The Unsteady Continuous Adjoint Method Assisted by the Proper Generalized Decomposition Method

V. S. Papageorgiou, K. D. Samouchos and K. C. Giannakoglou*

Abstract In adjoint-based optimization for unsteady flows, the adjoint PDEs must be integrated backwards in time and, thus, the primal field solution should be available at each and every time-step. There are several ways to overcome the storage of the entire unsteady flow field which becomes prohibitive in large scale simulations. The most widely used technique is checkpointing that provides the adjoint solver with the exact primal field by storing the computed primal solution at a small number of time-steps and recomputing it for all other time-steps. Alternatively, approximations to the primal solution time-series can be built and used. One of them relies upon the use of the Proper Generalized Decomposition (PGD), as a means to approximate the time-series of the primal solution for use during the unsteady adjoint solver and this is where this paper is focusing on. The original contribution of this paper it that, apart from the standard PGD method, an incremental variant, running simultaneously with the time integration of unsteady primal equation(s) is proposed and tested. For the purpose of demonstration, three optimization problems based on different physical problems (unsteady heat conduction and unsteady flows around stationary and pitching isolated airfoils) are worked out by implementing the continuous adjoint method to both of them. The proposed incremental PGD technique is generic and can be used in any problem, to support either continuous or discrete unsteady adjoint.

1 Introduction

The numerical solution of the unsteady adjoint PDEs requires the storage or recomputation of the time-varying primal solution at each time-step. In the literature,

V. S. Papageorgiou, K. D. Samouchos, K. C. Giannakoglou
National Technical University of Athens (NTUA), School of Mechanical Engineering, Parallel CFD & Optimization Unit, Athens, Greece,
e-mail: ksamouchos@yahoo.com, kgianna@central.ntua.gr

strategies to overcome the full storage of the primal solution time-series have been proposed. The most frequently used is (binomial) checkpointing [5]. This may ensure the user-defined balance between storage of the primal solution at selected time-steps and recomputations. A viable alternative is to approximate the primal solution time-series through an (incremental) method with low storage requirements, employed during the solution of the primal PDEs, which are integrated forwards in time. Approximation methods can be evaluated in terms of the accuracy of the reconstructed primal solution and its effect on the computed sensitivity derivatives, as well their computational cost. Approximation methods have the advantage of avoiding (even partial) recomputations of the unsteady primal solution, as required by methods such as checkpointing. Among them, linear interpolation, cubic-splines, Fourier series or data compression techniques such as the Singular Value Decomposition (SVD) [2, 10] should be reported.

In this paper, the PGD method [3, 1, 6] is used to reconstruct the primal field by reducing memory storage. The main idea is to represent a multi-dimensional (in space and time) field as the sum of products of 1D functions; for an unsteady 2D scalar field of u, for instance, one may write

$$u(x,y,t) \cong \sum_{\mu=1}^{M} \phi^{\mu}(x)\theta^{\mu}(y)\tau^{\mu}(t) \tag{1}$$

Assuming that a small number M of modes is enough, a noticeable gain in memory usage is expected since scalar modes ϕ^{μ}, θ^{μ} and τ^{μ} ($\mu = 1,\cdots,M$) are stored instead of the entire $u(x,y,t)$ field.

In an unsteady simulation, if the whole time-series of the solution $u(x,y,t)$ must be available before processing them by the PGD, no gain in storage requirements is expected. For this reason, an alternative method is proposed, in which once the instantaneous primal solution becomes available at each time-step, the already computed modes are incrementally updated. This will be referred to as **incremental PGD (iPGD)**.

In this paper, the programmed PGD (or iPGD) library is used to reconstruct the solution of an unsteady heat conduction and two unsteady inviscid flow problems within optimization workflows supported by the continuous adjoint method [7]. For the heat conduction problem, a 2D structured mesh is used and the spatial part of the space-time decomposition is performed in the transformed domain. In the unsteady flow problem, an immersed boundary approach (in specific, the cut-cell method [4, 9]) is used. The adaptive mesh used by the cut-cell method requires extra treatment that will be made clear in section 5.

2 Reconstructing Already Computed Fields by PGD

Consider a 2D time-dependent field $u = u(x,y,t)$, previously computed by any PDE solver on a standard structured mesh. In the PGD framework, this solution can be

approximated by the sum of a relatively small number (M) of 1D functional products (eq. 1). All these modes are built in M successive steps. At the m^{th} step ($m \leq M$), the corresponding 1D functions are computed so as to minimize the representation error, which is given in discrete form by:

$$E_m = \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{j=1}^{J} \left[\sum_{\mu=1}^{m} \phi_i^\mu \theta_j^\mu \tau_k^\mu - u_{i,j,k} \right]^2 \tag{2}$$

The problem of defining the modes is non-linear, so it has to be solved iteratively within each of the M steps of the successive enrichment by means of an alternating direction scheme. For the m^{th} modes ($1 \leq m \leq M$), ϕ^m is computed first, considering θ^m and τ^m to be known from the previous iterations or their initialization and so forth. For instance to compute ϕ^m, eq. 1, truncated by keeping only the first m terms, is multiplied by τ^m and θ^m and integrated along t and y. Since all the functions of y and t are known, the 2D integrals can be computed and the final equations for updating the modes are

$$\begin{aligned}
\phi^m &= \frac{\int_y \int_t u\theta^m t^m \mathrm{d}t\,\mathrm{d}y - \sum_{\mu=1}^{m-1} \phi^\mu \int_y \int_t \theta^m \theta^\mu \tau^m \tau^\mu \,\mathrm{d}t\,\mathrm{d}y}{\int_y \int_t (\theta^m)^2 (\tau^m)^2 \mathrm{d}t\,\mathrm{d}y} \\
\theta^n &= \frac{\int_x \int_t u\phi^n t^n \mathrm{d}t\,\mathrm{d}x - \sum_{i=1}^{n-1} \theta^i \int_x \int_t \phi^n \phi^i \tau^n \tau^i \,\mathrm{d}t\,\mathrm{d}x}{\int_x \int_t (\phi^n)^2 (\tau^n)^2 \mathrm{d}t\,\mathrm{d}x} \\
\tau^n &= \frac{\int_y \int_x u\phi^n \theta^n \mathrm{d}x\,\mathrm{d}y - \sum_{i=1}^{n-1} \tau^i \int_y \int_x \phi^n \phi^i \theta^n \theta^i \,\mathrm{d}x\,\mathrm{d}y}{\int_y \int_x (\phi^n)^2 (\theta^n)^2 \mathrm{d}x\,\mathrm{d}y}
\end{aligned} \tag{3}$$

The three above equations for the m^{th} modes are used iteratively until an appropriate convergence criterion be met, before proceeding to the computation of the next modes.

Through differentiation of eq. 2, it can be proved that the modes computed by eqs. 3 minimize E_m. Taking this into consideration, the incremental variant of PGD (iPGD) can be formulated. This formulation and implementation of iPGD in unsteady adjoint is the key originality of this paper.

3 Incremental PGD Method

In order to approximate a flow field $u(x,y,t)$ using the method developed in the previous section, the whole time-series should have been computed and stored beforehand. This storage should definitely be avoided. This section presents a new

method (iPGD) which overcomes this drawback. The concept of the iPGD method is that, the field reconstruction is gradually performed. During the integration of the primal PDEs, the solution field at each new time-step is used to enrich the previously computed modes.

Consider the same 2D time-dependent field $u = u(x,y,t)$ which, hereafter, will be in discrete form as $u_{i,j,k}$. The field approximation is still given by eq. 1, but modes should incrementally be updated at each time-step. Equations for updating the modes are extracted by minimizing an error function similar to eq. 2. Since only the current (time-index $k = K+1$) solution field is available, the error function must be decomposed as:

$$\begin{aligned} E_m = & \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m}\phi_i^{\mu}\,\theta_j^{\mu}\,\tau_{K+1}^{\mu} - u_{i,j,K+1}\right]^2 \\ & + \frac{w}{2}\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m}\left(\phi_i^{\mu}\,\theta_j^{\mu}\,\tau_k^{\mu} - \widetilde{\phi}_i^{\mu}\,\widetilde{\theta}_j^{\mu}\,\widetilde{\tau}_k^{\mu}\right)\right]^2 \end{aligned} \tag{4}$$

where the first term on the r.h.s. corresponds to the approximation error at the current time-step, whereas the second one to the overall error for all the previous time-steps, which have already been processed through the iPGD and yielded modes $\widetilde{\phi}_i^{\mu}$, $\widetilde{\theta}_j^{\mu}$, $\widetilde{\tau}_k^{\mu}$. The contribution to the error is weighted by w which is user-defined. At each time-step, modes $(\phi_i^m, \theta_j^m, \tau_k^m)$ are updated and new values τ_{K+1}^m are appended. The unknown quantities are calculated by setting the derivatives of the error against zero, getting

$$\phi_i^m = Q_{1x}^i / Q_{2x}^i, \qquad i = 1,..,I \tag{5a}$$

$$\theta_j^m = Q_{1y}^j / Q_{2y}^j, \qquad j = 1,..,J \tag{5b}$$

$$\tau_k^m = Q_{1t}^k / Q_{2t}^k, \qquad k = 1,..,K \tag{5c}$$

$$\tau_{K+1}^m = Q_{1T}^{K+1} / Q_{2t}^{K+1}, \tag{5d}$$

where

$$\begin{aligned} Q_{1x}^i = & \tau_{K+1}^m \sum_{j=1}^{J}\theta_j^m\,u_{i,j,K+1} - \tau_{K+1}^m \sum_{\mu=1}^{m-1}\left[\left(\sum_{j=1}^{J}(\theta_j^{\mu}\,\theta_j^m)\right)\phi_i^{\mu}\,\tau_{K+1}^{\mu}\right] \\ & + w\widetilde{\phi}_i^m \sum_{k=1}^{K}\sum_{j=1}^{J}\widetilde{\theta}_j^m\,\theta_j^m\,\widetilde{\tau}_k^m\,\tau_k^m - w\sum_{k=1}^{K}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m-1}\left(\phi_i^{\mu}\,\theta_j^{\mu}\,\tau_k^{\mu} - \widetilde{\phi}_i^{\mu}\,\widetilde{\theta}_j^{\mu}\,\widetilde{\tau}_k^{\mu}\right)\theta_j^m\,\tau_k^m\right] \\ Q_{2x}^i = & (\tau_{K+1}^m)^2 \sum_{j=1}^{J}(\theta_j^m)^2 + w\sum_{k=1}^{K}\sum_{j=1}^{J}(\theta_j^m)^2\,(\tau_k^m)^2 \end{aligned}$$

$$Q_{1y}^{j} = \tau_{K+1}^{m} \sum_{i=1}^{I} \phi_i^m \, u_{i,j,K+1} - \tau_{K+1}^{m} \sum_{\mu=1}^{m-1} \left[\left(\sum_{i=1}^{I} (\phi_i^{\mu} \, \phi_i^m) \right) \theta_j^{\mu} \, \tau_{K+1}^{\mu} \right]$$
$$+ w \widetilde{\theta}_j^m \sum_{k=1}^{K} \sum_{i=1}^{I} \widetilde{\phi}_i^m \, \phi_i^m \, \widetilde{\tau}_k^m \, \tau_k^m - w \sum_{k=1}^{K} \sum_{i=1}^{I} \left[\sum_{\mu=1}^{m-1} \left(\phi_i^{\mu} \, \theta_j^{\mu} \, \tau_k^{\mu} - \widetilde{\phi}_i^{\mu} \, \widetilde{\theta}_j^{\mu} \, \widetilde{\tau}_k^{\mu} \right) \phi_i^m \, \tau_k^m \right]$$
$$Q_{2y}^{j} = (\tau_{K+1}^m)^2 \sum_{i=1}^{I} (\phi_i^m)^2 + w \sum_{k=1}^{K} \sum_{i=1}^{I} (\phi_i^m)^2 \, (\tau_k^m)^2$$

$$Q_{1t}^{k} = \widetilde{\tau}_k^m \sum_{j=1}^{J} \sum_{i=1}^{I} \widetilde{\phi}_i^m \, \phi_i^m \widetilde{\theta}_j^m \, \theta_j^m - \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{\mu=1}^{m-1} \left(\phi_i^{\mu} \, \theta_j^{\mu} \, \tau_k^{\mu} - \widetilde{\phi}_i^{\mu} \, \widetilde{\theta}_j^{\mu} \, \widetilde{\tau}_k^{\mu} \right) \phi_i^m \, \theta_j^m$$
$$Q_{1T}^{K+1} = \tau_{K+1}^m = \sum_{i=1}^{I} \sum_{j=1}^{J} \phi_i^m \, \theta_J^m u_{i,j,K+1} - \sum_{\mu=1}^{m-1} \left[\tau_{K+1}^{\mu} \sum_{i=1}^{I} \sum_{j=1}^{J} \phi_i^{\mu} \, \phi_i^m \, \theta_j^m \, \theta_j^{\mu} \right]$$
$$Q_{2t}^{k} = \sum_{i=1}^{I} \sum_{j=1}^{J} (\phi_i^m)^2 (\theta_j^m)^2$$

Eqs. 5 are coupled and must be solved iteratively through the following algorithm:

- **Step 1:** Initialize ϕ, θ and τ for u(x,y,t) at the initial time instant, i.e. for $k=1$. This is, practically, equivalent to the PGD of a known 2D spatial field. Set $k=2$.
- **Step 2:** Compute all ϕ^m, θ^m and τ^m, $m=1,..,M$, using eqs. 5a to 5c.
- **Step 3:** Compute τ_{k+1}^m, using eq. 5d.
- **Step 4:** Set $k=k+1$. Return to step 2.

It is obvious that the above algorithm approximates the various instantaneous spatial fields with different error as it proceeds from one time-step to the next. However, this cannot be avoided since we are using a single error function E which amalgamates all time instants; recall that the final error to be minimized in the one corresponding to all time-steps.

4 Application 1: Optimization Based on the Unsteady Heat Conduction Equation

The first application is dealing with the optimization of the temperature (T) profile along the left–most straight boundary S_c of a 2D domain Ω (fig. 2). A 100×80 structured mesh is used. Along the remaining boundaries of Ω, fixed Dirichlet conditions are imposed on T. Temperature T (in Kelvin) along (S_c) is given by

$$T(\zeta,t) = \widetilde{T}(\zeta) + 20\,\zeta(1-\zeta)\sin\left(\frac{2\pi t}{T_a}\right) \tag{6}$$

where its "steady" part is

$$\begin{aligned}\widetilde{T}(\zeta) =&\beta_1(5\zeta-20\zeta^2+30\zeta^3-20\zeta^4+5\zeta^5)+\beta_2(10\zeta^2-30\zeta^3+30\zeta^4-10\zeta^5)\\&+\beta_3(10\zeta^3-20\zeta^4+10\zeta^5)+\beta_4(5\zeta^4-5\zeta^5)\\&+T_1(-5\zeta+10\zeta^2-10\zeta^3+5\zeta^4-\zeta^5)+T_2\zeta^5\end{aligned}$$

This rather complicated expression results from a Bézier-based parameterization of the unknown temperature profile. In the above formulas, $0 \leq \zeta \leq 1$ is the non-dimensional distance of any node on S_c measured from the bottom-left corner of Ω, and β_1, β_2, β_4 are given by

$$\beta_1 = \frac{1}{(b_1-b_2)^2+3} - \frac{1}{(b_3+2)^2+5}, \quad \beta_2 = \frac{1}{(b_2+b_3)^2+4} - \frac{1}{(b_3-1)^2+1},$$
$$\beta_4 = \frac{1}{(b_3+1)^2+2} - \frac{1}{(b_1-b_3)^2+5}$$

where $\mathrm{b_q}(\mathrm{q}=1,2,3)$ are the three optimization variables. The extra variable β_3 depends on the other three, its role being to ensure that the mean temperature on the boundary S_c remains constant and equal to $450K$. This leads to the constraint $\beta_3 = 1800-\beta_1-\beta_2-\beta_4$ to be met. The T profile in eq. 6 changes periodically in time, with period $T_a = 800\,s$. The unsteady heat conduction equation in the transformed (ξ,η) domain is

$$R = \rho C_p \frac{\partial T}{\partial t} - \frac{1}{J}\frac{\partial}{\partial \xi^i}\left(kJg^{ij}\frac{\partial T}{\partial \xi^j}\right) = 0 \tag{7}$$

where g^{ij} is the contravariant metrics and J the Jacobian of the transformation. The approximated temperature T, eq. 1, in the (ξ,η) domain, takes the form $T(\xi,\eta,t) \cong \sum_{\mu=1}^{M} \phi^\mu(\xi)\theta^\mu(\eta)\tau^\mu(t)$. In eq. 7, k=$k(T)$ is the thermal diffusivity, ρ the material density and C_p the heat capacity. Assuming that the material is aluminium, $k(T) = 0.0002213\,T^2 - 0.09592\,T + 211.5[W/mK]$ (T in Kelvin), $\rho = 2.7kg/m^3$ and $C_p = 0.910kJ/kgK$. The length of boundary S_c is equal to 2m. The objective is to minimize the area and time with $T > T_{crit} = 400K$. Since this objective is not differentiable, it was replaced by

$$F = \frac{1}{\Omega T_a}\int_{t^*}^{t^*+T_a}\int_{\Omega}\left(1-\frac{1}{1+e^{k_2(T-T_{crit})+k_1}}\right)(aT+b)\mathrm{d}\Omega\mathrm{d}t \tag{8}$$

where k_1=3, k_2=$k_1/(T_{safe}-T_{crit})$, a=$3/T_{crit}$, b=$1-aT_{safe}$ and T_{safe}=$450K > T_{crit}$ (a user defined relaxing threshold temperature). The time integral is extended over T_a with the lower limit of integration being t^*, at which the periodic solution has been established. The development of the continuous unsteady adjoint method is carried out in the standard way, leading to the field adjoint equation

$$-\rho C_p \frac{\partial \Psi}{\partial t} - \frac{1}{J}\frac{\partial}{\partial \xi^i}\left(kJg^{ij}\frac{\partial \Psi}{\partial \xi^j}\right) + g^{ij}\frac{\partial k}{\partial T}\frac{\partial T}{\partial \xi^i}\frac{\partial \Psi}{\partial \xi^j} = f \tag{9}$$

where Ψ is the adjoint field and f results from the differentiation of the objective function. Eq. 9 is solved by imposing time periodic conditions $\Psi|_{t=t^*} = \Psi|_{t=t^*+T_a}$ whereas along the whole boundary S, $\Psi|_{S,\forall t} = 0$. Finally, the sensitivity derivatives of F w.r.t. the design variables b_q are given by

$$\frac{\delta F}{\delta b_q} = \int_{t^*}^{t^*+T_a} \int_{S_c} k \frac{\partial \Psi}{\partial n} \frac{\delta T}{\delta b_q} \mathrm{dS}\,\mathrm{dt}, \quad q = 1,2,3 \tag{10}$$

where $\frac{\delta T}{\delta b_q}|_{S_c} = \frac{\delta T}{\delta \beta_k} \frac{\delta \beta_k}{\delta b_q}$ $(k = 1,2,3,4)$. The optimization was based on the steepest descent method. The case ran for about 7 periods in order to ensure that the temperature field becomes periodic; then, the next period was processed by the proposed iPGD algorithm and stored for use during the integration of the unsteady adjoint equation in reverse time. 100 time-steps per period T_a were used. For the purpose

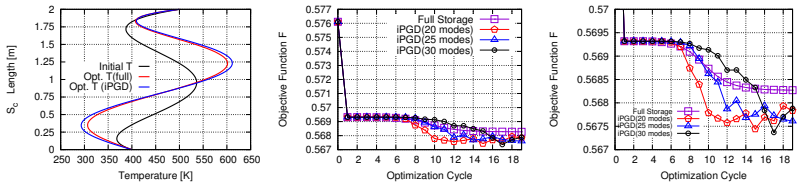
Table 1: Application 1: Sensitivity derivatives corresponding to the design variable data-set $b_1 = 0.05$, $b_2 = b_3 = 0.01$ for full storage of the primal field PGD and iPGD-based compression with various number of modes. Non-incremental a posteriori PGD (denoted by PGD $M = 20$) is also included in the plot.

	$\delta F/\delta b_1$	$\delta F/\delta b_2$	$\delta F/\delta b_3$
Full Storage	-1.3715605463E-4	4.5884641458E-6	1.359568712E-2
PGD $M=20$	-1.3749363010E-4	4.6796390425E-6	1.362148509E-2
iPGD $M=30$	-1.4020728461E-4	4.3113618801E-6	1.391765592E-2
iPGD $M=25$	-1.4067134121E-4	6.4219368409E-6	1.381274791E-2
iPGD $M=20$	-1.4357302527E-4	6.7785627830E-6	1.406576690E-2

of comparison, the same optimization was repeated twice: (a) with full storage of the results of the primal equations and (b) using the iPGD method for storing just the $\phi(\xi)$, $\theta(\eta)$ and $\tau(t)$ modes with various values of M. Of course, the full storage could have been replaced by checkpointing with identical sensitivity derivatives. In all cases, $w = 50$.

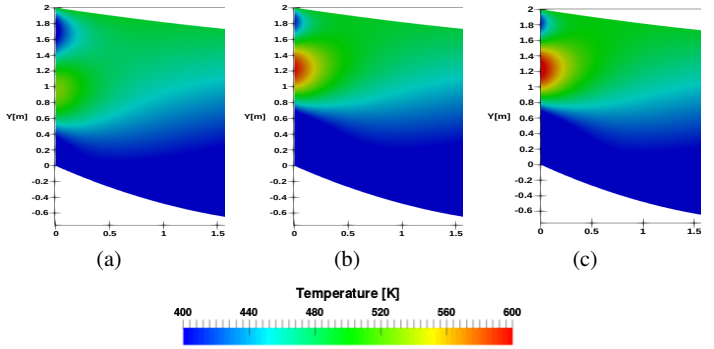
The sensitivity derivatives computed with fully stored and iPGD'ed primal fields are shown in table 1. Reasonable deviations due to the approximation were expected but these were harmless for the optimization itself. In fig. 2, initial and optimized time-averaged temperature fields are shown. The small overheated spot formed closed to S_c is not contradictory since a greater part of the area is kept at lower temperature and this yields lower values of F, eq. 8. In fig. 1(a), the corresponding $\widetilde{T}$ distributions are shown. The optimization follows a slightly different path in each case (fig. 1(b)), as the adjoint solver relies upon differently approximated temperature fields. However, all cases converge to close data-sets of the design variables by equally reducing the objective function. The memory needed, even with 30 modes, is about 32.5 % less than the full storage of the primal unsteady field. In terms of

memory usage , we should make clear that the comparison is made against full storage, instead of checkpointing, to avoid also accounting for the extra computational cost of the latter due to the partial recomputation of the primal field.



(a) (b) (c)

Fig. 1: Application 1: (a) Initial and optimal temperature profiles $\widetilde{T}(\zeta)$ along S_c for full storage of the unsteady temperature field and using iPGD with $M = 30$. (b) Convergence of the objective function. (c) Blow-up view of (b).



(a) (b) (c)

Fig. 2: Application 1: Time-averaged spatial T distribution on the left–most part of the computational domain for: (a) the initial $\widetilde{T}$ profile (eq. 7), with b_1=b_2=b_3=0.01, (b) the optimal one (b_1=0.6903, b_2=-0.3743, b_3=-13.7670) as computed with full storage of T and (c) the optimal solution (b_1=0.8287, b_2=-0.5367, b_3=-14.0181) computed using the iPGD with $M = 30$. Either optimization terminated after 20 cycles, using the same step of steepest descent.

5 Applications 2 & 3: Gradient Computation for the Unsteady Euler Equations with the Cut-Cell Method

Herein, the PGD method is implemented to compute the objective function gradient to be used during the shape optimization of an isolated airfoil parameterized using Bézier curves, where the design variables (b_q) are their control points' coordinates.

Gradient computation for a stationary airfoil, in which unsteadiness is introduced by the time-varying far-field flow angle and a pitching airfoil are demonstrated. In either case, the governing PDEs are the unsteady 2D flow Euler equations:

$$R_i = \frac{\partial U_i}{\partial t} + \frac{\partial f_{ij}}{\partial x_j} = 0, \qquad i = 1,4,\ x_j = x,y \tag{11}$$

where $\overrightarrow{U} = \left[\rho\ \ \rho u\ \ \rho v\ \ E\right]^T$ is the vector of conservative variables and f_{ij} are the inviscid fluxes in the Cartesian directions, ρ the fluid density, u and v the Cartesian velocity components, p the static pressure and E the total energy per unit volume. The objective function is the time-averaged lift over a single period, $F = \frac{1}{T_a}\int_0^{T_a}\int_{S_w} p n_k r_k dS dt$, where n_k, r_k are the components of the unit vectors normal to the airfoil surface and the freestream velocity, respectively.

The required derivatives of F w.r.t. b_q are computed by the continuous adjoint method. The adjoint equations are [9]

$$-\frac{\partial \Psi_i}{\partial t} - A_{jik}\frac{\partial \Psi_j}{\partial x_k} = 0 \qquad i,j = 1,4,\ x_k = x,y \tag{12}$$

where $A_{ijk} = \frac{\partial f_{ik}}{\partial U_j}$ and Ψ_i are the adjoint variable fields. The adjoint boundary conditions are omitted in the interest of space.

When the adjoint solution becomes periodic, the sensitivity derivatives are

$$\begin{aligned}\frac{\delta F}{\delta b_q} = {} & \frac{1}{T_a}\int_0^{T_a}\int_{S_w} p\frac{\delta(n_k r_k dS)}{\delta b_q}dt + \int_0^{T_a}\int_{S_w}\left(\Psi_{k+1}p - \Psi_i f_{ik}\right)\frac{\delta n_k}{\delta b_q}dSdt \\ & + \int_0^{T_a}\int_{S_w}\Psi_i\left(\frac{\partial f_{il}}{\partial x_l}\frac{\delta x_k}{\delta b_q} - \frac{\partial f_{ik}}{\partial x_l}\frac{\delta x_l}{\delta b_q}\right)n_k dSdt \\ & + \int_0^{T_a}\int_{S_w}\Psi_i\frac{\partial U_i}{\partial x_l}u_n^w\frac{\delta x_l}{\delta b_q}dSdt + \int_0^{T_a}\int_{S_w}\left(\Psi_i U_i + p\Psi_4\right)\frac{\delta u_n^w}{\delta b_q}dSdt\end{aligned}$$

The last two integrals are related to the normal velocity of the solid wall (u_n^w) and vanish for the stationary airfoil.

The flow solution is obtained using the cut-cell method [4, 9], according to which the flow simulation takes place on a Cartesian grid, covering both the fluid and solid regions. For higher accuracy, cells cut by the body contour along with their closest neighbours are subdivided into smaller ones (fig. 3). Consequently, the (i,j) data-structure, being a prerequisite for applying the PGD, is no more valid. It is beyond the scope of this paper to compare the accuracy of the cut-cell method with that of CFD on body-fitted grids. Here, we are exclusively interested in evaluating the adequacy of the (i)PGD approximations.

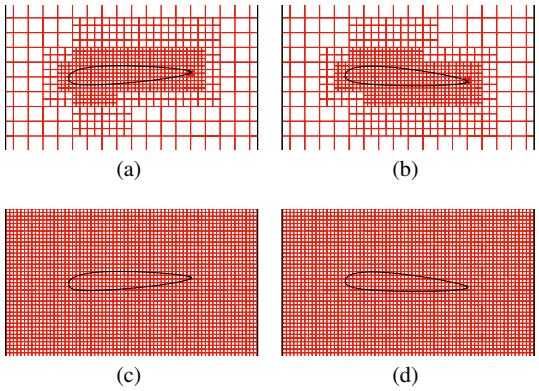


Fig. 3: Application 2.2: Adapted meshes, (a) and (b), at the two extreme positions of the pitching airfoil. Corresponding RR meshes, (c) and (d), used for the iPGD.

The integration of the flow equations is based on a cell-centered finite-volume method with second-order accuracy in both space and time; fluxes are computed using the Roe scheme [8]. Special treatment for cells cut by the airfoil is needed in order to satisfy the conservation laws near solid boundaries with the required accuracy. An extra difficulty appears, if the body is moving in time. In such a case, the mesh is continuously adapted to the new position of the body, as follows. Firstly, the mesh undergoes a coarsening process, where all cells close to the solid wall are amalgamated with bigger neighbouring cells. Then, the body moves to its new position and, starting from the already coarsened mesh, cells are split until a certain level of refinement be met. Meshes are shown in fig. 3.

By definition, the PGD (or iPGD) can be applied only to structured meshes. The lack of structure of the mesh used in the cut-cell method is overcome through a Reference/Regular (RR) mesh. After solving the unsteady Euler equations with the cut-cell method, the corresponding flow field is transferred to the RR mesh (fig. 3c & 3d), which is as fine as the smallest cell of the cut-cell mesh (fig. 3a & 3b). After that, the iPGD algorithm is implemented to the RR mesh as explained in section 2. The required transition from the adapted cut-cell meshes to the RR one, must be quick and accurate. Using an efficient algorithm based on quad-tree data structure, the correspondence between the cells of the two meshes is easily accomplished. Each cell of the RR mesh takes on the flow variables of the cut-cell mesh cell which is part of.

5.1 Application 2: Stationary Airfoil

The unsteady Euler equations are solved around a stationary airfoil. The far-field flow conditions are $M_\infty = 0.3$ and $a_\infty = Asin(\omega t)\ [deg]$ with amplitude $A = 3^o$ and period $T_a = \frac{2\pi}{\omega} = 0.015s$. The cut-cell mesh used for the simulation consists of 10500 cells; a constant time-step equal to $T_\alpha/20$ is used.

While integrating the flow equations from the previous to the current time-step, each flow field is processed by the iPGD with $M=10$ and $w=1000$. Fig. 4 illustrates the pressure fields, at two instants corresponding to the max. and min. angle of attack, as computed by the cut-cell method. The comparison between approximated and exact fields is satisfactory. After the time-integration of the flow equations, the

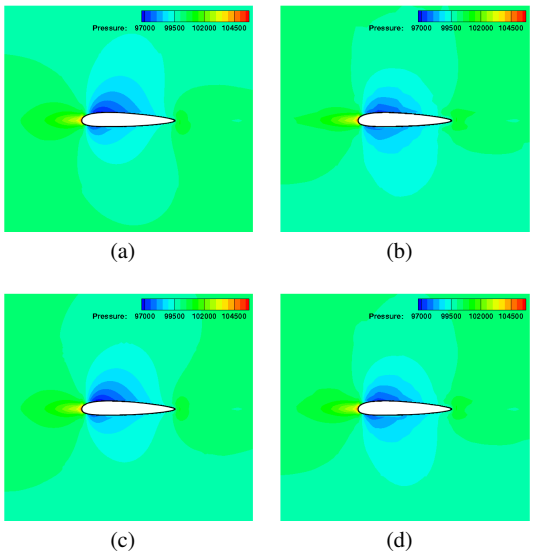
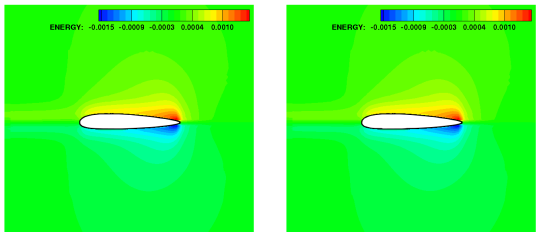


Fig. 4: Application 2: Instantaneous pressure fields for $a_\infty = -3^o$ (a) and $a_\infty = +3^o$ (c) are quite close each other with the corresponding fields (b) and (d) computed by the iPGD method.

iPGD'ed fields are made available to the adjoint software. A comparison of the computed adjoint energy field based on the exact and reconstructed flow solutions is presented in fig. 5.

Having made both the primal and the adjoint flows available, the sensitivity derivatives of F w.r.t. b_q are computed. Fig. 6 shows the effect on approximating the primal solution through the iPGD method on the accuracy of sensitivity derivatives. In the same figure, sensitivities computed by the posteriori PGD (i.e not incremental) compression of the primal unsteady solution (fully stored just for this purpose) are also shown. The extra deviation due to the incremental algorithm is much smaller than the total difference between the a posteriori PGD and the reference (from full storage) values of the derivatives, demonstrating the capabilities of the proposed incremental algorithm. Note that the only reason we additionally ran the adjoint based on the a posteriori PGD'ed primal solutions is for obtaining a good indication of the best accuracy we could ideally get by the iPGD.



(a) (b)

Fig. 5: Application 2: Instantaneous adjoint energy fields for $a_\infty = -3^o$ based on the flow solution by the cut-cell method (a) and the iPGD-based approximation to the unsteady flow solution (b).

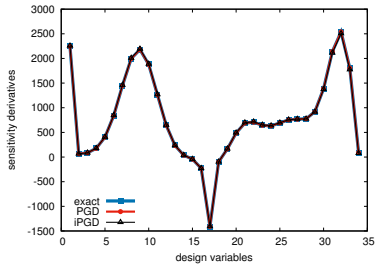


Fig. 6: Application 2: Comparison of sensitivity derivatives computed by the adjoint using (a) full storage, (b) the a posteriori PGD'ed primal solution and (c) the iPGD'ed one.

5.2 Application 3: Pitching Airfoil

In the pitching airfoil case, since the mesh is changing in time, its data structure should have also been stored at each and every time-step over and above to the unsteady flow solution. However, the special structure of the Cartesian meshes allows minimum data storage, overcoming the need of compressing also the time-changing mesh. The airfoil is oscillating around the point at chord/4 with position angle that is a sinusoidal function with amplitude (3^o) and period equal to $T_a = 0.015$ s, split into 20 time-steps. The average number of cells used at all time-steps is about 7000. The size of the RR mesh is 512×512. The far-field Mach number is equal to 0.3.

The flow field is compressed via the iPGD algorithm using $M = 10$ modes. In fig. 7 two fields are shown at the extreme instants of the period for the exact and the reconstructed fields.

The impact of the compressed primal fields on the adjoint solution was examined by solving the adjoint equations twice with full storage and the iPGD'ed primal data. Results of the adjoint solver are shown in fig. 8. For the two aforementioned cases, the sensitivity derivatives are computed, fig. 9. Moreover, two extra curves for 20

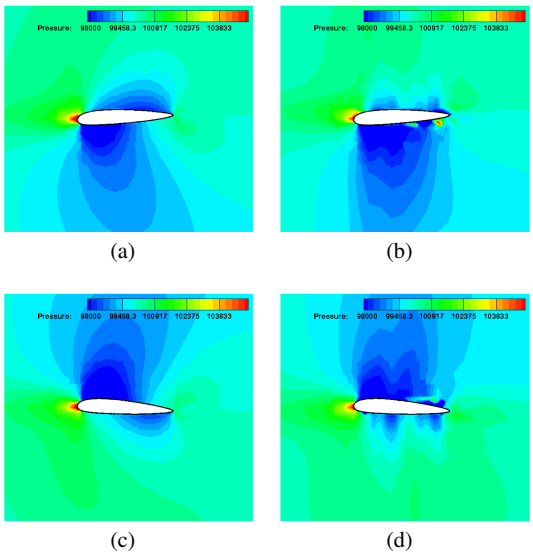


Fig. 7: Application 3: Instantaneous pressure fields at the lowermost (a) and the uppermost (c) positions of the airfoil motion are almost the same with the corresponding fields (b and d) computed by the iPGD method.

and 30 modes are shown. As the number of modes increases, the primal and adjoint fields match each other much better and the error in the computed derivatives diminishes. Theoretically, by increasing the number of modes, the sensitivity derivatives should tend to the "exact ones" (the one from full storage). However, in practice they seem to stagnate since, in order to save computational cost we end up with a reasonably low number of modes. The saving in memory by the usage of the proposed iPGD algorithm is remarkable. The full storage of the unsteady flow field needs an average of $7000\times 20 = 140000$ values to be saved in memory whereas, even with $M = 30$, this number drops to $30(512+512+20) = 31320$ with the iPGD. The reason for refraining to compare with checkpointing is exposed at the end of section 4.

6 Conclusions

The use of the PGD within adjoint-based optimization, for time-varying problems was presented. The role of PGD is to approximate the time-series of the solution to the primal PDEs to support the integration of the adjoint equations, backwards in time. For the first time in the relevant literature, at least to the authors' knowledge, an incremental PGD algorithm is proposed. Its distinguishing feature is that there is no need to store the time-series of the primal solution before processing them with

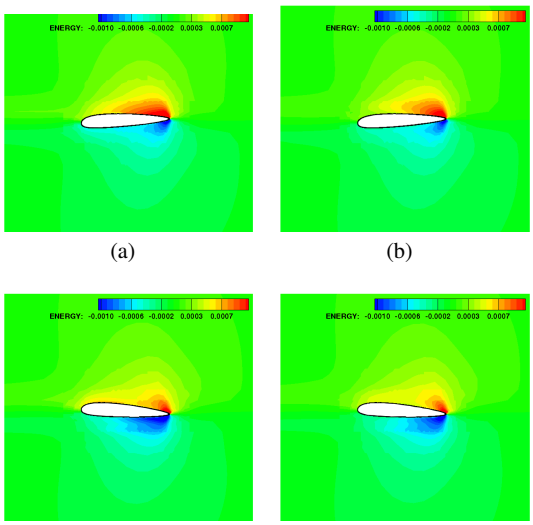


Fig. 8: Application 3: Instantaneous adjoint energy fields in the lowermost (a) and the uppermost (c) position of its motion are almost the same with the corresponding fields (b and d) computed by the iPGD.

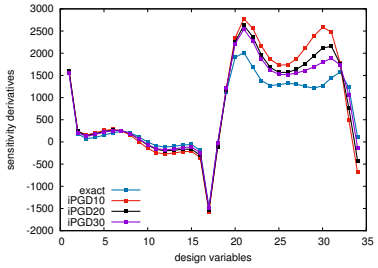


Fig. 9: Application 3: Comparison of the sensitivity derivatives computed using full storage and the iPGD'ed primal solution with $M = 10, 20, 30$. The fact that the sensitivity of the last design variable is approximated with the wrong sign is minor since this corresponds to the trailing edge which remains still.

the PGD. Instead, all modes are updated upon completion of a single step of the time–integration of the primal PDEs; more precisely, all spatial modes are updated to account for the new instantaneous primal field and a new element is appended to each one of the growing temporal modes. With the iPGD method, storage requirements are much lower compared to the full storage; no comparison with the checkpointing technique has been made since, the extra computation cost of partial recomputations of the primal solution of checkpointing should also be accounted for

and compared with the extra cost of running the iPGD algorithm. We refrained from doing so since work on the acceleration of the iPGD method is in progress. The mathematical formulation of the new iPGD method is provided. Unsteady adjoint runs supported by the iPGD were demonstrated in unsteady heat and aerodynamic optimization problems and proved to offer a great economy in storage requirements. Even though all applications presented in this paper relied upon the continuous adjoint, the iPGD can also be used with discrete adjoint, as it is not related to the way the adjoint problem is formulated and solved. Another contribution of this work is the use of the iPGD with the cut-cell method, in which case the CFD meshes dynamically adapt to the shape boundaries. However, even in this case, it suffices to use a Reference/Regular mesh and appropriate interpolation schemes in order to be able to apply the iPGD as with standard meshes.

References

1. A. Ammar, F. Chinesta, E. Cueto, and M. Doblar. Proper generalized decomposition of time-multiscale models. *International Journal for Numerical Methods in Engineering*, 90(5):569–596, 2012.
2. L. Balzano and S. Wright. On grouse and incremental SVD. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop*, St. Martin, Dec 2013.
3. F. Chinesta, R. Keunings, and A. Leygue. *The Proper Generalized Decomposition for Advanced Numerical Simulations, A Primer*. Springer International Publishing, Nantes, France, 2014.
4. D. Clarke, H. Hassan, and M. Salas. Euler calculations for multielement airfoils using cartesian grids. *AIAA Journal*, 24(3):353 – 358, 1986.
5. A. Griewank and A. Walther. Algorithm 799: Revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. on Math. Software (TOMS)*, 26(1):19–45, 2000.
6. P. Ladevèze. *PGD in Linear and Nonlinear Computational Solid Mechanics*. Springer, Vienna, 2014.
7. D. Papadimitriou and K. Giannakoglou. A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows.
8. P. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357 – 372, 1981.
9. K. Samouchos, S. Katsanoulis, and K.C. Giannakoglou. Unsteady adjoint to the cut-cell method using mesh adaptation on GPUs. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, Crete island, Greece, June 5-10 2016.
10. C. Vezyris, I. Kavvadias, E. Papoutsis-Kiachagias, and K. Giannakoglou. Unsteady continuous adjoint method using POD for jet-based flow control. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, Crete island, Greece, June 5-10 2016.