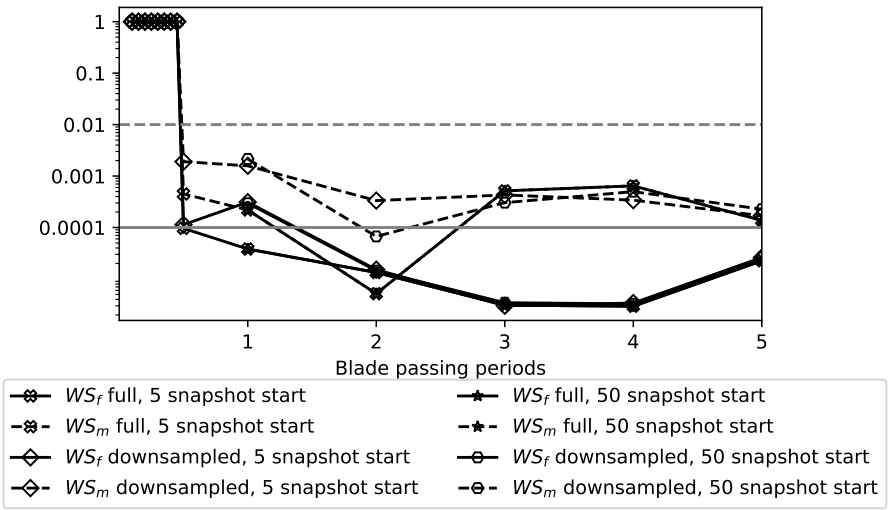
When spatial downsampling is used, an additional mode is captured at twice the blade passing frequency (52 Hz). The reason for DMD to predict this extra mode with spatial downsampling is likely due to the reduction in mesh resolution in the region where the wakes are located, shown in Fig. 8. As the wakes move unsteadily across the points, the uneven distribution causes the number of points over the wake to vary, giving the appearance of a faster-moving component superimposed on the main wakes moving at the blade speed. This frequency does also appear in the complete DMD outputs for the full data, but at such a low modal amplitude that it is not captured in the analysis of 99% of the total modal amplitude. This shows that the spatial downsampling amplifies the issue of the unsteady behavior aliasing due to discretization, although this could be avoided by using a filter (Tu 2013). Nevertheless, as will be discussed next, this has little effect on the convergence of the algorithm.

Using the same five snapshots to start and analyzing both mesh regions together, Fig. 10(a) shows the downsampled data converges after four blade passing periods, while the full data converges after just one blade passing period. This discrepancy is caused by a change in number of modes captured at three and four blade passing periods, from three, to four, then back to three modes. The extra mode is another (decaying) zero-frequency mode. This test case demonstrates the importance of the selection of the initial number of snapshots. In most turbomachinery applications, the user can usually make a reasonable guess to what a fundamental period ought to be – typically either a rotor revolution period or a blade passing period. In this case, it is the blade passing period since the domain spans a single blade pitch. Initially using 50 snapshots (half a blade passing period) results in the downsampled data converging in half the flow time (after two blade passing periods), as shown in Fig. 10(a). The full data continues to converge after 1 blade passing period.

Analysis of data from just the downstream duct is also conducted. Figure 10(b) shows a test starting with five snapshots, where the full data rapidly converges after half a blade passing period. The downsampled data continues to converge after two blade passing periods. Finally, Fig. 10(b) also shows the results of starting the algorithm with 50 snapshots on the downstream duct only, which yields similar results to Fig. 10(a). When analyzing both mesh regions together, around 15% of points are retained, whereas when considering just the downstream duct, roughly 65% of points are retained by the spatial downsampling. Thus, while the algorithm requires double the flow time to reach convergence using spatial downsampling in this case, it does so being able to operate on only 15% of the spatial points, leading to a significant reduction in the computational expense of carrying out the DMD-based convergence algorithm: the DMD computation on the final iteration for the full data required 9.00 seconds to complete, while only requiring 1.03 seconds to complete for the downsampled data - a reduction of 89%. Recall that these computations were conducted on a single core.



(a)



(b)

Figure 10 Weighted average of fractional change for for frequencies (WS_f, solid lines) and modes (WS_m,dashed lines) for full (X) and downsampled (diamonds) data for rotor in duct case. Gray horizontal lines are convergence criteria for frequencies (solid) and modes (dashed). Results are shown for starting the algorithm with 5 and 50 initial snapshots, using data from the (a) full domain, and (b) downstream zone data only

Furthermore, since the downstream zone is constructed from a structured, rectangular grid, the idea of spatial downsampling in one dimension only (akin to the spanwise downsampling mentioned above) is worth exploring. The downstream zone grid is 76 points in the axial direction and 96 points in the pitchwise direction. Since the wakes move pitchwise in this zone, convergence of the flow is incredibly insensitive to spatial reduction in the axial direction: it was found that, while maintaining full spatial resolution in the pitchwise direction, the spatial resolution can be reduced to merely 2 points in the axial direction before convergence reached later than one blade passing period. When reducing spatial resolution in the pitchwise direction only, a reduction factor of 10 does not affect convergence, beyond which DMD struggles to identify the correct spatio-temporal content. Thus, if the user has a good understanding of the behavior of the flow they are analyzing, they could use this in conjunction with the spatial downsampling algorithm to more aggressively downsample the input data to DMD, further reducing their computational cost. However, this method only works well on structured grids, where downsampling along one dimension only is a simple implementation.

This rotor in duct case has almost no transient, since the flow field is initialized with the flow field solved in the rotor frame of reference, where the flow is steady. Moreover, the small gird size allowed for a low computational cost to compute DMD for the full data regardless. For this particular case, the savings are not justified: the computation time required for CFD for an extra blade passing period was 0.77 core-hours. Cases such as these are thus not a good candidate for spatial downsampling. However, for most practical cases in turbomachinery, the initialized solution will not be exact, resulting in somewhat of a startup transient, similar to the behavior of the flow over cylinder case. This is particularly true for cases with complex, non-uniform inflows. For larger, more complex turbomachinery cases, the computational time required for settling the initial transient and to complete DMD will necessitate downsampling, even if it may require some more flow time. The algorithm still enables a rigorous determination of when sufficient flow time has been computed.

6 Conclusions

This paper has expanded upon a recently-introduced convergence assessment algorithm for unsteady CFD, including the addition of a spatial downsampling algorithm to reduce the computational and memory expense of the algorithm. The advantages of the approach is that no a priori insight into the nature of the flow field is required, including zones of interest or duration of a fundamental cycle. The spatial downsampling makes the approach useful for practical CFD applications where meshes are typically tens of millions to over 100 million cells.

The updated approach is assessed on 2 different CFD cases - of a fully turbulent flow over a cylinder, and a 2D flow over a turbomachinery rotor blade. These two cases span extremes: the cylinder case involved solving the time evolution of the flow field, requiring many vortex shedding periods before reaching steady-state conditions; the rotor case, on the other hand, involves fully attached flow and is initialized from a steady solution in the rotor frame, so there is zero start-up transient. Both cases demonstrate that the algorithm is able to determine when sufficient flow time has elapsed, and the rotor case demonstrates the impact of the choice of initial sample duration on the total time required, especially when employing spatial downsampling.

The ability to downsample spatial points allows this tool to be useful in practical turbomachinery applications, which are often computationally expensive, and require long run times.

7 Nomenclature

Roman letters

a DMD amplitudes

A Scaled modes matrix

B Vector of normalized scaled modes

d Number of dimensions in flow field

D Diameter

f Frequency

k Number of selected modes

m Number of temporal points minus 1

n Number of spatial points

q Scaling factor to account for difference in spatial points

S Number of standard deviations to map the RMS of velocity fluctuations

Δt Time step of CFD

ΔT Time step of DMD

T_{cycle} Period of fundamental cycle

u Velocity in x direction

v Velocity in y direction

$\vec{V}$ Velocity vector

WS Weighted sum

Greek letters

Θ DMD modeshapes
λ DMD eigenvalues
ν Kinematic viscosity
τ Length of 1 shedding period
β Length of 1 blade passing period
Φ Scaled modes
Dimensionless groups
Re Reynolds number
Subscripts
f Frequency
m Scaled mode
s Selection representing 99% cumulative scaled modal amplitude
Acronyms
CFD Computational Fluid Dynamics
DMD Dynamic Mode Decomposition
DNS Direct Numerical Simulation
RMS Root mean square
SST Shear stress transport
URANS Unsteady Reynolds-averaged Navier-Stokes

8 Acknowledgements

This research was enabled in part by support provided by Compute Ontario (computeontario.ca) and the Digital Research Alliance of Canada (alliancecan.ca).

References

Clainche, Soledad Le et al. (Mar. 2015). "Flow around a hemisphere-cylinder at high angle of attack and low Reynolds number. Part II: POD and DMD applied to reduced domains." In: *Aerospace Science and Technology* 44, pp. 88–100.

Courrieu, Pierre (Aug. 2005). "Fast Computation of Moore-Penrose Inverse Matrices." In: *Neural Information Processing - Letters and Reviews* 8 (2), pp. 25–29.

George, Amelia and Jeff Defoe (2023). "Convergence of Dynamic Mode Decomposition as an Assessment Criteria for Completion of Internal, Unsteady Flow Simulations." In: DOI: GT2023-101480.

Gunn, E. J. and C. A. Hall (June 2014). "Aerodynamics of Boundary Layer Ingesting Fans." In: American Society of Mechanical Engineers. ISBN: 978-0-7918-4557-8. DOI: 10.1115/GT2014-26142.

Krake, T. et al. (2021). "Visualization and selection of Dynamic Mode Decomposition components for unsteady flow." In: *Visual Informatics* 5 (3). ISSN: 2468502X. DOI: 10.1016/j.visinf.2021.06.003.

Li, Cruz Y. et al. (Mar. 2022). "A parametric and feasibility study for data sampling of the dynamic mode decomposition: range, resolution, and universal convergence states." In: *Nonlinear Dynamics* 107 (4), pp. 3683–3707.

Menter, F. R. (Aug. 1994). "Two-equation eddy-viscosity turbulence models for engineering applications." In: *AIAA Journal* 32 (8), pp. 1598–1605. ISSN: 0001-1452. DOI: 10.2514/3.12149.

Popovac, M. and K. Hanjalic (Mar. 2007). "Compound Wall Treatment for RANS Computation of Complex Turbulent Flows and Heat Transfer." In: *Flow, Turbulence and Combustion* 78 (2), p. 177. ISSN: 1386-6184. DOI: 10.1007/s10494-006-9067-x.

Tu, Jonathan H. (2013). "Dynamic Mode Decomposition: Theory and Applications." PhD thesis. Princeton University.

A Appendix A

Algorithm 5 Exact Dynamic Mode Decomposition (Krake et al., 2021)

1. Define $X = [x_0, x_1, \ldots, x_{m-1}]$, $Y = [x_1, x_2, \ldots, x_m]$.
2. Calculate reduced SVD where $X = U\Sigma V^*$.
3. Calculate r =rank(X).
4. Calculate $S = U^* Y V \Sigma^{-1}$.
5. Calculate the eigenvalues and eigenvectors of S. Keep only the non-zero eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_r)$ and their associated eigenvectors $(\varphi_1, \varphi_2, \ldots, \varphi_r)$
6. Calculate $\Theta_j = \frac{1}{\lambda_j} Y V \Sigma^{-1} v_j$
7. Calculate the amplitudes $a = \Lambda^{-1} \Theta^+ x_1$, where Λ =diag$(\lambda_1, \lambda_2, \ldots, \lambda_r)$, and $\Theta^+$is the Moore-Penrose pseudo-inverse of Θ.