11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
E. Oñate, J. Oliver and A. Huerta (Eds)

A NOVEL SOLVER ACCELERATION TECHNIQUE BASED ON DYNAMIC MODE DECOMPOSITION

Niklas Andersson[1] **and Lars-Erik Eriksson**[2]

[1] Chalmers University of Technology, SE-41296 Gothenburg Sweden,
niklas.andersson@chalmers.se
[2] Chalmers University of Technology, SE-41296 Gothenburg Sweden,
lars-erik.eriksson@chalmers.se

Key words: Computational Fluid Dynamics, Compressible Flows, Computing Methods, Dynamic Mode Decomposition.

Abstract. The speed-up of finite-volume solvers for compressible flows is a difficult task. There are several ways to achieve solver speed-up, more or less difficult to implement and more or less suitable for implementation in a parallel, unstructured type of solver. Examples of such techniques are the multi-grid method and Implicit Residual Smoothening (IRSM). In this article, a solver acceleration technique based on Dynamic Mode Decomposition (DMD) is proposed. The technique does not depend on data format or mesh structure and is thus as straightforward to implement in an unstructured parallel code as in a structured sequential one. The main idea behind the proposed method is that it is possible to use the information available in flow field modes extracted using the DMD technique to find a correction that will bring the solution closer to a steady-state condition, *i.e.* the method is only applicable to steady-state problems. In the presented work the proposed DMD-based acceleration technique has been implemented in a massively parallel block-structured finite-volume Navier-Stokes solver for compressible flows. The method has been tested on a turbine cascade case with promising results. To the knowledge of the authors, the proposed method is not previously published in the open literature.

1 INTRODUCTION

The Dynamic Mode Decomposition (DMD) was first introduced by Schmid [1] as a method for extracting coherent dynamic flow structures from a previously generated set of data samples. The fact that the method does not require any information about the origin of the data means that it can be applied to data obtained from for example transient numerical simulations or experiments. The DMD method is a Krylov subspace method related to the iterative Arnoldi algorithm for extraction approximate eigenvalues

and eigenvectors of large matrices. The technique proposed by Arnoldi [2], also described in detail by *e.g.* Ruhe [3] and Eriksson *et al.* [4], is based on the projection of a high-dimension system matrix onto a subspace of significantly lower dimension. The nature of this subspace is such that its eigenvalues represents the least damped modes of the extensively larger system matrix. One of the benefits of the DMD method over a method based directly on the Arnoldi algorithm is that it can be applied to an existing set of samples as a post processing procedure whereas an eigenmode extraction method based on the Arnoldi algorithm would require that the data set is generated as part of the eigenmode extraction procedure and thus it is impossible to use this kind of method on an already existing data set.

Eigenmode extraction methods such as the DMD method and methods based on the Arnoldi algorithm have recently been used for the identification of dynamic flow structures by a number of research groups. Rowley *et al.* [5] used an Arnoldi-based algorithm for the identification of modes for a jet in cross flow. Seena *et al.* [6] used DMD to study self-sustained oscillations in cavity flows. Muld *et al.* [7] analyzed flow structures around a high-speed train obtained using Detached Eddy Simulation (DES) and DMD. Jourdain *et al.* [8] used DMD to analyze the acoustic modes in a combustion chamber configuration. Lárusson *et al.* [9] used an Arnoldi-based method to study eigenmodes in a separated nozzle flow.

The Generalized Minimal Residual (GMRES) technique proposed by Saad *et al.* [10] is a Krylov subspace method for solving linear systems. GMRES is related to the above-mentioned Arnoldi algorithm and has been used to speed up the convergence of flow solvers in steady-state application, see *e.g.* Nigro *et al.* [11] and Stridh *et al.* [12]. Since the GMRES method is a Arnoldi-based technique that may be applied for solver speed-up purposes, it seems natural that it is possible to utilize the mode information that can be extracted using the DMD method to speed up the convergence of flow solvers as well. The presented DMD-based acceleration technique uses a set of stored solver flow fields to calculate a low-dimension projection of the system matrix describing the flow field development. This approximate system matrix is than used to generate a flow field correction that brings the solution closer to convergence. The flow corrections are done in a step-wise manner alongside an otherwise normal solver execution. The most efficient size and sample frequency of the generated data sets are very problem dependent and have to be found by trial and error. However, with suitable settings, the proposed DMD-based acceleration technique is able to produce significant speed-up.

2 METHOD DESCRIPTION

This section describes the underlying theory of the proposed method and an example of code implementation.

2.1 DMD-based Correction

The proposed correction technique is based on the assumption that the solver time stepping scheme may be described using a linear expression

$$x^{(n+1)} = Ax^{(n)} + b \tag{1}$$

where x represents the degrees of freedom (in our case the flow field variables in all grid cells), n denotes time and A is the system matrix describing the temporal development of x. Our aim is to use a Krylov subspace method to generate a projected system matrix of significantly lower dimension that still contains the most essential information available in A. To achieve this, a Krylov subspace is built up by saving a number of solution states (flow field snap shots) with a specified sampling frequency which results in a set of vectors $\{x^{(1)}, x^{(2)}, \ldots, x^{(n+2)}\}$. Differences of consecutive sampled solver states are used to define matrices V_n and V_{n+1}

$$\begin{aligned} V_n &= \left[x^{(2)} - x^{(1)}, \ldots, x^{(n+1)} - x^{(n)}\right] \\ V_{n+1} &= \left[x^{(3)} - x^{(2)}, \ldots, x^{(n+2)} - x^{(n+1)}\right] \end{aligned} \tag{2}$$

Note that $(n+2)$ samples are needed to generate the matrices V_n and V_{n+1}. The usage of solver state differences has been shown to be beneficial for the performance of the DMD algorithm [13].

The matrix V_n can be rewritten as

$$V_n = E_n U_n \tag{3}$$

where E_n is a $(m \times n)$ matrix built up from a set of orthonormal vectors and U_n is an upper triangular $(n \times n)$ matrix. The sizes m and n corresponds to the number of degrees of freedom and the number of samples, respectively.

According to the linear relation given by equation (1), V_{n+1} can be expressed in terms of V_n as

$$V_{n+1} = AV_n \tag{4}$$

Now, combining equation (4) and (3) gives

$$V_{n+1} = AE_n U_n \tag{5}$$

Multiplying both sides of equation (5) by the transpose of the orthonormal base E_n from the left gives

$$E_n^T V_{n+1} = \underbrace{E_n^T A E_n}_{\widetilde{A}_n} Un \tag{6}$$

where $\widetilde{A}_n = E_n^T A E_n$ is the projection of the system matrix A on the orthonormal base E_n. Multiplying each side of equation (6) by the inverse of U_n from the right, the projected system matrix, $\widetilde{A}_n$, can now be obtained as

$$\widetilde{A}_n = E_n^T V_{n+1} U_n^{-1} \tag{7}$$

Now, let's go back to the linear relation (equation (1)). The solution to this relation is found when $x^{(n)}$ equals $x^{(n+1)}$. Inserting $x^{(n+1)} = x^{(n)} = x$ in equation (1) gives

$$(I - A)x = b \tag{8}$$

Introducing a correction Δx, obtained by subtracting sample $x^{(n+1)}$ from x, equation (8) can be rewritten as

$$(I - A)x = (I - A)\Delta x + (I - A)x^{(n+1)} = b \tag{9}$$

Rearranging gives

$$(I - A)\Delta x = \underbrace{(A - I)x^{(n+1)} + b}_{x^{(n+2)} - x^{(n+1)}} \tag{10}$$

where, since $x^{(n+2)} = Ax^{(n+1)} + b$, the right hand side equals $x^{(n+2)} - x^{(n+1)}$, which corresponds to the last vector in V_{n+1}. We now introducing Δy as the projection of the correction Δx on the orthonormal base E_n

$$\Delta x = E_n \Delta y \tag{11}$$

replacing Δx in equation (10) with the expression in equation (11) gives

$$(I - A)E_n \Delta y = x^{(n+2)} - x^{(n+1)} \tag{12}$$

Multiplying both sides of equation (12) with the transpose of E_n from the left gives

$$E_n^T (I - A)E_n \Delta y = E_n^T \left(x^{(n+2)} - x^{(n+1)}\right) \tag{13}$$

From before we have that $\widetilde{A}_n = E_n^T A E_n$ is the projection of A on E_n. Inserting this relation in equation (13) gives

$$(I_n - \widetilde{A}_n)\Delta y = E_n^T \left(x^{(n+2)} - x^{(n+1)}\right) \tag{14}$$

now, by multiplying with $(I_n - \widetilde{A}_n)^{-1}$ from the left on each side we get an expression for the correction Δy in terms of the projected system matrix $\widetilde{A}_n$, the orthonormal base E_n and the last vector in V_{n+1}.

$$\Delta y = (I_n - \widetilde{A}_n)^{-1} E_n^T \left(x^{(n+2)} - x^{(n+1)} \right) \tag{15}$$

If we go back to the definition of Δx we see that what we have achieved now is a relation between the sampled solver state x^{n+1} and the assumed converged steady-state solution x thus we can get a an estimate of the converged solution x as

$$x^u = x^{(n+1)} + \Delta x = x^{(n+1)} + E_n \Delta y \tag{16}$$

Inserting (15) in (16) gives

$$x^u = x^{(n+1)} + E_n (I_n - \widetilde{A}_n)^{-1} E_n^T \left(x^{(n+2)} - x^{(n+1)} \right) \tag{17}$$

2.2 A Simple Example

The correction method described above is applied to a simple linear relation described by

$$x^{n+1} = a x^n + b$$

where the constants a and b equals 0.90 and 1.00, respectively. The exact solution for this simple example is $x = b/(1-a) = 10$. x is updated three times starting with $x^{(0)} = 0.00$ producing the following sequence of values

$$\begin{aligned} x^{(0)} &= 0.00 \\ x^{(1)} &= 1.00 \\ x^{(2)} &= 1.90 \\ x^{(3)} &= 2.71 \end{aligned}$$

According to the definition of V_n and V_{n+1} these matrices (or scalars in this case) can now be obtained from the samples $x^{(0)}$ to $x^{(3)}$ as

$$\begin{aligned} V_n &= x^{(2)} - x^{(1)} = 0.90 \\ V_{n+1} &= x^{(3)} - x^{(2)} = 0.81 \end{aligned}$$

Since V_n is a scalar, E_n equals 1.0 and $U_n = V_n$, see equation 3. Inserting the values calculated above in equation (7) gives us the projected system matrix for this problem

$$\widetilde{A}_n = E_n^T V_{n+1} U_n^{-1} = 0.90$$

The projected correction is obtained using equation (15)

$$\Delta y = (I_n - \widetilde{A}_n)^{-1} E_n^T (x^{(3)} - x^{(2)}) = 8.10$$

The correction Δx is obtained from equation (11) by multiplying the projected correction with the orthonormal base E_n

$$\Delta x = E_n \Delta y = 8.10$$

Finally, a corrected solution is obtained by using equation (16)

$$x^u = x^{(2)} + \Delta x = 10.0$$

2.3 Solver Implementation

The above-described method has been implemented in a parallel Navier-Stokes solver for compressible flows. The orthogonalization of V_n is done using the Gram-Schmidt algorithm. The reason for not using a more robust technique like Singular Value Decomposition (SVD) is that using the Gram-Schmidt algorithm, one only have to handle two vectors in V_n at the same time whereas if using SVD one would have to work with the entire matrix V_n. Moreover, when calculating the scalar products in parallel, only the local part of the matrix has to be accessed and the only communication needed between processes is the local contribution to each scalar product, i.e. a scalar value. Thus, the Gram-Schmidt algorithm is more suitable for parallel computations than other available alternatives. However, there are packages available that handles parallel SVD efficiently such as for example SLEPc [14], which is an extension of the PETSc [15] library that can be used for solving large matrix eigenvalue problems in parallel.

The DMD correction algorithm is set up by specifying the number of samples to be used in each DMD-cycle, N_S, and the number of solver iterations between each of these samples, N_I. The optimal choice of values for these two parameters for best performance varies from application to application and has to be found by trial and error. Once a DMD-cycle is completed, the sample database is reset and the sampling process is restarted. In order to remove high frequency modes and thereby improve the efficiency of the DMD method, a user-defined number of solver time steps, N_P, are completed in each DMD correction cycle before the sample generation is started. This will effectively remove start-up transients and transients related to a previous correction, which otherwise might disturb the acceleration procedure.

The implementation of the DMD correction method is done in an on-the-fly manner meaning that it uses the same solver as when DMD correction is deactivated but in the DMD case, samples are stored on disk with a user defined time step increment, N_I, and a DMD correction is calculated when a user defined number of samples, N_S, have been extracted. This makes the overhead of this method significantly lower than related methods such as for example GMRES.

3 TEST CASES

3.1 One-dimensional diffusion

The first test case is a simple one-dimensional diffusion case. The proposed acceleration technique should perform very well for these types of problems since the system matrix for such a problem has a spectrum that is suitable for Krylov subspace methods such as DMD.

one-dimensional diffusion of a quantity, q is described by the following relation

$$\frac{\partial q}{\partial t} = \frac{\partial^2 q}{\partial x^2} \tag{18}$$

Discretizing equation (18) using a finite difference approach gives

$$\frac{q_i^{n+1} - q_i^n}{\Delta t} = \frac{q_{i-1}^n - 2q_i^n + q_{i+1}^n}{(\Delta x)^2} \tag{19}$$

Inserting $\lambda = \frac{\Delta t}{(\Delta x)^2}$ in equation (19) gives

$$q_i^{n+1} = \lambda q_{i-1}^n + (1 - 2\lambda) q_i^n + \lambda q_{i+1}^n \tag{20}$$

It should be noted that solving these types of problems using a finite-difference approach will result in very slow convergence and it is thus not the natural way forward but it will demonstrate the capability of the DMD-based solver acceleration.

3.1.1 Solver Settings and Boundary Conditions

In this specific case, the one-dimensional diffusion problem is discretized using 100 nodes. The parameter λ in equation (20) is set to 0.25. Direchlet boundary conditions for q are used both at the left and the right end. At the left end $q = 1.0$ and at the right end $q = 2.0$. The test convergence criteria used is that the residual should be lower than 10^{-7}. Also, the total number of iterations was limited to 30000.

3.1.2 Finding Optimum DMD Settings

In order to find settings for the DMD algorithm that improves the solver performance as much as possible a number of tests were done. In each configuration a combination of values of the parameters N_S and N_I were tested. The result is shown in Table 1. It can be seen, that for this specific case the best result is obtained for $N_S = 15$ and $N_I = 75$. It should be mentioned that when the DMD acceleration is deactivated convergence is not achieved within the limited number of time steps specified for the test. Also, a more efficient combination of number of samples (N_S) and number of iterations (N_I) can most likely be found by refining the test setup. However, the data presented in Table 1, gives an indication of the trend and, what is more important, it shows that the DMD-based

correction method has a great potential in accelerating the convergence of a diffusion type of problem significantly.

Table 1: One-dimensional diffusion test matrix: number of iterations to reach convergence for different combinations of number of samples in each DMD-cycle, N_S, and number of iterations between each sample, N_I

		N_I				
		50	75	100	125	150
N_S	5	7659	7599	7304	7897	8495
	10	3501	4030	4186		
	15	2381	**1280**			
	20	2006				

3.1.3 DMD Performance

Figure 1 shows the samples and corrected result for the first DMD cycle using the optimal configuration found in the test described above, see Table 1. In this case one DMD cycle is enough to get a solution that is very close to convergence.

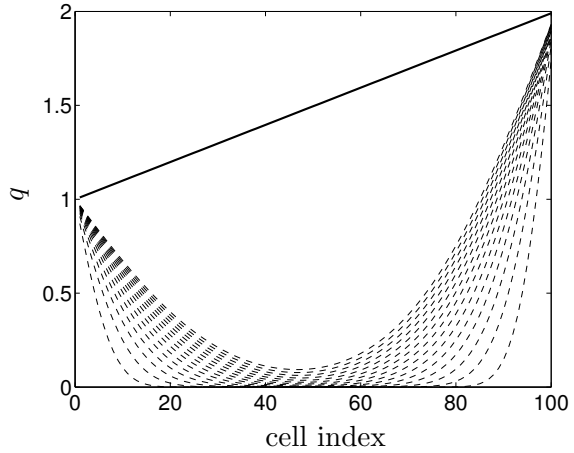


Figure 1: The samples of variable q (dashed lines) and the corrected solution after the first DMD cycle (solid line)

3.2 One-dimensional Convection-Diffusion

In section 3.1 it was shown that the proposed DMD-based correction technique works for a simple diffusion problem. In order to show that the method is suitable for a Navier-Stokes type of problem, it is applied to a simple convection-diffusion problem. Introducing convection changes the spectrum of the system matrix and will therefore have an impact on the performance of the DMD acceleration.

A one-dimensional convection-diffusion problem can be described by the following relation

$$\frac{\partial Q}{\partial t} + C\frac{\partial Q}{\partial x} = \varepsilon \frac{\partial^2 Q}{\partial x^2} \tag{21}$$

where

$$C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ and } Q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

Discretizing equation (21) using a finite difference approach gives

$$\Delta x \frac{d}{dt}\overline{Q}_i + C\left[\widehat{Q}_{i+\frac{1}{2}} - \widehat{Q}_{i-\frac{1}{2}}\right] = \varepsilon\left(\frac{\partial Q}{\partial x}\right)_{i+\frac{1}{2}} - \varepsilon\left(\frac{\partial Q}{\partial x}\right)_{i-\frac{1}{2}} \tag{22}$$

In equation (22), $\overline{Q}_i$ denotes cell average of Q in cell i, $\widehat{Q}_{i+\frac{1}{2}}$ is an estimate of Q at the right cell face of cell i obtained from cell averages of Q as

$$\widehat{Q}_{i+\frac{1}{2}} \approx -\frac{1}{12}\overline{Q}_{i-1} + \frac{7}{12}\overline{Q}_i + \frac{7}{12}\overline{Q}_{i+1} - \frac{1}{12}\overline{Q}_{i+2} \tag{23}$$

and $\left(\frac{\partial Q}{\partial x}\right)_{i+\frac{1}{2}}$ is the spatial derivative of Q at the right cell face of cell i estimated as

$$\left(\frac{\partial Q}{\partial x}\right)_{i+\frac{1}{2}} \approx \frac{\overline{Q}_{i+1} - \overline{Q}_i}{\Delta x} \tag{24}$$

$\widehat{Q}_{i-\frac{1}{2}}$ and $\left(\frac{\partial Q}{\partial x}\right)_{i-\frac{1}{2}}$ are the corresponding properties at the left cell face of cell i. Equation (22) is solved using a three-stage Runge-Kutta time-marching solver.

3.2.1 Solver Settings and Boundary Conditions

The one-dimensional convection-diffusion problem is discretized using 100 cells, each with a cell width $\Delta x = 0.001\ m$. The solver time step is set using $CFL = \frac{\Delta t}{\Delta x} = 0.7$. The diffusion coefficient, ε, is set to 10^{-4}. At the left boundary, $q_1 = 1.0$ and q_2 is extrapolated from the interior and at the right boundary, q_1 is extrapolated from the interior and $q_2 = 1.0$. As in the diffusion case, the test convergence criteria used was that the solver residual should be lower than 10^{-7} and if the solution was still not converged after 30000 iterations the execution was halted.

3.2.2 Finding Optimum DMD Settings

As for the one-dimensional diffusion case, a number of combinations of settings for N_S and N_I were tested in order to find the combination that produces the most speed-up. For the convection-diffusion case the best results was found for $N_S = 50$ and $N_I = 10$. The results from the tests are presented in Table 2. Comparing results from the one-dimensional diffusion case and the convection-diffusion case, Table 1 and Table 2, the difference in nature of the two problems is obvious. For the diffusion case, the most speed-up is obtained when using 15 samples and 75 iterations between each sample whereas in the convection-diffusion case, the most efficient configuration has 50 samples in each DMD-cycle and only 10 iterations between each sample, i.e. in the diffusion case, the number of samples is 30% of the number of samples used in the convection-diffusion case but the number of iterations between each sample is 7.5 times higher.

Table 2: One-dimensional convection-diffusion test matrix: number of iterations to reach convergence for different combinations of number of samples in each DMD-cycle, N_S, and number of iterations between each sample, N_I

		N_I						
		10	25	50	75	100	125	150
N_S	10		30000	5852	5036	5519	9441	5458
	15		2299	3359	4557	4501	7660	4792
	20		2016	3355	4501	4001		
	25	30000	1909	2795	2667			
	30	2871	1870	2044	2301			
	35	1071	1760	1773				
	40	809	1160	2001				
	45	628	1126	2251				
	50	**547**	1251					
	55	551	1376					
	60	601	1501					

3.2.3 DMD Performance

Figure 2 shows the samples and corrected result for the first DMD cycle using the optimum configuration found in the test described above, see Table 2. As for the one-dimensional diffusion case, one DMD cycle is sufficient to get a solution that is quite close to convergence.

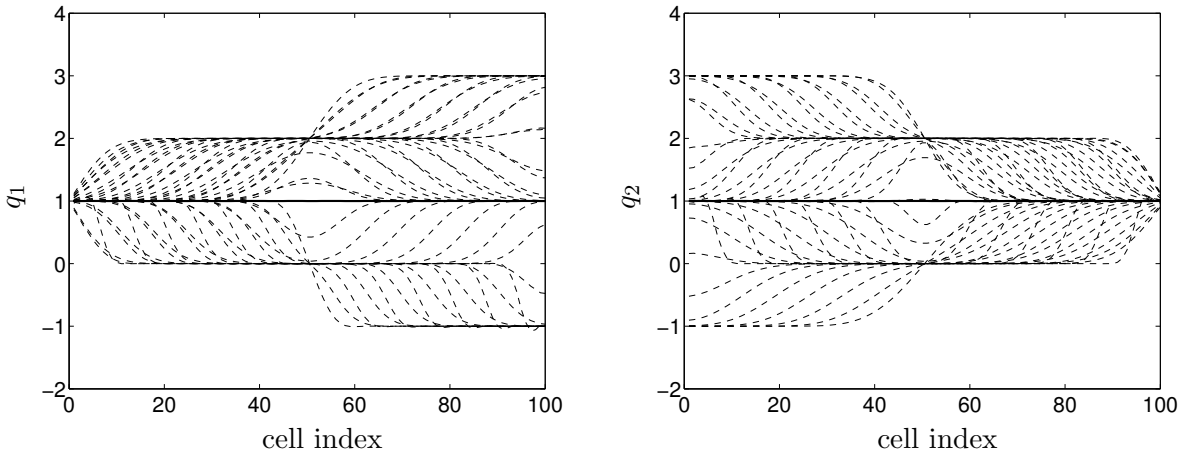


Figure 2: Samples of variable q_1 and q_2 (dashed lines) and the corrected solutions after the first DMD cycle (solid lines)

3.3 2D Cascade Flow

In this section, the DMD-based acceleration technique is used for accelerating the convergence of a steady-state Reynolds-Averaged Navier-Stokes (RANS) simulation for a low-Mach-number turbine cascade configuration. The code used for this test is a CFD code for compressible flows called VolSol^{++}, jointly developed by Chalmers and GKN Aerospace Sweden AB (former Volvo Aero Corporation). VolSol^{++} is based on the G3D family of codes developed by Eriksson [16], which has previously successfully been used for eigenmode extraction using Arnoldi and DMD methods [8, 9, 13]. The solver is based on a three-stage third-order explicit Runge-Kutta time-marching technique. Convective fluxes are evaluated using a third-order upwind scheme and the diffusive fluxes are calculated using a second-order centered-difference approach. The turbulence model is a standard realizable $k - \varepsilon$ model. Without the DMD-acceleration activated, i.e. the reference simulation, convergence was reached after 11610 iterations. In order to achieve significant acceleration using the DMD correction technique, a wide range of DMD settings were tested. In this case, finding optimum settings is not as straightforward as in the cases described in the previous sections. Preliminary studies indicated that, for this specific case, it was beneficial to activate the DMD acceleration after an initial sequence where the solver runs without any modifications. This is probably due to the fact that this removes modes only present in the start-up of the simulation that dominates the flow field at this stage. After these start-up modes has been damped out, the DMD technique is more likely to describe the modal content of the flow field and thus the flow field corrections will be more accurate. Of the wide variety of settings tested the best results in terms of solver speed-up were obtained when using the setup described in Table 3. As indicated in

Table 3 convergence was reached after three completed DMD cycles using these settings.

Table 3: Parameters defining the DMD settings that were used for the turbine cascade test case. N_P is the number of iterations that are completed before the sampling is initiated for each DMD cycle, N_S is the number of samples in each cycle and N_I is the number of iterations completed between each of these samples.

Cycle	N_P	N_S	N_I
1	900	19+2	56
2	200	18+2	120
3	200	18+2	140

Figure 3 shows a comparison of the residual history (continuity residuals) for the reference simulation and for the simulation with the DMD correction activated. The vertical dashed lines in the figure marked 1, 2 and 3 represent the three DMD corrections. As can be seen in the figure the corrections have significant effects on the residual history. This is especially true for the first correction, which makes the residual drop about one order of magnitude. The following two corrections do not leave as pronounced marks on the residual history but there are noticeable changes related to these corrections as well. The second correction leads to a significant reduction of residual fluctuations and the third correction is followed by another distinct drop in residual and shortly after this correction the convergence criteria is reached. This happens after 8216 completed iterations which should be compared to the 11610 iterations needed to reach convergence using the reference solver setup, i.e. the DMD correction technique reduces the number of iterations needed to reach convergence with 3394 or almost 30%.

The converged solutions obtained using the reference solver settings and the DMD-based correction method with settings as described above has been compared in order to ensure that the same solution is reached in both cases. Figure 4 shows a comparison of Mach number contours obtained from the two simulations. As can be seen there are absolutely no differences in these two solutions that can be detected by the naked eye. In order to make a more qualitative comparison, the pressure coefficient, C_p, and the skin friction coefficient, C_f, were calculated using estimates of blade surface pressure and wall friction obtained from the two simulations. A comparison of these quantities is shown in Figure 5 and again, there are no visible differences between the two data sets.
C_p and C_f are here defined as follows:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty {U_\infty}^2} \tag{25}$$

$$C_f = \frac{\tau_{wall}}{\frac{1}{2}\rho_\infty {U_\infty}^2} \tag{26}$$

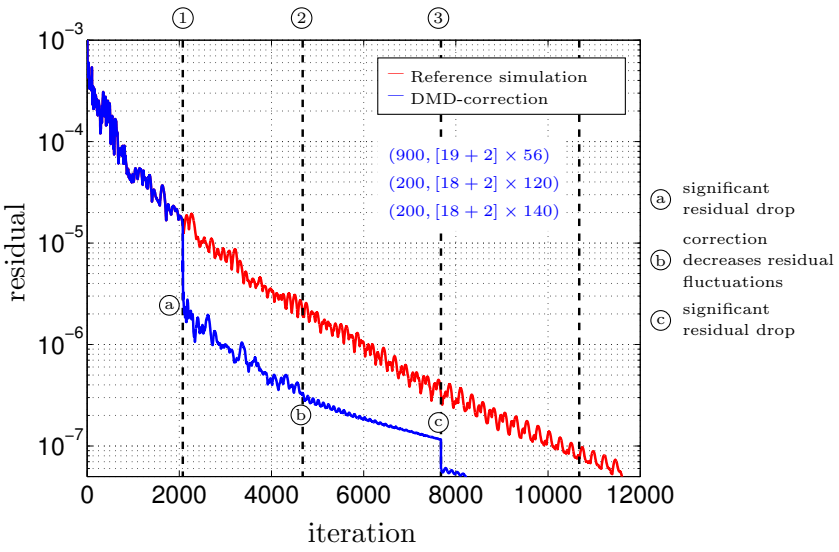
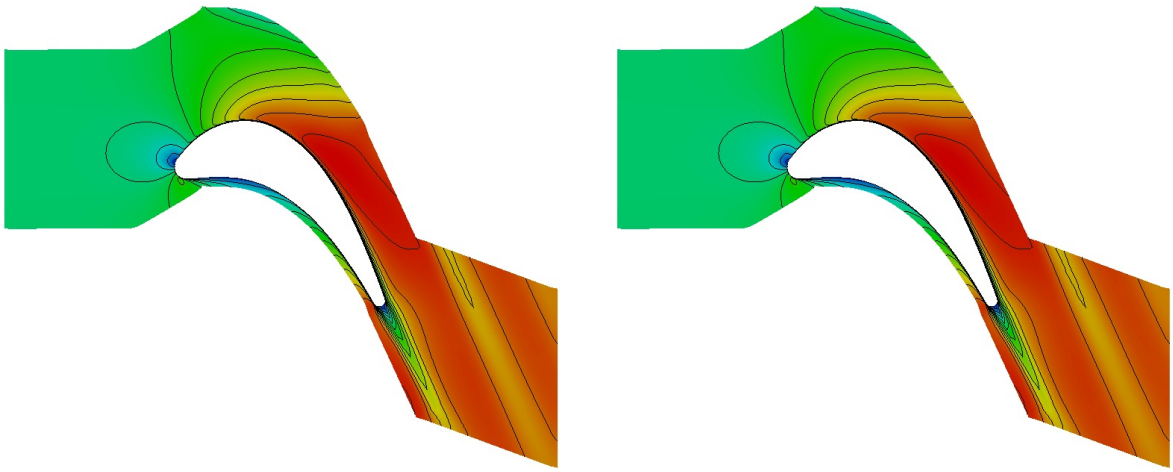


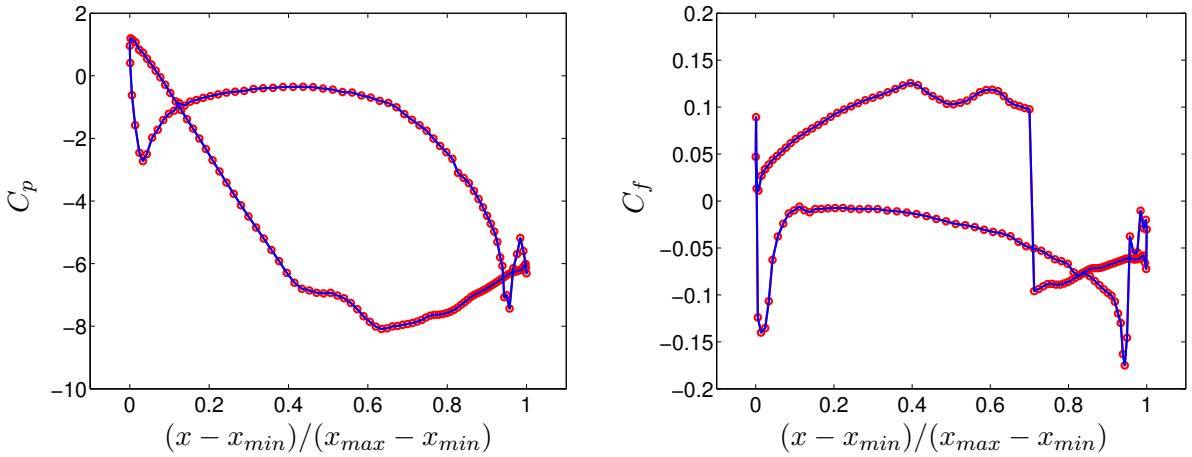
Figure 3: Convergence history. The solid red line represents the reference simulation, i.e. no correction, and the solid blue line represents the simulation with the DMD-based correction activated. The dashed vertical lines indicate the iterations where the DMD corrections are done.

Figures 6-8 shows the extracted corrections for each of the three completed DMD cycles. The corrections are represented by contours of density, internal energy and the cartesian momentum components in the x and y directions. These four variables along with the corresponding k and ε fields are added to the solver variables. It should be noted that the k and ϵ correction fields, not shown in Figures 6-8, are focused to the boundary layers and thus reveals less information about the modal content of the flow corrections. There are a few interesting differences in the contour plots obtained for the correction flow fields. First of all, it is quite evident that the levels of the variables representing the first of the three corrections are much higher than those in the two following corrections. For example if one compares the density values in the first correction field (Figure 6a) with those in the second correction field (Figure 7a), they are roughly 100 times higher in the first field. It is of course natural that the levels will decrease as the solution gets closer to the converged steady-state solution but the change in levels between the first two corrections is quite remarkable. This is of course related to the residual drop caused by the first correction (see Figure 3). Looking closer on the correction fields, it seems that the first correction is more related to some large-scale pressure fluctuation whereas the later corrections seems to be more related to boundary layer and wake flow (compare Figure 6a with Figures 7a and 8a). This is natural since it is the boundary layer that is usually most difficult to converge. Also, most likely the boundary layer and wake flow modes are hidden by the higher amplitude large-scale pressure modes when they are still present. Therefore, after the first correction when the high-amplitude modes are removed, the DMD algorithm is capable of catching the modes with lower amplitude.



(a) (b)

Figure 4: Mach contours for turbine cascade obtained using a RANS solver for compressible flows with and without the DMD-based acceleration technique activated. The left figure (a) shows Mach contours representing the converged solution obtained using the reference solver and the right figure (b) shows the corresponding quantity obtained from the converged solution for the solver using DMD-based correction.



(a) (b)

Figure 5: Comparison of turbine blade pressure coefficient, C_p, and skin friction coefficient, C_f. Solid red lines with red circles represent the reference simulation and the solid blue lines represent the simulation where DMD-based correction is utilized for solver acceleration.

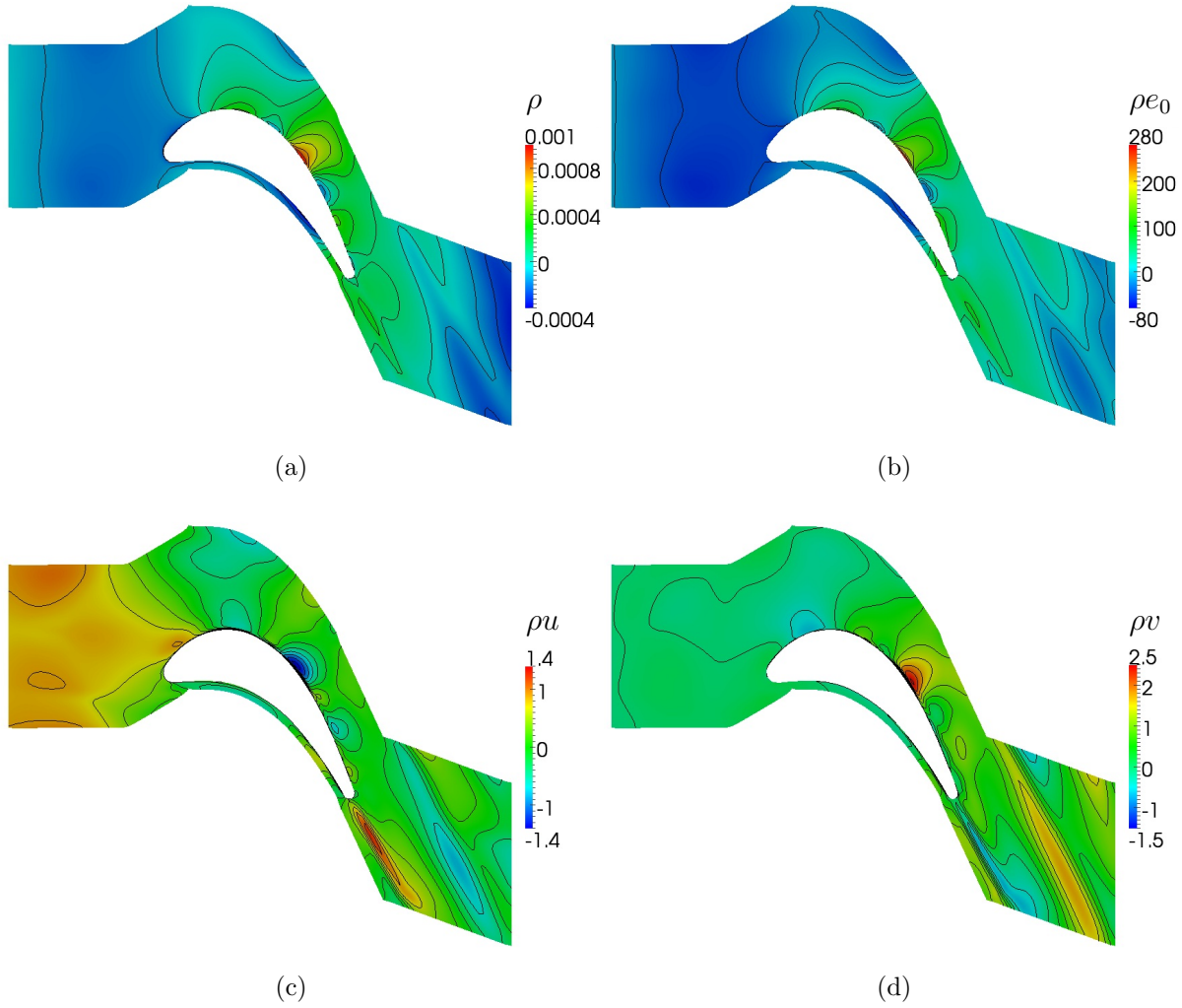


Figure 6: DMD correction number one represented by (a) contours of density ρ, (b) density times internal energy ρe_0, (c) the x-direction momentum component ρu and (d) the y-direction momentum component ρv. The first correction is done after 2076 iterations (see Figure 3).

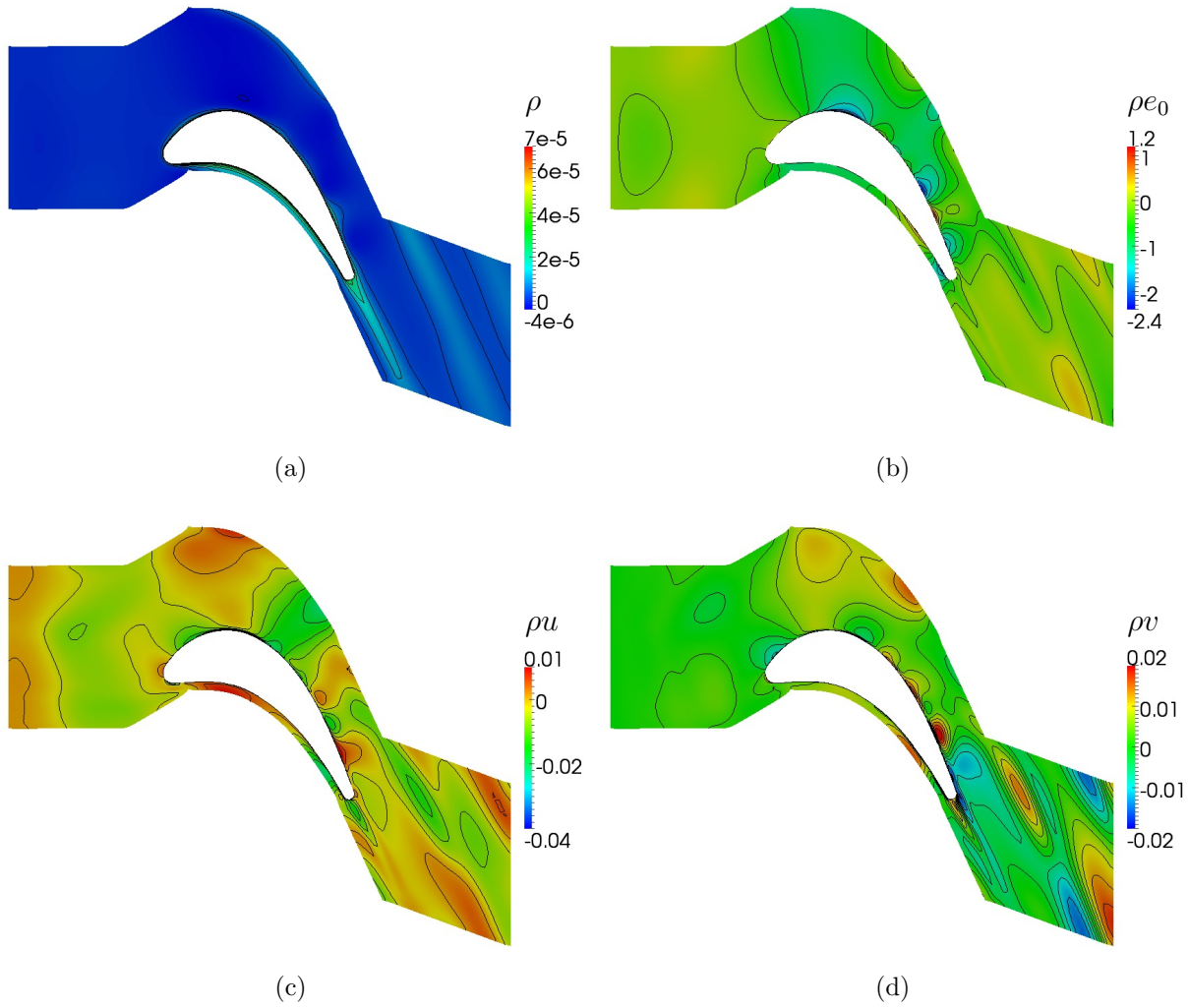


Figure 7: DMD correction number two represented by (a) contours of density ρ, (b) density times internal energy ρe_0, (c) the x-direction momentum component ρu and (d) the y-direction momentum component ρv. The second correction is done after 4676 iterations (see Figure 3).

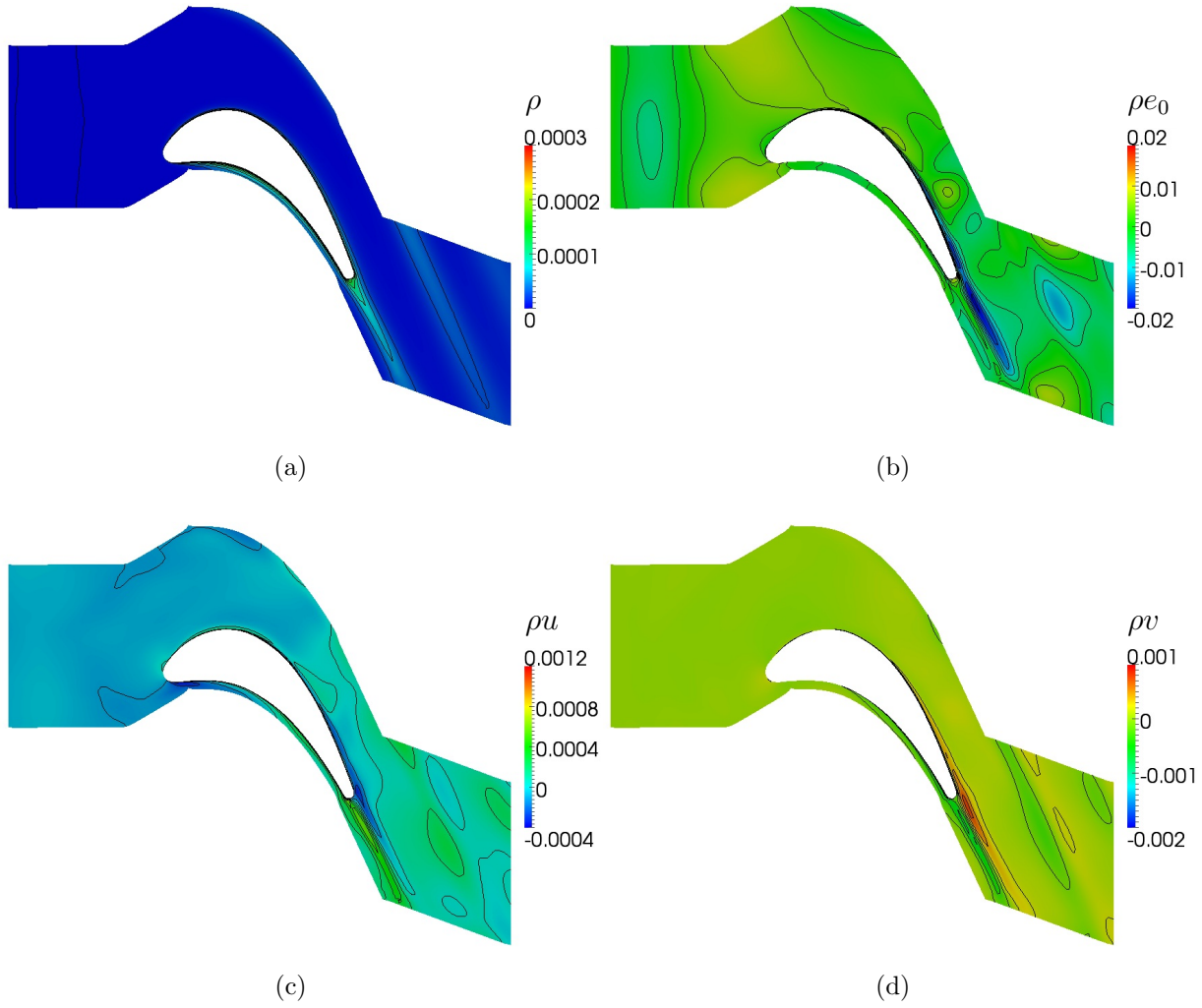


Figure 8: DMD correction number three, the last correction before reaching convergence, represented by (a) contours of density ρ, (b) density times internal energy ρe_0, (c) the x-direction momentum component ρu and (d) the y-direction momentum component ρv. The third correction is done after 7676 iterations (see Figure 3).

4 CONCLUSIONS

A solver speed-up technique for steady-state problems based on Dynamic Mode Decomposition (DMD) has been presented. In the proposed method, a Krylov subspaces built up from solver flow state snap shots is used to extract a correction that brings the solution closer to the converged steady-state solution. The snap shot sampling is done alongside an otherwise normal solver execution and the method does not interfere with the solver other than when the corrections are done. The new method has been tested for low-Mach-number turbine cascade configuration with promising results. For the specific solver setup presented in this article, the number of iterations needed to reach convergence was reduced from 11610 to 8216, i.e. a reduction of almost 30%. Since the implementation of the DMD-based acceleration is made such that solver overhead is almost negligible, the convergence time for the simulation with DMD correction activated is about 70% of that of the reference simulation. In order for solver speed-up based on the proposed method to be considered successful, more speed-up would be needed. It should be noted, however, that the maximum speed-up is limited by the need for a number of statistically independent samples in each Krylov subspace in order to be able to extract the most dominating modes. There is, however, room for improvement when it comes to the extraction of these modes. In the presented study, Gram-Schmidt orthogonalization is used in the DMD algorithm. The correction cycles could probably be made more efficient by using a DMD algorithm based on Singular Value Decomposition (SVD). Also, combining the DMD-based correction method with Implicit Residual Smoothing (IRSM) would probably be beneficial.

The setup of the DMD-based correction is very case dependent and the optimum settings for a specific case has to be found by testing, i.e. there is no generic way to find the best values for the parameters defining the DMD-based correction that will give the most efficient solver. However, in an engineering situation where often the same type of analysis is done repeatedly, the solver settings can be reused and thus the search for optimum settings will pay off. The DMD correction method is set up by defining the number of samples to be extracted in each correction cycle, N_S, the number of solver iterations between each of these samples, N_I and the number of iterations to be completed before starting the sampling of snap shots, N_P. Each correction cycle can be defined independently with specific values of N_S, N_I and N_P. The results presented for the low-Mach-number turbine cascade shows that as one correction cycle is completed, the nature of the problem can change significantly which justifies the use of independent definition of the correction parameters for each cycle.

REFERENCES

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

[2] Arnoldi, W. E., "The principle of minimized iteration in the solution of the matrix

eigenproblem," *Quart. Appl. Math.*, Vol. 9, 1951, pp. 17–29.

[3] Ruhe, A., "Rational Krylov Sequence Methods for Eigenvalue Computation," *Linear Algebra and its Applications*, Vol. 58, 1984, pp. 391–405.

[4] Eriksson, L.-E. and Rizzi, A., "Computer-Aided Analysis of the Convergence to Steady State of Discrete Approximations to the Euler Equations," *Journal of Computational Physics*, Vol. 57, 1985, pp. 90–128.

[5] Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P., and Henningson, D. S., "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, Vol. 641, 2009, pp. 115–127.

[6] Seena, A. and Sung, J., "Dynamic mode decomposition of turbulent cavity flows for self-sustained oscillations," *International Journal of Heat and Fluid Flow*, Vol. 32, 2011, pp. 1098–1110.

[7] Muld, T. W., Efraimsson, G., and Henningson, D. S., "Flow structures around a high-speed train extracted using Proper Orthogonal Decomposition and Dynamic Mode Decomposition," *Computers & Fluids*, Vol. 57, 2012, pp. 87–97.

[8] Jourdain, G., Eriksson, L.-E., Kim, S. H., and Sohn, C. H., "Application of dynamic mode decomposition of acoustic-modes identification and damping in a 3-dimensional chamber with baffled injectors," *Journal of Sound and Vibration*, 2013.

[9] Lárusson, R., Andersson, N., Eriksson, L.-E., and Östlund, J., "Linear Stability Analysis Using the Arnoldi Eigenmode Extraction Technique Applied to Separated Nozzle Flow," *49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, San Jose, CA, USA, 14-17 July 2013.

[10] Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, Vol. 7, No. 3, 1986, pp. 856–869.

[11] Nigro, N., Sorti, M., Idelsohn, S., and Tezduyar, T., "Physics based GMRES preconditioner for Compressible and Incompressible Navier-Stokes Equations," *Comput. Methods Appl. Mech. Engrg.*, Vol. 154, 1998, pp. 203–228.

[12] Stridh, M. and Eriksson, L.-E., "Solving harmonic linear problems in unsteady turbomachinery flows using a preconditioned GMRES solver," *ECCOMAS CFD 2006*, Egmond an See, Holland, Sept. 2006.

[13] Jourdain, G. and Eriksson, L.-E., "Numerical comparison between the Dynamic mode decomposition and the Arnoldi extraction technique on an afterburner test case," *18th AIAA/CEAS Aeroacoustic Conference*, Vol. AIAA 2012 of *2012-2147*, Colorado Springs, Colorado, USA, 04-06 June 2012.

[14] Hernandez, V., Roman, J. E., and Vidal, V., "SLEPc: A Scalable and Flexible Toolkit for the Solution of Eigenvalue Problems," *ACM Trans. Math. Software*, Vol. 31, No. 3, 2005, pp. 351–362.

[15] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, Birkhäuser Press, 1997, pp. 163–202.

[16] Eriksson, L.-E., "Development and Validation of Highly Modular Flow Solver Versions in G2DFLOW and G3DFLOW," Internal report 9970-1162, Volvo Aero Corporation, Trollhättan, Sweden, 1995.