Computationally efficient simulation of unsteady aerodynamics using POD on the fly

Ruben Moreno-Ramos^{1,2,3}, José M Vega² and Fernando Varas²

¹ Gulfstream Aerospace Corporation, Savannah, GA 31408, USA

² E.T.S.I. Aeronáutica y del Espacio, Universidad Politécnica de Madrid, E-28040 Madrid, Spain

E-mail: ruben.morenoramos@altran.com

Abstract

Modern industrial aircraft design requires a large amount of sufficiently accurate aerodynamic and aeroelastic simulations. Current computational fluid dynamics (CFD) solvers with aeroelastic capabilities, such as the NASA URANS unstructured solver FUN3D, require very large computational resources. Since a very large amount of simulation is necessary, the CFD cost is just unaffordable in an industrial production environment and must be significantly reduced. Thus, a more inexpensive, yet sufficiently precise solver is strongly needed. An opportunity to approach this goal could follow some recent results (Terragni and Vega 2014 *SIAM J. Appl. Dyn. Syst.* **13** 330–65; Rapun *et al* 2015 *Int. J. Numer. Meth. Eng.* **104** 844–68) on an adaptive reduced order model that combines ‘on the fly’ a standard numerical solver (to compute some representative snapshots), proper orthogonal decomposition (POD) (to extract modes from the snapshots), Galerkin projection (onto the set of POD modes), and several additional ingredients such as projecting the equations using a limited amount of points and fairly generic mode libraries. When applied to the complex Ginzburg–Landau equation, the method produces acceleration factors (comparing with standard numerical solvers) of the order of 20 and 300 in one and two space dimensions, respectively. Unfortunately, the extension of the method to unsteady, compressible flows around deformable geometries requires new approaches to deal with deformable meshes, high-Reynolds numbers, and compressibility. A first step in this

direction is presented considering the unsteady compressible, two-dimensional flow around an oscillating airfoil using a CFD solver in a rigidly moving mesh. POD on the Fly gives results whose accuracy is comparable to that of the CFD solver used to compute the snapshots.

Keywords: aeroelasticity, unsteady aerodynamics, transonic flow, shock waves, reduced order model, POD on the fly

(Some figures may appear in colour only in the online journal)

1. Introduction

The prediction of steady forces over lifting surfaces and aerodynamic bodies by means of computational fluid dynamics (CFD) is becoming a standard in aeronautical industry. In fact, CFD allows for accurately predicting the vehicle lift, drag, loads, and handling qualities, among others. Moreover, this is done with an affordable computational cost and the support of a decreasing number of testing campaigns, which are mainly needed to tune some solver parameters, e.g., associated with turbulence models.

The use of detailed CFD to predict unsteady aerodynamic forces is also intended to increase the accuracy of current, less precise methods used in production tasks, such as the doublet lattice method (DLM), which is based on potential theory and does not capture well nonlinear/viscous effects appearing in detached flows and shock waves. For aeroelastic simulations, however, detailed CFD requires a huge computational effort and its accuracy is not guaranteed by the large majority of current solvers, as a result of the complexity of the flow, as highlighted in recent meetings on the subject (Heeg *et al* 2013). The research community is making a tremendous effort on improving the unsteady CFD solvers available to the aeroelastic engineers. In some cases, a good correlation with wind tunnel tests has been reported (Moreno *et al* 2012), but the computational cost still prevents CFD from being implemented in industry, due to the large amount of simulations that are needed to account for different combinations of the parameters defining the vehicle configuration and flight conditions. On the other hand, increasing attention is being paid to *reduced order models* (ROMs). However, current ROMs for aeroelasticity still rely somehow on the linearisation of the aerodynamic forces. The resulting equations are treated using various reduced order modeling methods, such as proper orthogonal decomposition (POD) (Dowell *et al* 2001), Volterra kernels (Silva and Bartels 2004), and frequency response functions (Raveh 2004). Thus, these linearized models still exhibit some of the drawbacks mentioned above in connection with DLM.

Fully nonlinear ROMs have already been developed in various fields, including aerodynamics. These models take advantage of the well-known fact that, in extended systems (modeled by partial differential equations and systems), the number of *physically relevant degrees of freedom* is much smaller than the number of *numerical degrees of freedom* (i.e. grid points or cells) required by standard CFD solvers. This is because of the redundancies associated with the underlying physical laws (e.g., mass and momentum conservation in the Navier–Stokes equations). A large class of ROMs first identify the physically relevant degrees of freedom by using POD and extracting the most energetic POD modes, and then project the governing equations (or the CFD numerical scheme) onto these modes. Most ROMs based on this idea are designed to simulate attractors (not transients) and could be called *pre-processed ROMs* because POD modes are calculated from a CFD run in a pre-

process step. In principle, the CFD run should cover the relevant dynamics to be approximated by the ROM; this strategy can be improved using additional ingredients for fluid flows (Balajewicz *et al* 2013).

A different perspective has been tested for simpler fluid dynamics problems, such as the unsteady lid-driven cavity (Terragni *et al* 2011), and to simpler equations such as the complex Ginzburg–Landau equation (Rapún *et al* 2015), a paradigm of a pattern forming system that exhibits quite complex dynamic behaviors. The method consists in an adaptive strategy in which POD modes are calculated on demand, as the simulation proceeds. This strategy has proven to be very robust in both calculating particular solutions and constructing bifurcation diagrams (Terragni and Vega 2014), which require a huge number of particular time-dependent runs. The method uses both a CFD solver and a low dimensional system in interspersed time intervals and can thus be called a *POD on the Fly* method. The CFD solver is used to compute appropriate snapshots, which in turn are used to first calculate (at the outset of the temporal run) and later update (when needed) the POD manifold. The ROM, which as mentioned is obtained by projecting the governing equations onto the POD modes, is much less computationally expensive than the CFD solver and is intended to be used for the majority of the time. In other words, the adaptive strategy is implemented in a numerical scheme capable of switching seamlessly between the full integration of the governing equations and an adaptive ROM that guarantees capturing the present simulation dynamics. Note that such an adaptive strategy requires two main ingredients, associated with:

- The necessity of updating the POD manifold, which must be detected using only information from the ROM, via, e.g., appropriate *a priori* error/residual estimates.
- A convenient updating strategy of the POD modes at each updating event.

Additionally, the computational cost of projecting the governing equations (especially, nonlinear terms) can be further reduced if only a given set of cells, where the dynamics of the problem resides, are utilized. Such reduction of the computational cost has been used in the reduced modeling of several systems, arising in nonlinear elasticity (Ryckelynck 2009), steady aerodynamic forces (Alonso *et al* 2009, 2010), and unsteady incompressible flows (Terragni *et al* 2011). Mode libraries (both customized and generic) can also be used to further decrease the computational effort (Rapún *et al* 2015). In particular, for the complex Ginzburg–Landau equation, these ideas lead to acceleration CPU factors in the range 10–15 and 300–350 for simulations in one and two space dimensions, respectively (Rapún *et al* 2015).

This article presents the extension of the POD on the Fly method to the unsteady, compressible aerodynamic flows, including transonic flows, which is relevant in aeroelasticity. Such extension is not straight forward. In particular, the necessity of using moving meshes and the presence of shocks involve highly non-trivial difficulties that must be overcome to obtain a robust and computationally efficient ROM. In order to have full control over the CFD solver that will be combined on the fly with POD, a purposely defined CFD solver will be used that is described in section 2. The basics of POD-based reduced modeling are revisited in section 3, and the construction of a POD on the Fly ROM presented in section 4. An application of the developed method to the unsteady aerodynamics around a two-dimensional airfoil is addressed in section 5, considering both a subsonic and a transonic flow. The paper ends with some concluding remarks in section 6.

2. The unsteady CFD solver

The computational cost of the CFD solver will be reduced by neglecting viscosity, which avoids both the expensive calculation of the stress tensor and the necessity of either considering a huge number of cells or including turbulence modeling. Neglecting viscous terms might be seen as a loss of accuracy but, in practice, the absence of dissipation in the fluid solver poses an additional challenge to the stability of the simulation.

In a general coupled fluid-structure interaction problem, the position and speed of each point and cell in the computational domain Ω depend on the body's motion. At the same time, the body motion depends on the elastic, aerodynamic, inertial, and control forces, as well as the forces acting on the shared boundary with the fluid. However, in this paper we consider the simpler case in which the solid moves as a rigid body.

For three-dimensional, compressible aerodynamic flows, the governing equations account for mass, momentum, and energy conservation and can be written as

$$\frac{\partial(\mathcal{V}_i(\bar{x}, t) Q_i)}{\partial t} + \mathcal{R}_i(Q, \bar{x}) = 0, \quad (1)$$

where, the deformable fluid domain Ω is discretized in $i = 1, \dots, M$ cells of volume $\mathcal{V}_i$ and, with the usual notation, $Q = (\rho, u, v, w, p)$ is the *state vector*, whose components are the density, the three velocity components, and the pressure of the fluid, respectively, and $\mathcal{R}$ includes pressure and convective terms. The time-dependent generalized coordinates of the body are prescribed.

As anticipated, the intention is to construct a ROM based on the CFD solver. This requires to have full control over the flow solver residual calculation, boundary condition enforcement, time integration and flow reconstruction. Hence, already available codes (OpenFoam[®] and SU2, to cite some open source examples) are discarded and a new fluid dynamics solver developed following an arbitrary Lagrangian–Eulerian (ALE) formulation. A cell-centered finite volume scheme, based on the work from Greenshields *et al* (2010), has been chosen due to both its high quality to computational cost ratio and its simplicity in handling unstructured meshes.

Time evolution of the fluid state is achieved using an implicit Euler algorithm that includes a pseudo time sub-iteration step, namely

$$\mathcal{V}_i \left(\frac{Q_i^{m+1} - Q_i^m}{\Delta\tau} + \frac{Q_i^{m+1} - Q_i^n}{\Delta t} \right) + \mathcal{R}_i^m + \frac{\partial \mathcal{R}_i^n}{\partial Q} (Q_i^{m+1} - Q_i^m) = 0, \quad (2)$$

where $\Delta\tau$ varies from one cell to another, imposing a constant CFL number, $(\mu\Delta\tau/\Delta x)$ (Courant *et al* 1967). The numerical fluxes in $\mathcal{R}_i^n$ are computed using a Kurganov/Tadmor scheme (Kurganov and Tadmor 2000) and the Jacobian $\frac{\partial \mathcal{R}_i^n}{\partial Q}$ uses a diagonal approximation. The solution is iterated in m until $\|Q^{m+1} - Q^m\|$ is appropriately small, using a Newton method

$$\left(\frac{\mathcal{V}_i}{\Delta\tau} + \frac{\mathcal{V}_i}{\Delta t} + \frac{\partial \mathcal{R}_i^n}{\partial Q_i} \right) (Q_i^{m+1} - Q_i^m) = - \left(\mathcal{R}_i^m + \mathcal{V}_i \frac{Q_i^m - Q_i^n}{\Delta t} \right). \quad (3)$$

As convergence is achieved, the forces acting on the body are calculated for recording purposes and the body state is evolved in time using the prescribed motion. The motion of the body rotates and translates the computational domain mesh, whose new definition is used to further carry on the fluid integration in the next time step.

3. The POD-based ROMs

The simplest POD-based ROM, constructed from the CFD solver described above, involves two steps: (i) application of POD to a set of CFD-calculated snapshots, which gives a set of POD modes, and (ii) projection of the CFD scheme onto the POD modes.

3.1. POD computation

Once the CFD scheme has been applied for a convenient number of time steps in an appropriate timespan, a set of $N < M$ flow snapshots is considered for appropriate values of $t = t_1, \dots, t_N$. Denoting hereafter, vectors (i.e., column matrices) and general matrices with boldface lower case and capitals, respectively, the discretized state vector (denoted above as $\mathcal{Q}$) giving the five flow variables at the M cells is written as

$$\mathbf{q} = [\rho_1, \dots, \rho_M, u_1, \dots, u_M, v_1, \dots, v_M, w_1, \dots, w_M, p_1, \dots, p_M]^T, \quad (4)$$

where T denotes the transpose. Then, the $5M \times N$ snapshots matrix, whose columns are the snapshots, is

$$\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_N], \quad \text{with } \mathbf{s}_j = \mathbf{q}(t_j) - \mathbf{q}_0 \quad \text{for } j = 1, \dots, N, \quad (5)$$

where $\mathbf{q}_0$ can be selected as, e.g., the mean flow in an appropriate time interval.

For a given inner product $\langle \cdot, \cdot \rangle$, the POD modes of a given set of snapshots, $\mathbf{s}_1, \dots, \mathbf{s}_N$, can be calculated using the covariance matrix $\mathbf{R}$, whose elements are

$$R_{ij} = \langle \mathbf{s}_i, \mathbf{s}_j \rangle \equiv \mathbf{s}_i^T \mathbf{G} \mathbf{s}_j \quad \text{or} \quad \mathbf{R} = \mathbf{S}^T \mathbf{G} \mathbf{S}, \quad (6)$$

where the matrix $\mathbf{G}$ is the metric tensor associated with the inner product. The spectral decomposition of $\mathbf{R}$ is expressed as

$$\mathbf{R} = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T, \quad \text{with } \mathbf{V}^T \mathbf{V} = N \times N \text{ unit matrix}, \quad (7)$$

where the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{R}$ and $\mathbf{\Sigma}^2$ is the diagonal matrix of the eigenvalues of $\mathbf{R}$, which are the squares of the singular values σ_j and are usually sorted in decreasing order. Namely

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N. \quad (8)$$

Then, the POD modes are the columns of the $5M \times N$ matrix- Φ , defined as

$$\Phi = \mathbf{S} \mathbf{V} \mathbf{\Sigma}^{-1}, \quad (9)$$

which is called the *POD-modes matrix*. It turns out that the POD modes are orthonormal with the inner product $\langle \cdot, \cdot \rangle$. An additional important property of the POD modes is that, if the snapshot matrix is projected onto the $N_0 < N$ most energetic POD modes, the error can be estimated as

$$\text{RRMSE}_n^N = \sqrt{\frac{\sum_{i=N_0+1}^N (\sigma_i)^2}{\sum_{i=1}^N (\sigma_i)^2}}. \quad (10)$$

However, due to the fact that the covariance matrix involves the multiplication of the snapshots, the POD modes associated with singular values such that $(\sigma_j)/(\sigma_1) < 10^{-8}$ are meaningless in typical double precision calculations. This is improved as follows. The singular value decomposition (SVD) is applied to the modified snapshots matrix

$$\tilde{\mathbf{S}} = \mathbf{G}^{1/2} \mathbf{S}, \quad (11)$$

which yields

$$\tilde{\mathbf{S}} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T, \quad \text{with} \quad \mathbf{U}^T \mathbf{U} = \mathbf{V}^T \mathbf{V} = N \times N \text{ unit matrix.} \quad (12)$$

Substituting these into (11) and solving the resulting equation for $\mathbf{S}$ gives

$$\mathbf{S} = \mathbf{G}^{-1/2} \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T. \quad (13)$$

A further substitution of these into the definition of the covariance matrix (6), taking into account that $\mathbf{D}$ and $\mathbf{G}$ are diagonal matrices, yields

$$\mathbf{R} = \mathbf{S}^T \mathbf{G} \mathbf{S} = \mathbf{V} \mathbf{\Sigma} \mathbf{U}^T \mathbf{G}^{-1/2} \mathbf{G} \mathbf{G}^{-1/2} \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T. \quad (14)$$

Finally, comparing (14) and (7) and invoking (9) leads to the same expression for the POD-modes matrix obtained above, namely

$$\Phi = \mathbf{S} \mathbf{V} \mathbf{\Sigma}^{-1}. \quad (15)$$

In typical applications, the total number of cells may well be of the order of several million, while the number of retained modes can be of the order of several hundred. This dimension reduction is the main benefit of the method.

3.2. Projection of the governing equations onto the POD modes

For a given set of selected (most energetic) modes, the state vector of the flow field, defined as in (4), can be expressed as a linear combination of the N_0 retained modes, as

$$\mathbf{q}(t) = \mathbf{q}_0 + \Phi \xi(t), \quad (16)$$

where $\mathbf{q}_0$ is as defined in (5) and the vector ξ contains the unknown mode amplitudes. The aim is to project the $5M$ scalar components of the governing equation onto these N_0 modes, which gives N_0 nonlinear scalar equations for the unknown amplitudes, instead of the original $5M$ equations. Such projection can be made in various ways. *Galerkin projection* consists in replacing (16) into the governing equations (2) and orthogonally projecting the residual of the resulting equations onto the POD modes. A projection that is more consistent with the fully implicit CFD solver described in the last subsection consists in minimizing (using, e.g., a Newton method) the least squares residual, which leads to the following optimization problem

$$\min \left(\sum_{k=1}^{M_0} \mathcal{V}_{i_k} \left[\sum_{j=1}^{N_0} \Phi_j^{i_k} \left(\frac{\xi_j^{j,n+1} - \xi_j^{j,n}}{\Delta t} \right) + \mathcal{R}_{i_k}(\mathcal{Q}_o + \Phi_j \xi_j^{j,n+1}) \right]^2 \right), \quad (17)$$

where the index i_k labels $M_0 \leq M$ properly selected cells, called here *master cells*. This projection has been already used in steady aerodynamics (Alonso *et al* 2009, 2010) and will be called the *least squares projection*. It must be noted that the various ingredients in the objective function in (17), including the residual appearing in the second term can be reconstructed (via appropriate handling of (15) and (16)) using only information from the master cells. Thus, it is essential that M_0 be much smaller than the total number of cells, M ; a value of M_0 comparable to the number of selected POD modes (say, $M_0 = 3N_0$) may be enough provided that the master cells be appropriate (see below). Such selection should also help to avoid the *mode truncation instability*, which is a major difficulty in the field. Consistently with this, the inner product used to define POD modes may be an inner product based precisely on the master cells. Namely, the metric tensor $\mathbf{G}$ appearing in (6) is diagonal and their elements are all zero except for

Figure 1. Adaptive ROM flowchart.

$$G_{j_k j_k} = 1 \quad \text{for } k = 1, \dots, 5M_0, \quad (18)$$

where the index k labels the $5M_0$ components of the state vector (4) associated with the M_0 master cells. In other words, the inner product is also defined as

$$\langle \mathbf{q}^1, \mathbf{q}^2 \rangle = \sum_{k=1}^{M_0} (\rho_{i_k}^1 \rho_{i_k}^2 + u_{i_k}^1 u_{i_k}^2 + v_{i_k}^1 v_{i_k}^2 + w_{i_k}^1 w_{i_k}^2 + p_{i_k}^1 p_{i_k}^2), \quad (19)$$

where the index k labels now the master cells.

Let us now turn into the selection of the master cells. Noting that the non-selected cells fluid variables evolution is made slave to the selected ones, the selection requires some care, especially in connection with an adequate imposition of the boundary conditions. In other words, it is important to include in this selection those regions where significant and unique dynamics may exist. On the other hand, it is typical of CFD solvers to accumulate errors in

specific zones of the computational domain, such as the trailing edge of lifting surfaces. Ignoring these regions, the robustness and significance of both the extracted modes and the least squares projection are both increased.

4. Adaptive POD on the Fly

This methodology, already tested and evolved in a series of papers (Alonso *et al* 2009, 2010, Terragni *et al* 2011, Terragni and Vega 2014, Rapún *et al* 2015) for related problems (e.g., steady aerodynamics, low-Reynolds unsteady flows, and complex dynamical systems), is a very good alternative to standard *pre-processed POD-based ROMs*. In these, the snapshots to create the ROM are obtained from a CFD simulation in the complete time history. This means that the CFD cost needed to calculate the snapshots may be prohibitive. Also, these ROMs allow to simulate problems with similar dynamics to the previous solutions, but they are not able to capture behaviors of the system not previously encountered. Also, they may suffer from mode truncation stability issues, mainly due to the fact that the full numerical scheme can be stable, but the projection on a truncated basis is unstable because of the neglected higher order modes.

The new methodology, introduced for unsteady aerodynamics in this work, overcomes the above mentioned drawbacks of pre-processed ROMs by switching between the ROM solver and the full numerical solver (see figure 1) as needed. Initially, the integration of the governing equations starts using the full numerical solver. Then, after a prescribed number of time steps have been completed, a POD basis is obtained and truncated to a number N_0 of modes in such a way that

$$\text{RRMSE}_{N_0}^N \leq \varepsilon_1, \quad (20)$$

where $\text{RRMSE}_{N_0}^N$ is estimated in (10), and ε_1 is a tunable parameter.

4.1. Monitoring the ROM approximation

At this point, the integration is switched to the ROM solver, and the system evolved in time, until the ROM loses its validity due to truncation instabilities or changes in system dynamics. It is therefore paramount to introduce in the time evolution the ability to detect this phenomenon, which is accomplished here by imposing that a conveniently defined residual be appropriately small. In other words, the ROM integration terminated (and the POD manifold updated, see below) if the following condition holds

$$E_{\text{res}} = \frac{\left(\sum_{k=1}^{M_0} \nu_k \left[\sum_{j=1}^{N_0} \Phi_j^k \left(\frac{\xi^{j,n+1} - \xi^{j,n}}{\Delta t} \right) \right]^2 \right)^{1/2}}{\|\mathcal{R}(\mathcal{Q}_o + \Phi_j^i \xi^{j,n+1})\|} > \frac{\varepsilon_1}{K}. \quad (21)$$

Here, K is a tunable parameter and $\|\cdot\|$ is the norm associated with the previously defined inner product (19). Note that the left hand side of this condition (which, as it happened with (17), can be calculated using information from only the master cells) is the ratio between the two different components that form the solution residual, i.e. the time derivative $\partial_t \mathcal{Q}$ and the flow solver residual $\mathcal{R}$, and can be seen as a scaled measure of the residual. If this condition holds, then the governing equations are not conveniently satisfied by the low-dimensional approximation.

4.2. Updating the POD modes

Condition (21) controls when the former POD modes $\hat{\phi}_1, \hat{\phi}_2, \dots$ (collected in a matrix denoted here as $\hat{\Phi}$) are not convenient and need to be updated. To this end, the CFD solver is run (using the last ROM outcome as initial condition) for a convenient timespan ΔT^{CFD} , where some new snapshots are computed, POD applied, and a new set of POD modes $\phi_1, \phi_2, \dots$, (organized in a new matrix Φ) are calculated. Since some of the previous system dynamics are most likely part of the system evolution, the former snapshots need not be completely calculated, but slightly rotated. This is done by combining the columns of $\hat{\Phi}$ and Φ to create a new set of POD modes, which result by applying POD to the set of vectors

$$\hat{\nu}_1 \hat{\phi}_1, \dots, \hat{\nu}_{N_{\text{old}}} \hat{\phi}_{N_{\text{old}}}, \nu_1 \phi_1, \dots, \nu_{N_{\text{new}}} \phi_{N_{\text{new}}}. \quad (22)$$

The weights $\hat{\nu}_j$ and ν_j are defined as

$$\hat{\nu}_j = \frac{\hat{\sigma}_j}{\sqrt{\sum_{i=1}^{N_{\text{old}}} (\hat{\sigma}_i)^2}}, \quad \nu_j = \frac{\sigma_j}{\sqrt{\sum_{i=1}^{N_{\text{new}}} (\sigma_i)^2}}, \quad (23)$$

where $\hat{\sigma}_j$ and σ_j are the singular values in the former and new applications of POD. In this way, those modes with higher relative ‘energy’ (based on the inner product) are retained in the newly created ROM, and the process is ready to start over again.

4.3. Summary of the adaptive POD on the Fly ROM

The proposed ROM proceeds in five steps, as follows:

- (i) The CFD solver described in section 2 is applied over a certain timespan $0 \leq t \leq T_0^{\text{CFD}}$. Snapshots are defined and POD applied as explained in section 3.1, which selects a set of POD modes.
- (ii) The governing equations are projected onto the selected modes as explained in section 3.2, which gives a low dimensional ROM.
- (iii) This ROM is integrated in a new time interval $T_0^{\text{CFD}} \leq t \leq T_1^{\text{ROM}}$, where the upper bound T_1^{ROM} is defined as explained in section 4.1, by condition (21).
- (iv) CFD is applied again in a new interval $T_1^{\text{ROM}} \leq t \leq T_1^{\text{ROM}} + \Delta T^{\text{CFD}} \equiv T_1^{\text{CFD}}$, where new snapshots are computed and a new set of POD modes calculated. The old and new POD modes are combined as explained in section 4.2, which gives a new set of updated POD modes.
- (v) Using the updated POD modes, steps (ii)–(iv) are repeated again and again (which defines new CFD and ROM timespans, $T_k^{\text{CFD}} \leq t \leq T_{k+1}^{\text{ROM}}$ and $T_{k+1}^{\text{ROM}} \leq t \leq T_{k+1}^{\text{CFD}}$, respectively, for $k = 1, 2, \dots$) until the end of the simulation.

5. Practical application: oscillating airfoil

The prediction of the aerodynamic forces that an oscillating airfoil immersed in a fluid flow experiences is one of the key problems in dynamic aeroelasticity. The analytical expressions for the aerodynamic forces acting on these airfoils, based on perturbation potential theory (Theodorsen and Center 1935), has made possible the prediction of instability boundaries of lifting surfaces in the origins of aeroelasticity. However, more precise simulations based on CFD are intended nowadays.

Figure 2. Computational domain (left) and computational mesh (right).

In the current example, the unsteady two-dimensional flow around an airfoil subject to either pitch or heave oscillations is considered; see below.

In order to illustrate the benefits of the novel approach presented in this article, such an oscillating airfoil has been coupled to the inviscid flow solver described in section 2. As anticipated, using a CFD solver based on the Euler equations represents a challenge for the construction of the POD on the Fly ROM.

5.1. Computational fluid domain and mesh

The aerodynamic shape has been chosen, without loss of generality, as a four digits symmetric, uncambered, NACA airfoil with a chord c and a 10% thickness to chord length ratio, known as NACA-0010 (Garrick 1933) (see Moran 1984 for an analytical expression giving the shape of the airfoil). This type of airfoil has been commonly used in computational and experimental aeroelastic problems (Batina 1985, Rivera *et al* 1992, Conner *et al* 1997).

The computational domain (see figure 2-left, where the free stream flow goes from left to right) is the smooth union of (i) a half circle with center at $x = 0.25c$ and radius $25c$ and (ii) a $25c \times 50c$ -rectangle whose downstream edges have been smoothed to avoid numerical singularities. Such domain is extruded to a thickness of 1 chord.

The computational mesh (see figure 2-right) has been generated using Gmsh, a three-dimensional mesh generator with built-in pre- and post-processing utilities (Geuzaine and Remacle 2009). The mesh construction proceeds in three steps.

First, a one-dimensional mesh seed is created along the boundaries (both internal and external). 40 points are distributed using a 0.95 distance growth rate in the two upstream quarters of the airfoil, and 80 points in the remaining two quarters (near the trailing edge), with 1.12 growth rate. Each of the far-field upstream quarters has 40 points, with 1.03 growth rate, and its radius includes 80 points with 1.14 growth rate. Finally, the downstream boundary and upper/lower boundaries include 80 equidistant points each.

In a second step, the two-dimensional mesh (see figure 2-right) is created using transfinite interpolation (Gordon and Hall 1973).

Finally, since this two-dimensional problem is to be solved using the three-dimensional CFD solver constructed as explained in section 2, the two-dimensional mesh is extruded in the span direction to create the three-dimensional mesh consisting of 38 080 points, 75 446 faces, and 18 802 cells.

Figure 3. Sub-mesh containing the master cells (thick dots); the remaining cells are plotted with thin dots.

5.2. Sub-mesh containing the master cells and tunable parameters

A substantial and critical part of the novel adaptive ROM developed here relies on the use of a limited number of *master cells*, which are used to both apply POD and project the governing equations. The master cells conform the sub-mesh that is developed here.

We insist that, in some sense, the remaining cells just act as ‘slaves’ of the master cells through the POD modes. Thus, the master cells must be distributed in the computational domain such that they allow the enforcement of the boundary conditions and the transmission of information between the different regions. They should as well contain the interesting dynamics of the problem. Regions with known numerical noise, like the trailing edge of lifting surfaces, should be avoided instead.

Several methods have been proposed in literature, such as *missing point estimation* (Astrid *et al* 2008) and *hyper-reduction* (Ryckelynck 2009). These methods somehow optimize the sub-mesh selection according to the expected dynamics and/or the nature of the operators acting on the POD modes. However, since the goal is to adapt to unknown possible dynamics, a more robust (and simpler) procedure is used here. A series of (radial) lines are traced at regular intervals from the airfoil wall to far-field faces, and 960 cells (see figure 3) in total are selected along these lines. No additional points are included to capture the shock that is present in the high transonic flow considered below. This is because the position of the shock is unknown, and the sub-mesh is selected beforehand here. Of course, an adaptive selection would be more convenient, but this is well ahead the scope of this paper.

This sub-mesh is used both to calculate the POD modes and to project the governing equations. Note, invoking (4) and (18), that the inner product (6) based on these $M_0 = 960$ cells, labeled with the index k , can be written as defined in (19). The least squares projection of the governing equations based on these M_0 mesh points (see section 3.2) is performed similarly.

Figure 4. Rescaled forcing function for the pitch (left) and heave (right) motions, plotting the rescaled angle of attack and height versus ft .

5.3. Oscillatory forcing and tunable parameters of the POD on the Fly method

The unsteady flow field is forced by monochromatic pitch or heave oscillations of the airfoil, with either the angle of attack or the vertical position of the airfoil, varying as $AoA = a[\cos(2\pi ft) - 1]$ or $h = a \cos(2\pi ft)$ (see figure 4). For convenience, the reduced frequency (which is nondimensional and proportional to the Strouhal number), is defined in terms of the frequency f , the incoming velocity v , and the chord c as

$$k = \pi fc / v. \quad (24)$$

As anticipated, we consider pitch and heave motions, with the following combinations of the forcing amplitude

$$\text{pitch: } a = 0.01 \text{ and } 0.1 \text{ rad,} \quad \text{heave: } a = 0.05c \quad (25)$$

and two forcing frequencies, one giving a small value of the reduced frequency k and another giving $k \sim 0.5$. Thus, we have six combinations for the forcing mode, which moreover will be considered in the next two subsections for a subsonic ($M = 0.2$) and a transonic ($M = 0.8$) case.

The CFD code has been validated for small amplitude, subsonic, pitching motions against the analytical expressions from (Theodorsen and Center 1935). In particular, with $M = 0.2$ and a forced pitching with reduced frequency $k = 0.116$, a lift coefficient $C_{l\alpha} = 5.2 + 0.3i$ is obtained with our CFD code, while the analytical prediction by Theodorsen and Center is $C_{l\alpha} = 5.2 + 0.1i$. Validation in the transonic regime, also for small motions, is made against results reported in (Batina 1985). For $M = 0.8$, with $k = 0.115$, the CFD code produces a lift coefficient due to pitch $C_{l\alpha} = 5.8 - 3.2i$ while Batina reports a value $C_{l\alpha} = 5.7 - 4.3i$. With $M = 0.8$ and a larger reduced frequency, $k = 0.462$, the corresponding values for $C_{l\alpha}$ obtained with the CFD code and reported by Batina are $4.1 + 0.1i$ and $4.5 + 0.2i$, respectively.

5.4. Results for the time dependent subsonic flow at $M = 0.2$

As a first case to test the adaptive POD on the Fly methodology, we consider the low subsonic regime, $M = 0.2$, at sea level altitude. This is not a very demanding test problem, but helps to illustrate the main ideas behind the methodology.

The timespan of the first CFD interval in which the POD modes are first computed, the CFD update run duration, and the tunable parameters of the problem appearing in (21) are selected for the subsonic case considered here as

Figure 5. POD on the Fly versus the CFD solver, with $M = 0.2$ at sea level, considering pitch motion with $a = 0.01$ rad and $f = 2.5$ Hz. Top: the nondimensional lift versus time, with those intervals in which the low dimensional ROM is used in the POD on the Fly method indicated with shaded regions. Bottom: the residuals versus time.

$$T_0^{\text{CFD}} = 0.1 \text{ s}, \quad \Delta T^{\text{CFD}} = 0.1 \text{ s}, \quad \epsilon_1 = 10^{-8}, \quad K = 100. \quad (26)$$

In order to emphasize the robustness of the method, these tunable parameters will be maintained in all cases considered below in the subsonic case. Moreover, the CFD update run duration, ΔT^{CFD} , will be maintained as constant along the simulation. The performance of the POD on the Fly method would be increased by varying this parameter along the simulation, noting that as more updatings are performed, the quality of the POD modes increases and thus the subsequent CFD update intervals could be decreased. Such selection of ΔT^{CFD} could be dynamically chosen along the simulation in an adaptive way (see (Rapún *et al* 2015) and references therein), but this improvement is beyond the scope of this paper.

For the pitch oscillations at the lowest forcing amplitude in (25) and a small forcing frequency, $f = 2.5$ Hz, which in the present case gives a reduced frequency $k = 0.12$, the outcome of the POD on the Fly method is given in figure 5. The curves labeled NS are obtained using the CFD solver alone, while those labeled ROM are used combining the CFD solver and the low-dimensional ROM, using the POD on the Fly method. As it can be seen, the length of the time intervals during which the ROM is used (shaded region in figure 5) steadily increases along the simulation, meaning that the updating strategy described in section 4.2 is very efficient to improve the POD modes as more information from the CFD solver (namely, a larger number of snapshots) is accounted for. The CPU acceleration factor, defined as

Figure 6. Counterpart of figure 5-top for $\alpha = 0.1$ rad and $f = 10$ Hz ($k = 0.48$).

Figure 7. Complete color maps (top) and blow up near the leading edge (bottom) of the instantaneous (rescaled) pressure distributions of the run in figure 6 at the minimum angle of attack ($\beta t = 1/2$, see figure 4), as obtained using the CFD alone (left) and the POD on the Fly method (right).

$$\text{CPU acceleration} = \frac{\text{NS CPU time}}{\text{ROM CPU time}}, \quad (27)$$

1.6 in the considered time interval ($0 \leq t \leq 1.5$ s) and still larger in larger time intervals (because the length of the intervals in which the ROM is used steadily increases). The upper plot also shows that the lift provided by both, the CFD alone and the POD on the Fly methods are plot-indistinguishable. Plot indistinguishable approximations mean hereinafter that relative errors are smaller than 1%. Moreover, the ROM lift curve does not show any artifacts at those values of t in which the POD on the Fly method switches between the CFD solver and the ROM. All these are consistent with the very small, comparable residuals of both, the CFD solver and the POD on the Fly ROM (lower plot). In fact, the plots of the residuals versus time will be qualitatively similar in the remaining results below and thus the counterpart of the lower plot will be omitted.

In the subsonic case, the performance of the ROM slightly worsens as either the pitch amplitude or frequency are increased. However, even for the largest selected amplitude in (25) and the largest considered frequency, the performance of the POD on the Fly method is still reasonable. This is seen in figure 6, which shows that the total length of the time intervals in which the low dimensional ROM is used in the POD on the Fly method is smaller than in figure 5. Consequently, the CPU acceleration factor defined in (27) is also smaller, namely CPU acceleration ~ 1.3 . However, the approximation of the lift is as good as in the former case and, moreover, the pressure field is also very well approximated, which is seen in figure 7. Note that both pressure fields are plot indistinguishable.

The results above are concerned with pitch motions. Several tests for heave motions (omitted here) have shown that these motions are less demanding than pitch motions for the POD on the Fly performance. This is due to the fact that the airfoil heave motion produces a constant incremental downwash along the airfoil chord.

Summarizing, the POD on the Fly method simulates the flow in a robust and efficient way. In particular, the truncation instability is avoided and the updating strategy recognizes well the relevant dynamical information. As expected, the performance of the POD on the Fly method slightly worsens as either the forcing amplitude or frequency increases. However, the results are reasonably good even for the worst considered case.

5.5. Results for the transonic flow at $M = 0.8$

Let us now increase the Mach number to $M = 0.8$, maintaining the sea level altitude to facilitate comparison with the subsonic case. For this transonic flow, the NACA0010 airfoil exhibits strong shocks both in the pressure and suction sides of the airfoil, even at zero angle of attack. The presence of such discontinuities makes the ROM operation more challenging, since it needs to predict the (quite noticeable) shock movement as the angle of attack varies. Moving discontinuities are specially demanding for POD because, in the end, the POD modes are linear combinations of the snapshots, and the linear combination of shifted discontinuities is not a discontinuity, but a stair-like structure. The difficulty was already illustrated for steady transonic aerodynamics in (Alonso *et al* 2010), where a specific (fairly involved and difficult to implement in three space dimensions) shock treatment was developed. Such treatment is not used here, meaning that it must be the updating strategy that adapts the POD modes to capture well the discontinuities. For the present transonic case, due to the presence of moving shocks, it is convenient that the updating events be more frequent than in the subsonic case. In other words, it is convenient that the updated POD modes correspond to not too different positions of the shocks. More frequent updating events can be forced by selecting smaller duration of the CFD update intervals, in which the new snapshots are calculated. Thus, a lower value of ΔT^{CFD} (than for the subsonic case) has been proved to show better ROM efficiency. Specifically, the tunable parameters of the POD on the Fly method appearing in (21) are now selected as

$$T_0^{\text{CFD}} = 0.1 \text{ s}, \quad \Delta T^{\text{CFD}} = 0.005 \text{ s}, \quad \epsilon_1 = 10^{-8}, \quad K = 100. \quad (28)$$

Comparison with the tunable parameters selected in the subsonic case (see equation (26)) shows that all tunable parameters coincide except for ΔT^{CFD} , which further emphasizes the robustness of the method.

In principle, since the ROM needs to capture the shock motion, one could expect that the performance of the POD on the Fly method worsens as either the forcing amplitude or frequency increases. Anticipating the results below, this conclusion turns out to be true for the forcing amplitude, but not in connection with the frequency. This is because large forcing

Figure 8. Counterpart of figure 5-top for $M = 0.8$, considering the following values of the forcing amplitude and frequency: $a = 0.01$ rad, $f = 2.5$ Hz, $k = 0.029$ (top); $a = 0.1$ rad, $f = 2.5$ Hz, $k = 0.029$ (middle) and $a = 0.1$ rad, $f = 40$ Hz, $k = 0.46$ (bottom). The CPU acceleration factors defined in equation (27) are 2.3, 1.7, and 2.2 in the upper, middle and lower plots, respectively.

frequencies do not necessarily promote a larger (back and forth) motion of the shock wave, but a smaller motion. In other words, for a given forcing amplitude a , the amplitude of the motion of the shock wave decreases as f increases, specially for $k \leq 0.12$ (namely, for $f \leq 10$ Hz). This should be due to the fact that for large forcing frequency the shock is not able to move as fast as the airfoil moves.

As in the former subsection, we first concentrate in pitch motions. For the transonic case, the maximum pitch amplitude at the maximum frequency is a too extreme case for which the CFD run does not converge. This is seemingly due to the fact that the considered mesh and time intervals are not fine enough for this case. Hence, the effects of the forcing amplitude and frequency are shown individually in figure 8. As can be seen, the performance of the ROM is qualitatively similar (in fact, even better in terms of the CPU acceleration factor) to that for the subsonic case considered in the former subsection. Note, in particular, that as expected the acceleration factor decreases as either the forcing frequency or the forcing amplitude increase, and it is more sensitive to increasing the forcing amplitude, as anticipated.

On the other hand, as also expected, the oscillations of the residual (not shown in the figure) are somewhat larger and the timespans in which the ROM approximation is acceptable

Figure 9. Counterpart of figure 7 for the run considered in figure 8-middle plot at times with a minimum (top) and zero (bottom) angle of attack (namely, at $ft = 1/2$ and $ft = 1$, respectively, see figure 4).

Figure 10. POD modes number evolution along the various timesteps performed by the POD on the Fly ROM for the $M = 0.8$ case.

are smaller (and evolve somewhat irregularly) than in the subsonic case. However, the POD on the Fly ROM still works fairly well. In particular, the relevant POD modes are efficiently captured as time increases. It is also obvious in the figure that the length of the intervals in which the low-dimensional ROM is used is fairly small after the first computation of the POD modes. This is because the POD modes do not contain initially enough information on the shock movement. However, the method recovers well and the lengths of the ROM intervals increase. All these are in spite of the fact that, in this unsteady simulation, the shocks significantly move along the pressure and suction sides of the airfoil, as seen in figure 9. Note

that the pressure fields calculated with the ROM and the standard CFD solver are plot indistinguishable. In these plots, consistently with the CFD—computed value using a cell—centered finite volume scheme, a constant value of the pressure field is represented on each cell. Thus, the (apparent) artifacts appearing in the plots are solely due to the fact that the mesh is fairly coarse except in the vicinity of the airfoil.

As in the subsonic case, the heave motion has been shown to be a less demanding condition for the ROM efficiency and robustness.

Now, if the computational cost of the ROM were negligible compared to that of the CFD solver, the CPU acceleration factor defined in equation (27) would be proportional to the total timespan in which the ROM is used, which is calculated by adding up the lengths of the various time intervals in which the low-dimensional ROM is used. However, the computational cost of the ROM is not negligible. Instead, it increases as the number of retained modes increases. Thus, it is convenient to also take into account the required number of POD modes, which are considered in figure 10. As shown, the number of modes increases as either the forcing amplitude or frequency increases. Moreover, this increase is larger when increasing the forcing amplitude than when increasing the forcing frequency. However, in this latter case, the timespans in which the ROM is used are larger than in the former case (see figure 8), which compensates the larger number of retained POD modes.

In any event, further increase in the computational efficiency may result by a better selection of the the CFD update run duration, ΔT^{CFD} , as anticipated, and also by using generic mode libraries (Rapún *et al* 2015), which highly reduces the required value of the CFD update run duration, ΔT^{CFD} (thus, increasing the overall efficiency of the method) when simulating very complex dynamics in the Ginzburg–Landau equation.

6. Conclusions

A novel approach has been developed on the creation of ROMs for unsteady aerodynamics, keeping in mind aeroelastic calculations. The method is based on a already well established POD on the Fly methodology, which has been successfully applied in a variety of problems (e.g., the complex Ginzburg–Landau equation) that exhibit fairly complex dynamics, but are far less demanding from the reduced order modeling point of view than unsteady aerodynamics. This is because the Euler equations are hyperbolic and thus amenable of promoting shocks. Also, a moving mesh puts strong difficulties in the construction of ROMs. Therefore, it was not clear at all whether the POD on the Fly methodology would work for this more demanding problem. These difficulties have been solved in this paper.

After describing the method, it has been applied to the inviscid, compressible, unsteady flow around a rigidly moving two-dimensional airfoil, in both subsonic and transonic conditions. Moreover, the simulations have covered both pitch and heave motions and various values of the forcing amplitude and frequency, which covered both small and significant values of the reduced frequency. The results have shown a very good accuracy (plot indistinguishable results) and significant CPU acceleration factors (as defined in (27)). Thus, the results are very promising. In contrast to traditional preprocessed POD-based ROMs, the present ROM is capable of adapting to new dynamics that might be present along the simulation.

As an obvious extension, the new POD on the Fly ROM will be adapted to complex deformable geometries, which require deforming the computational mesh, instead of moving it as a rigid body. In addition, the integration with a structural solver (omitted in the present paper) in truly aeroelastic calculations will be addressed elsewhere.

Acknowledgments

This research was supported by the Spanish Ministry of Economy and Competitiveness and FEDER (EU), under Grants TRA2013-45808-R and MTM2013-47800-C2-2-P. The authors are indebted to two anonymous referees for some useful comments to an earlier version of the paper.

References

- Alonso D, Vega J and Velazquez A 2010 *AIAA J.* **48** 1946–58
- Alonso D, Velazquez A and Vega J M 2009 *Comput. Method Appl. Mech. Eng.* **198** 2683–91
- Astrid P, Weiland S, Willcox K and Backx T 2008 *IEEE T. Automat. Contr.* **53** 2237–51
- Balajewicz M, Dowell E and Noack B 2013 *J. Fluid Mech.* **729** 285–308
- Batina J 1985 Effects of airfoil shape, thickness, camber, and angle of attack on calculated transonic unsteady airloads *NASA Technical Memorandum* 86320
- Conner M, Tang D, Dowell E and Virgin L 1997 *J. Fluid. Struct.* **11** 89–109
- Courant R, Friedrichs K and Lewy H 1967 *IBM J. Res. Dev.* **11** 215–34
- Dowell E, Thomas J and Hall K 2001 *42nd AIAA ASME ASCE AHS ASC Structures, Structural Dynamics and Materials Conf. and Exhibit* (Seattle, WA)
- Garrick I 1933 *Determination of the Theoretical Pressure Distribution for Twenty Airfoils* NACA Technical Report 465
- Geuzaine C and Remacle J F 2009 *Int. J. Numer. Meth. Eng.* **79** 1309–31
- Gordon W and Hall C 1973 *Int. J. Numer. Meth. Eng.* **7** 461–77
- Greenshields C, Weller H, Gasparini L and Reese J 2010 *Int. J. Numer. Methods Fluids* **63** 1–21
- Heeg J, Chwalowski P, Florance J, Wieseman C, Schuster D and Perry R 2013 *51st AIAA Aerospace Sciences Meeting* (Grapevine, TX, US)
- Kurganov A and Tadmor E 2000 *J. Comput. Phys.* **160** 241–82
- Moran J 1984 *An Introduction to Theoretical and Computational Aerodynamics* (Courier Corporation) (New York: Dover)
- Moreno R, Taylor P and Newsom J 2012 *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf. (Honolulu, HI, United States)*
- Rapún M, Terragni F and Vega J 2015 *Int. J. Numer. Meth. Eng.* **104** 844–68
- Raveh D 2004 *J. Aircraft* **41** 620–32
- Rivera J, Dansberry B, Bennett R, Durham M and Silva W 1992 *NACA0012 Benchmark Model Experimental Flutter Results with Unsteady Pressure Distributions* National Aeronautics and Space Administration Langley Research Center NASA Technical Memorandum 107581
- Ryckelynck D 2009 *Int. J. Numer. Meth. Eng.* **77** 75–89
- Silva W and Bartels R 2004 *J. Fluid. Struct.* **19** 729–45
- Terragni F, Valero E and Vega J 2011 *SIAM J. Sci. Comput.* **33** 3538–61
- Terragni F and Vega J 2014 *SIAM J. Appl. Dyn. Syst.* **13** 330–65
- Theodorsen T and Center N L R 1935 *General Theory of Aerodynamic Instability and the Mechanism of Flutter* National Advisory Committee for Aeronautics NACA Technical Report 496