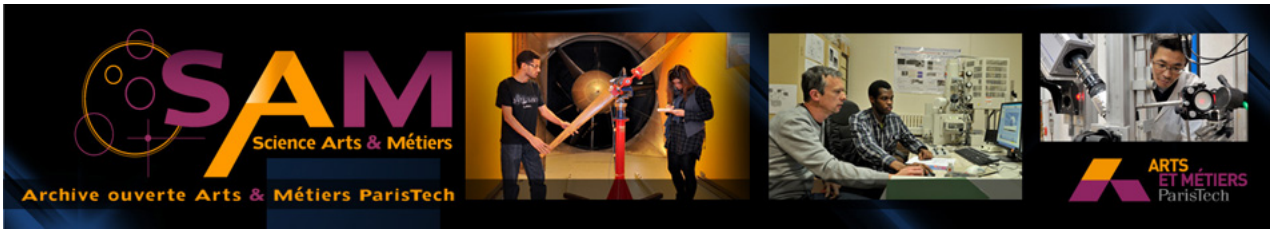
Proper general decomposition (PGD) for the resolution of Navier-Stokes equations

Antoine Dumon, Cyrille Allery, Amine Ammar



Proper general decomposition (PGD) for the resolution of Navier–Stokes equations

A. Dumon [a,*], C. Allery [a], A. Ammar [b]

[a] LEPTIAB, Pôle Sciences et Technologie, Avenue Michel Crepeau, 17042 La Rochelle Cedex 1, France
[b] Arts et Metiers ParisTech, 2 Bvd du Ronceray, 49035 Angers Cedex 01, France

A B S T R A C T

In this work, the PGD method will be considered for solving some problems of fluid mechanics by looking for the solution as a sum of tensor product functions. In the first stage, the equations of Stokes and Burgers will be solved. Then, we will solve the Navier–Stokes problem in the case of the lid-driven cavity for different Reynolds numbers (Re = 100, 1000 and 10,000). Finally, the PGD method will be compared to the standard resolution technique, both in terms of CPU time and accuracy.

Keywords:
Reduced order model
Proper generalized decomposition
Lid-driven cavity
Navier–Stokes equations

1. Introduction

The numerical simulation of complex fluid flows gives rise to very large systems that cannot be easily solved numerically. This situation is not convenient for optimization problems where multiple solutions are usually required, or for feed-back control problems where realtime solutions are requested. Consequently, innovative alternative methods have been developed.

In this context, various reduced-order models (ROM) have been developed in order to decrease the computing time. Model reduction generally assumes that the response u of a physical problem can often be approximated with reasonable precision by

$$u(\mathbf{x},t) \simeq \sum_{k=1}^{n} a^k(t)\Phi^k(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is the 2D or the 3D coordinate vector, $\Phi^k(\mathbf{x})$ is a low-dimensional reduced basis, and n is the reduced-basis size, which is usually much smaller than the full grid size of the discretized solution. The temporal coefficients are solutions of a very low-order system obtained by projection of the initial equation of the problem over this basis.

The most popular reduced-order modelling technique is the proper orthogonal decomposition (POD) [1–8]. POD uses a set of snapshots generated by evaluating the computational solution of a transient problem at several time points. The POD basis

* Corresponding author.
E-mail addresses: adumon01@univ-lr.fr (A. Dumon), cyrille.allery@univ-lr.fr (C. Allery), Amine.Ammar@ensam.eu (A. Ammar).

is given by the most representative vector corresponding to the most dominant eigenvalue of the snapshot vector matrix. As this technique requires at least one simulation to obtain a set of snapshots, this decomposition is known as an "*a posteriori*" reduced-order modelling technique. It is important to note that the so-called 'snapshot method' is not the only way to deal with POD; for example, Sengupta and Dey in [9] used the Lanczos algorithm to determine the representative eigenvalues of a problem.

The Centroidal Voronoi Tessellations (CVT) [10,11] is another "*a posteriori*" model reduction technique. In this method a special Voronoi clustering is constructed from the change in the snapshot set over time for which the means of the clusters are also the generators of the corresponding Voronoi clusters. These generators constitute the CVT reduced-order basis.

As mentioned before, the main drawback of techniques used previously is the need for a set of snapshots of the solution in order to construct the reduced basis. A lengthy computing time may be required for the calculation of these snapshots, that is why an "*a priori*" model reduction techniques have been developed. These consist in constructing a reduced basis without an "*a priori*" knowledge of the solution. A priori model reduction (APR) [12–16] has been the subject of several developments. Thanks to this approach, the basis is adaptively improved and expanded with the residuals of the full discretized model. The incremental process is carried out by taking into account the whole time interval during which the reduced equation is solved.

In this paper we will focus on another "*a priori*" model reduction technique. This method involves looking for a solution to a PDE as a product sum of the functions of each space variable. For example, if we search a field u dependent on N variables, this can be expressed by

$$u(x_1,\ldots,x_N)=\sum_{i=1}^{Q}\prod_{k=1}^{N}F_{ki}(x_k), \tag{2}$$

(x_i can be any scalar or vector variable involving space, time or any other parameter of the problem). Thus, if M degrees of freedom are used to discretize each variable, the total number of unknowns involved in the solution is $Q\times N\times M$ instead of the M^N degrees of freedom involved in mesh based discretization techniques. In most cases, where the field is sufficiently regular, the number of terms Q in the finite sum is generally quite small (a few dozen) and in all cases the approximation converges towards the solution of the full grid description (see [17,18]). It must be emphasized here that the functions are not known 'a priori'. These functions are adaptively computed by introducing the separated approximation of the representation into the model and then solving the resulting non-linear problem. PGD was first developed by Ammar et al. to solve the Fokker–Planck equation related to the probability distribution function of the multi-bead-spring chain described in a high dimension configuration space [19,20]. It has been also used in the quantum chemistry field [21]. For stochastic non-linear problems, PGD was introduced by Nouy [22–24] (in this context, PGD was initially called Generalized Spectral Decomposition). Finally, in the context of efficient non-linear solvers, the space–time PGD was successfully applied in [25] as well as within the LATIN framework in [26]. The multiscale approaches in space and time were analyzed in [27,28]. In [29], the authors considered the separated representation of models defined in general domains with non-homogeneous essential boundary conditions. Recently, in [30], it was demonstrated that for degenerate time-dependent partial differential equations, the PGD representation coincides with the POD solution.

In the context of fluid mechanics equations, PGD could be applied using one of the three following decompositions:

- The first one consists in a time space decomposition: $u(t,\mathbf{x})=\sum_i F^i_{t_i}(t)F^i_{\mathbf{x}_i}(\mathbf{x})$. In this case we are looking for a similar representation to an 'a priori' reduced model but obtained using a non incremental approach. We are looking directly for a time/space solution. The main drawback encountered here is that a full grid description is required to define the '$F_{\mathbf{x}_i}$' functions over the physical space.
- To circumvent this difficulty the second possibility consists in writing a full decomposition involving the two (or three) dimensions of the physical space: $u(t,x,y)=\sum_i F^i_{t_i}(t)F^i_{x_i}(x)F^i_{y_i}(y)$. In practice, making such a decomposition leads to a significant increase in the number of terms required in the sum.
- The third possibility, which was used in our work, consists in keeping the incremental approach and doing the separation over the physical space at each time step: $u^t(x,y)=\sum_i F^i_{x_i}(x)F^i_{y_i}(y)$.

In all cases the generic resolution strategy is applicable for each of these three possibilities. However, in our simulations, only the third one was adopted as it seemed to give the best results. In particular, it is able (conceptually) to solve problems involving 1000^3 degrees of freedom (in the space description); which is not possible within the POD or the APR resolution. In this context we applied PGD to the Navier–Stokes resolution framework in order to reduce CPU time, by decreasing the cost of the space representation, and to obtain a sufficiently accurate solution. We used a traditional fractional step algorithm to decouple the velocity and the pressure for the resolution of the Navier–Stokes equations.

This paper is organized as follows: first, we will summarise briefly the general idea of proper generalized decomposition (PGD); secondly, we will describe the Navier- Stokes discretisation and the model used to solve this problem; finally, we will present the results for different problems (Stokes, Burgers and Navier–Stokes).

2. Description of the PGD

For the sake of clarity and without losing its general scope, PGD will be examined in the case of a 2D space decomposition. The considered problem is expressed as follows:

$$\text{Find } U(x,y) \text{ as } \begin{cases} \mathcal{L}(U) = \mathcal{G}, & \text{in } \Omega, \\ +\text{Boundary Conditions}, \end{cases} \tag{3}$$

where $\mathcal{L}$ is a differential operator and $\mathcal{G}$ is the second member.

PGD, which is an iterative method, consists in finding an approximation of the solution $U(x,y) \in \Omega = X \times Y \subset \mathbb{R}^2$ with $x \in X \subset \mathbb{R}$ and $y \in Y \subset \mathbb{R}$ as:

$$U(x,y) \approx \sum_{i=1}^{n} F^i(x) G^i(y). \tag{4}$$

2.1. Progressive PGD

The first $n-1$ functions F^i and G^i are assumed to be known and the functions $F^n \in X$ and $G^n \in Y$ are calculated such that:

$$U(x,y) \approx \sum_{i=1}^{n-1} F^i(x) G^i(y) + F^n(x) G^n(y). \tag{5}$$

Introducing this expression into (3) gives:

$$\mathcal{L}\left(\sum_{i=1}^{n-1} F^i(x) G^i(y) + F^n(x) G^n(y) \right) = \mathcal{G} + Res^n, \tag{6}$$

where Res^n is a residual due to the fact that Eq. (5) is an approximation of the solution. To determine F^n and G^n, Eq. (6) is projected onto each of the unknowns F^n and G^n:

$$\left\langle \mathcal{L}\left(\sum_{i=1}^{n} F^i(x) G^i(y) + F^n(x) G^n(y) \right), F^n \right\rangle_{L^2(X)} = \left\langle \mathcal{G}, F^n \right\rangle_{L^2(X)} + \left\langle Res^n, F^n \right\rangle_{L^2(X)} \tag{7}$$

and

$$\left\langle \mathcal{L}\left(\sum_{i=1}^{n-1} F^i(x) G^i(y) + F^n(x) G^n(y) \right), G^n \right\rangle_{L^2(Y)} = \left\langle \mathcal{G}, G^n \right\rangle_{L^2(Y)} + \left\langle Res^n, G^n \right\rangle_{L^2(Y)}, \tag{8}$$

where $\langle \cdot, \cdot \rangle_{L^2(X)}$ ($\langle \cdot, \cdot \rangle_{L^2(Y)}$, respectively) are the scalar products[1] on L^2, in the x direction (in the y direction, respectively). In addition with this approach, the residual must be orthogonal to the F^n and G^n functions, thus

$$\left\langle \mathcal{L}\left(\sum_{i=1}^{n-1} F^i(x) G^i(y) + F^n(x) G^n(y) \right), F^n \right\rangle_{L^2(X)} = \left\langle \mathcal{G}, F^n \right\rangle_{L^2(X)} \tag{9}$$

and

$$\left\langle \mathcal{L}\left(\sum_{i=1}^{n-1} F^i(x) G^i(y) + F^n(x) G^n(y) \right), G^n \right\rangle_{L^2(Y)} = \left\langle \mathcal{G}, G^n \right\rangle_{L^2(Y)}. \tag{10}$$

In order to obtain the new functions F^n and G^n, Eqs. (9) and (10) must be solved simultaneously. The natural algorithm which allows this is the iterative power (or fixed point) algorithm.

Before giving a general description, the following functions will first be described:

- $S_n : X \to Y$ is the application that associates a function dependent on x, $F^{n+1} \in X$, with a function that is dependent on y, $G^{n+1} \in Y$, defined by Eq. (9).
- $T_n : Y \to X$ is the application that associates a function dependent on y, $G^{n+1} \in Y$, with a function that is dependent on x, $F^{n+1} \in X$, defined by Eq. (10).

The algorithm associated with a **progressive PGD** is thus expressed as:

[1] The scalar product H^1 could also have been used.

1. ***for*** $n = 1, n_{max}$ ***do***
2. *Initialisation of* $G^{(0)}$
3. ***for*** $k = 1, k_{max}$ ***do***
4. *Compute* $F^{(k)} = T^n\left(G^{(k-1)}\right)$
5. *Compute* $G^{(k)} = S^n(F^{(k)})$
6. *Verification of the convergence of* $\left(F^{(k)}G^{(k)}\right)$
   $\left(\text{it is required that } \left\|F^{(k)}G^{(k)} - F^{(k-1)}G^{(k-1)}\right\| \leqslant \eta, \text{ where } \eta \text{ is set by the user, and } \|\bullet\| = \langle\bullet,\bullet\rangle_{L^2(\Omega)}\right)$
7. ***end for***
8. *U is defined as* $\sum_{i=1}^{n} F^i(x)G^i(y)$ *and the convergence is checked for U*
   (it is required that $\|Res^n\|_{L^2(\Omega)} \leqslant \epsilon$, *where* ϵ *is set by the user)*
11. ***end for***

Nouy showed in [30] that the best results were obtained when the residual was orthogonal to each of the functions of the decomposition, which is why a second algorithm was introduced.

2.2. Progressive PGD with projection

A solution is sought in the form

$$U(x,y) \approx \sum_{i=1}^{n} \alpha^i F^i(x) G^i(y). \tag{11}$$

Assuming we begin at step $(n-1)$, the functions $F^n(x)$ and $G^n(y)$ are progressively calculated with $\alpha^n = 1$. The following two problems must then be solved

$$\left\langle \mathcal{L}\left(\sum_{i=1}^{n-1} \alpha^i F^i(x)G^i(y) + F^n(x)G^n(y)\right), F^n \right\rangle_{L^2(X)} = \langle \mathcal{G}, F^n\rangle_{L^2(X)} \tag{12}$$

and

$$\left\langle \mathcal{L}\left(\sum_{i=1}^{n-1} \alpha^i F^i(x)G^i(y) + F^n(x)G^n(y)\right), G^n \right\rangle_{L^2(Y)} = \langle \mathcal{G}, G^n\rangle_{L^2(Y)}. \tag{13}$$

Eqs. (12) and (13) are solved using the fixed point method. After convergence of the fixed point, the first n functions F^i and G^i are known and the $n\alpha^i$ coefficients must be computed.

In order to do this, solution (11) is introduced into (3):

$$\mathcal{L}\left(\sum_{i=1}^{n} \alpha^i F^i(x)G^i(y)\right) = \mathcal{G} + Res^n. \tag{14}$$

The α^i coefficients are then computed in such a way that the residual is orthogonal to each of the n products of the F^iG^i functions, by projecting the above equation according to the F^iG^i:

$$\left\langle \mathcal{L}\left(\sum_{i=1}^{n} \alpha^i F^i(x)G^i(y)\right), F^kG^k \right\rangle = \langle \mathcal{G}, F^kG^k\rangle, \quad \text{for } 1 \leqslant k \leqslant n. \tag{15}$$

The following functions can be defined

- $S_n^1 : X \to Y$ is the application that associates a function dependent on x, $F^n \in X$, to a function dependent on y, $G^n \in Y$, defined by Eq. (12).
- $T_n^1 : Y \to X$ is the application that associates a function dependent on y, $G^n \in Y$, to a function dependent on x, $F^n \in X$, defined by Eq. (13).
- $W : (X)^n \times (Y)^n \to \mathbb{R}^n$ is the application that associates functions $\mathbb{F}^n = \{F^i\}_{i=1}^n$ et $\mathbb{G}^n = \{G^i\}_{i=1}^n$ to the vector $\Lambda = \{\alpha_i\}_{i=1}^n \in \mathbb{R}^n$, defined by Eq. (15).

The PGD resolution algorithm combining the enrichment phase and the computation of the coefficients using a projection based on an approximation called **progressive PGD with projection** is summarized below:

1. **for** $n = 1, n_{max}$ **do**
2. *Initialisation of* $G^{(0)}$
3. **for** $k = 1, k_{max}$ **do**
3. *Computation of* $F^{(k)} = T_n^1\left(G^{(k-1)}\right)$
4. *Computation of* $G^{(k)} = S_n^1\left(F^{(k)}\right)$
7. *Verification of the convergence of* $(F^{(k)}G^{(k)})$
8. **end for**
9. *Let* $\mathbb{F}^n = \left\{\mathbb{F}^{n-1}, F^{(k)}\right\}$ *and* $\mathbb{G}^n = \left\{\mathbb{G}^{n-1}, G^{(k)}\right\}$
10. *Computation of* $\Lambda_m = \{\alpha^i\}_{i=1}^n = W(\mathbb{F}^n, \mathbb{G}^n)$
11. *Let* $U = \sum_{i=1}^n \alpha^i F^i(x) G^i(y)$ *and check convergence for* U $\left(\|Res^n\|_{L^2(\Omega)} \leqslant \epsilon\right)$
12. **end for**

In the work that follows the latter method was used, which is the reason why an algebraic version is given in the appendix (see Appendix A).

3. Navier–Stokes discretization

This section deals with the numerical solution of the unsteady incompressible Navier–Stokes equations

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{f}, \quad \text{in } \Omega\times[0,T], \\ \nabla\cdot\mathbf{u} = 0, \\ \mathbf{u}(t=0) = \mathbf{u}_0, \\ \mathbf{u} = \mathbf{g}_D, \quad \text{on } \partial\Omega, \end{cases} \tag{16}$$

where $\mathbf{u}$ and p are the velocity and the pressure fields, respectively, and $\mathbf{f}$ is a known body force. ν is the kinematic viscosity and ρ is the constant fluid density.

3.1. Fractional step algorithm

The main difficulty encountered in the resolution of the Navier–Stokes equations resides in the velocity–pressure coupling within the continuity equation. In this work these fields are decoupled using a fractional step method that consists in solving the momentum equation at each time step in order to find an estimated velocity. A Poisson's equation can then be solved with this estimated velocity, thus providing a pseudo-pressure. Solving these two equations gives a reconstruction of the velocity and the pressure within these two estimated fields with respect to the continuity equations. For this purpose, we used the Van-Kan algorithm (see [31,32]).

At the time step n, we are looking for $\mathbf{u}^{n+1}$ and p^{n+1}. The estimated velocity $\tilde{\mathbf{u}}$ is the solution of the following problem:

$$\begin{cases} \frac{1}{\Delta t}(\tilde{\mathbf{u}} - \mathbf{u}^n) - \nu\Delta\left(\frac{\tilde{\mathbf{u}}-\mathbf{u}^n}{2}\right) + \frac{1}{\rho}\nabla(p^n) + \frac{1}{2}\left[3\nabla\cdot(\mathbf{u}\mathbf{u})^n - \nabla\cdot(\mathbf{u}\mathbf{u})^{n-1}\right] = 0, \\ \tilde{\mathbf{u}}^n = \mathbf{g} \text{ sur } \partial\Omega. \end{cases} \tag{17}$$

An explicit Adams–Bashforth scheme is used for the advection term and a Crank- Nicholson discretization is applied to the diffusion term.

A pseudo-pressure $\tilde{p}$ is now defined and we have to solve the following Poisson's equations with Neumann's boundary conditions:

$$\begin{cases} \Delta\tilde{p} = \frac{2}{\Delta t}\nabla\cdot\tilde{\mathbf{u}}, \\ \frac{\partial\tilde{p}}{\partial n} = 0 \text{ sur } \partial\Omega. \end{cases} \tag{18}$$

Therefore, in order to verify the continuity equation, the new pressure p^{n+1} and velocity $\mathbf{u}^{n+1}$ are updated with:

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \frac{\Delta t}{2}\nabla\tilde{p}, \tag{19}$$

$$p^{n+1} = p^n + \tilde{p}. \tag{20}$$

3.2. Spatial discretization

A staggered grid discretization was chosen for the resolution of problems (17) and (18). With this kind of grid, the calculation of velocity components (u_1 and u_2) and pressure are not made at the same points in order to avoid a non physical

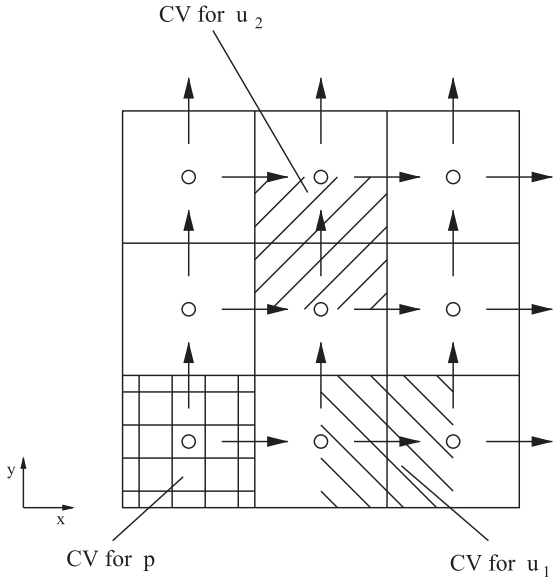


Fig. 1. Arrangements of variables in a staggered grid.

pressure gradient (see [33]). Fig. 1 represents schematically the considered discretization. Each estimated velocity component is calculated at the same control volume as the corresponding velocity component. Similarly, the pseudo-pressure is calculated at the same control volume as for the pressure field. The equations are solved with a finite volume method, using a second-order scheme.

3.3. Associated PGD

This problem is solved using the PGD algorithm. Assuming we are at time step n, p^n and $\mathbf{u}^n$ are both known and we want to find p^{n+1} and $\mathbf{u}^{n+1}$. To do this the following steps are carried out:

(1) The intermediate velocity $\tilde{u}_i$ is sought in the form

$$\tilde{u}_i = \sum_{k=1}^{N_{\tilde{u}_i}} \alpha_{\tilde{u}_i}^k F_{\tilde{u}_i}^k(x) G_{\tilde{u}_i}^k(y), \quad \text{for } i = 1, 2. \tag{21}$$

To do this problem equation (17) is solved by using the progressive PGD with projection (Section 2.2).

(2) The intermediate pressure $\tilde{p}$ is sought in the form

$$\tilde{p} = \sum_{k=1}^{N_{\tilde{p}}} \alpha_{\tilde{p}}^k F_{\tilde{p}}^k(x) G_{\tilde{p}}^k(y). \tag{22}$$

To do this problem equation (18) is solved.

(3) The velocity and pressure fields $\mathbf{u}^{n+1}$ and p^{n+1} are updated using Eqs. (19) and (20).

4. Numerical results

We propose to solve Navier–Stokes equations with the Van-Kan scheme as described in Section 3.1. In order to validate our approach, we first considered some academic problems of Stokes and Burgers equations in two dimensions where analytical solutions are provided. Then we treated the problem of the lid-driven cavity in the 2D case. In the following we will denote the two velocity components by u_1 and u_2.

4.1. Stokes equation

4.1.1. Problem

The incompressible Stokes problem may be expressed as follows:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} = -\nabla p + \mathbf{f}, \quad \text{on } \Omega \times [0, T], \\ \nabla \cdot \mathbf{u} = 0, \\ \mathbf{u}(t = 0) = \mathbf{u}_0, \\ \mathbf{u} = \mathbf{g}_D, \quad \text{on } \partial\Omega. \end{cases} \tag{23}$$

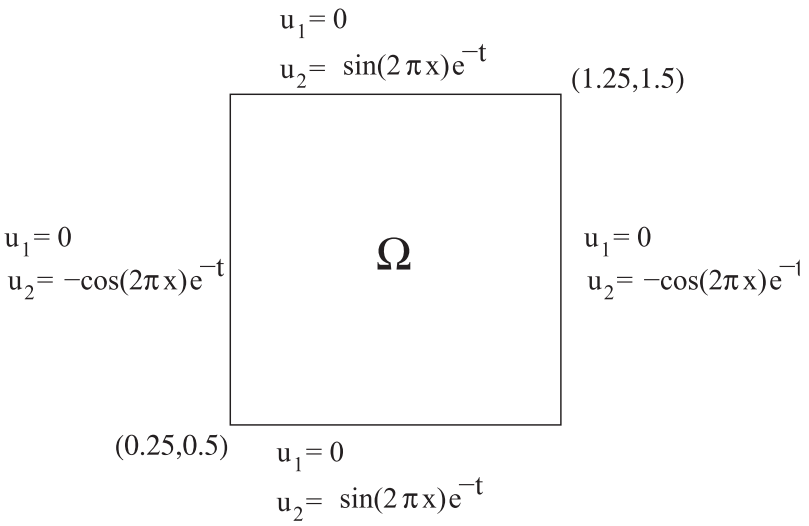


Fig. 2. Geometry and boundaries conditions for the Stokes problem.

These equations are defined over a 2D square domain $\Omega = (0.25, 1.25) \times (0.5, 1.5)$. We impose the following source term $f(f_x \otimes f_y)$:

$$\begin{aligned} f_x &= (1-2\pi)(8\pi^2 v - 1)cos(2\pi x)sin(2\pi y)e^{-t}, \\ f_y &= (1+2\pi)(1-8\pi^2 v)cos(2\pi x)sin(2\pi y)e^{-t}. \end{aligned} \tag{24}$$

In this case the problem has the following analytical solution:

$$\begin{aligned} u_1 &= cos(2\pi x)sin(2\pi y)e^{-t}, \\ u_2 &= -sin(2\pi x)cos(2\pi y)e^{-t}, \\ p &= (1-8\pi^2 v)sin(2\pi x)sin(2\pi y)e^{-t}. \end{aligned} \tag{25}$$

The boundary conditions were chosen to verify this analytical solution. Fig. 2 shows the domain and the boundary conditions used for this problem.

Here we solved the Navier–Stokes equations with the Van-Kan algorithm. Problem[2] was solved with this algorithm, detailed previously, using the PGD solver (detailed in Appendix A) and another full grid standard solver (bi-conjugate gradient), for comparison. The simulations were done with a viscosity v equal to 10^{-2} and a time step Δt equal to 10^{-3}. The real time simulation was set to 1 s.

4.1.2. Results

Fig. 3 represents the velocity vectors computed from the PGD method and the difference between the PGD and the analytical velocity fields using 250 nodes for each direction. This figure shows the good correlation between the velocity field obtained by PGD and the analytical one. Fig. 4 represents the pressure field computed from the PGD and the difference between the PGD and the analytical pressure fields. The difference between these two fields exhibits an error with an absolute maximum of 10^{-3}. Thus it may be concluded that PGD is a very efficient tool for solving Stokes equations. To underline the benefits of PGD, Fig. 5 illustrates the change in velocity and pressure errors in relation to the number of discretization nodes (being the same in each direction). This was done for the results obtained with the PGD method and with the standard method. The velocity and pressure errors are defined by:

$$\begin{aligned} \|\mathbf{e}\| &= \|\mathbf{u}_h - \mathbf{u}\|_{L^\infty(L^2(\Omega))} = \max_{0<t\leqslant T}\left[\int_\Omega |\mathbf{u}_h - \mathbf{u}|^2 d\omega\right]^{\frac{1}{2}}, \\ \|\eta\| &= \|p_h - p\|_{L^2(L^2(\Omega))} = \left[\int_0^T \int_\Omega |p_h - p|^2 d\omega\, dt\right]^{\frac{1}{2}}. \end{aligned} \tag{26}$$

Fig. 5 shows that the results for velocity are very similar. Indeed, with PGD and with the standard solver, the error in pressure seems to be slightly different, although the general behaviour was the same. We compared the CPU time between the PGD simulation and the standard solver. These results are illustrated in Fig. 6. As can be seen, beyond 50 nodes in each direction the PGD CPU time seems to be shorter than the standard solver CPU time. When the number of nodes was increased,

[2] This is the same problem as described in Section 3.1, without the convection term.

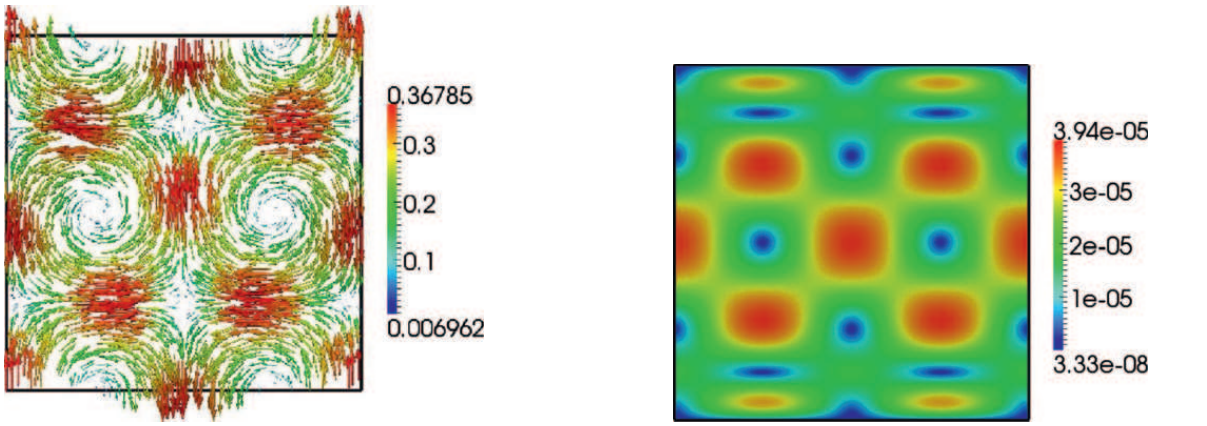


Fig. 3. Velocity vectors computed from PGD (left) and the difference between PGD and analytical velocity fields (right) after 1 s real time simulation with $N_h = 250$.

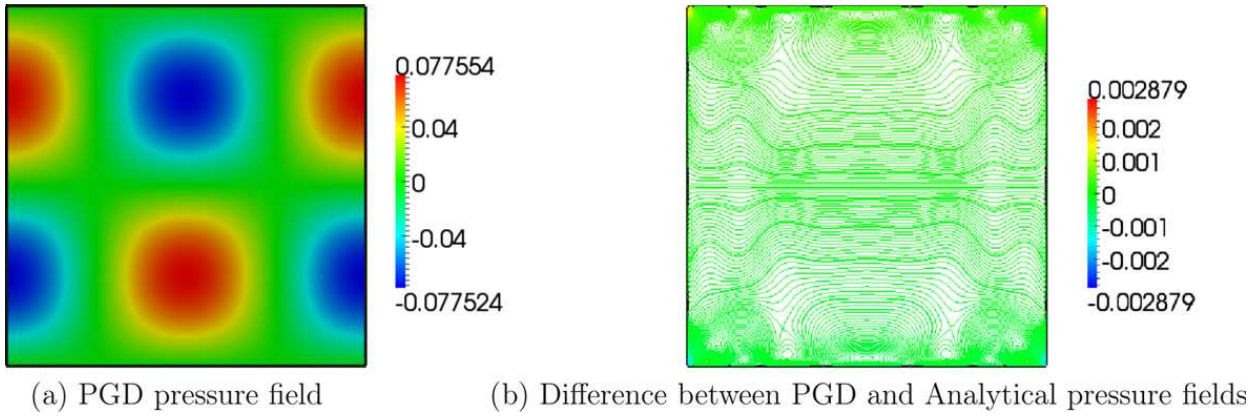


Fig. 4. Pressure field computed from PGD (left) and the difference between PGD and analytical pressure fields (right) after 1 s real time simulation with $N_h = 250$.

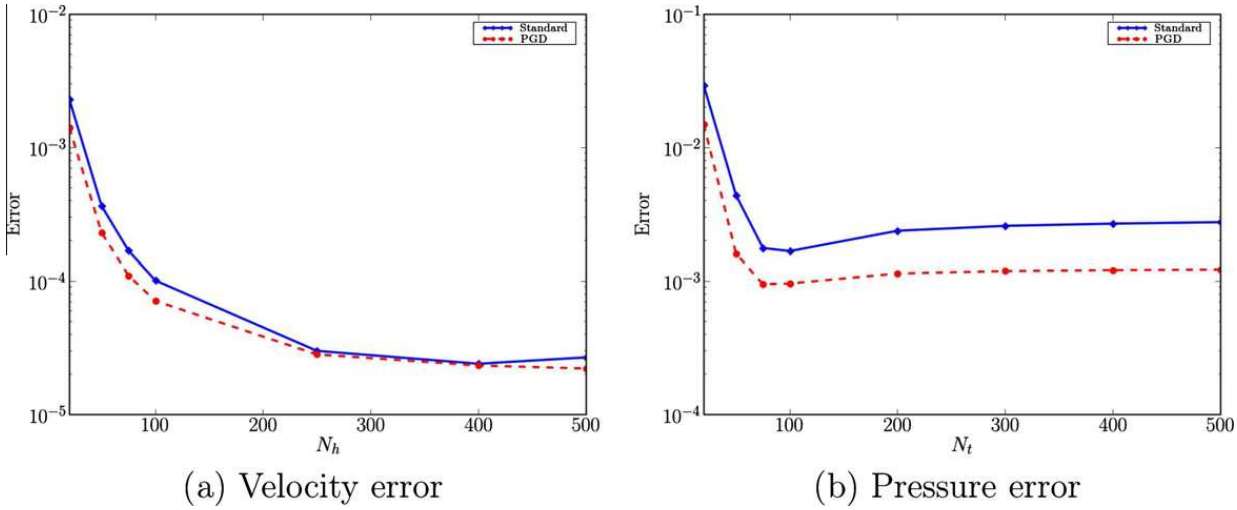


Fig. 5. Change in error for the velocity and for the pressure with the number of nodes for PGD and standard solver.

PGD exhibited a greater efficiency. Indeed, with 100 nodes in each direction, the calculation with PGD was seven times faster than the calculation with the standard full grid solver. Similarly, with 500 nodes in each direction, PGD was thirty-five times faster.

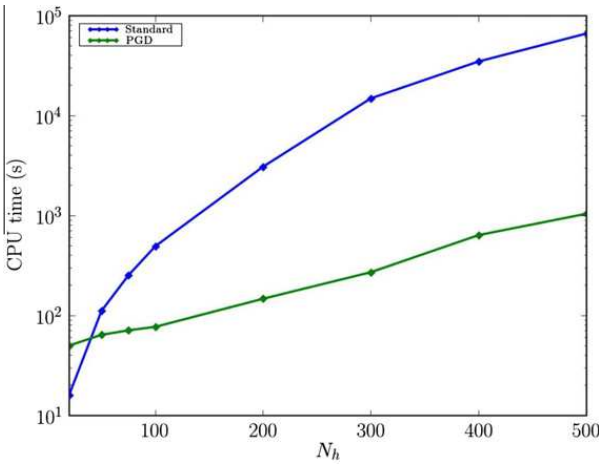


Fig. 6. CPU time for the PGD and standard solvers.

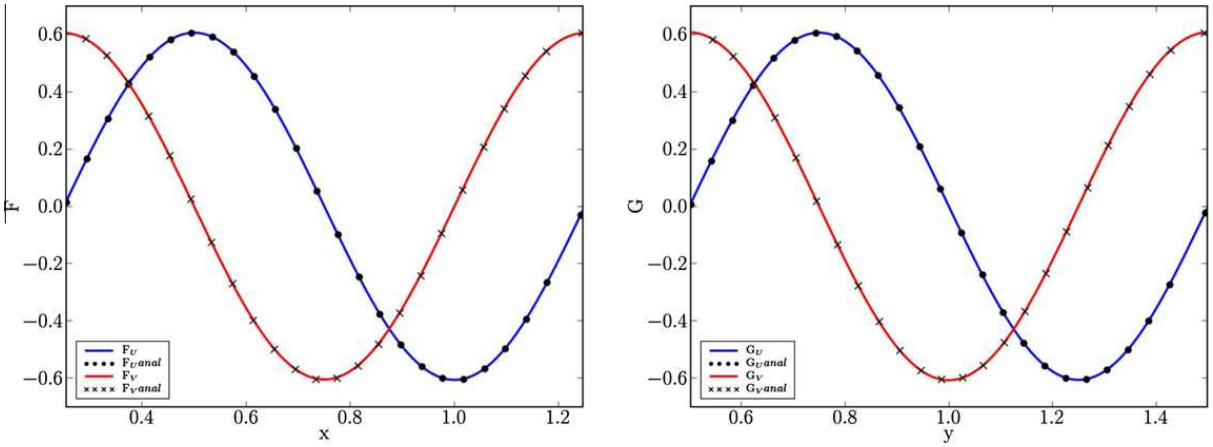


Fig. 7. Functions of the tensor product computed with PGD and from an analytical solution of the fields $\tilde{U}_1$ and $\tilde{U}_2$ after 1 s real time simulation.

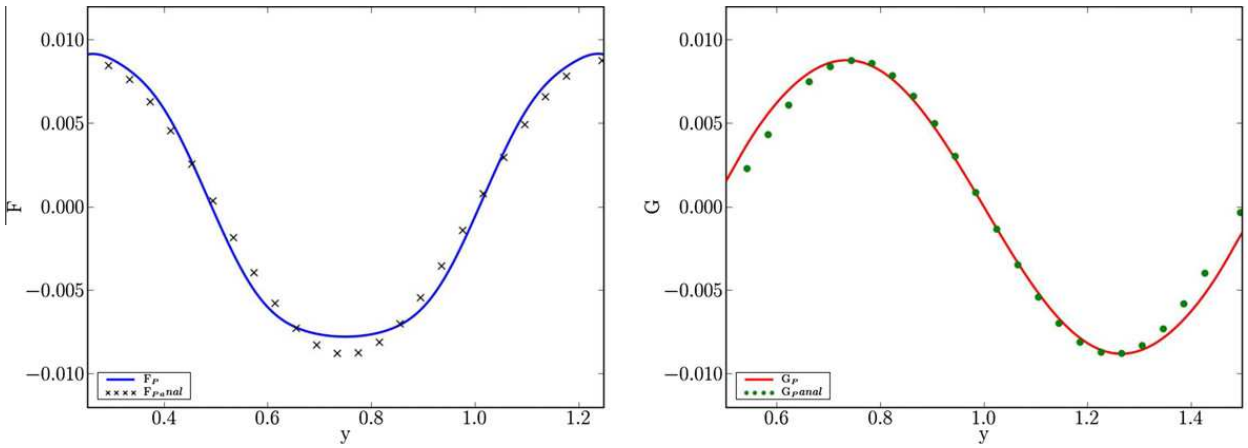


Fig. 8. Functions of the tensor product computed with PGD and from an analytical solution that represent the field $\tilde{p}$ after 1 s real time simulation.

Finally, the estimated pseudo-velocity and the pseudo-pressure of Eqs. (17) and (18) were found in one enrichment (see Figs. 7 and 8). This corresponds to the estimated analytical solution,[3] which can be decomposed into one function product. This implies that PGD is an optimal representation of the solution to the 2D Stokes Problem.

[3] Calculated from the analytical solution.

4.2. Burgers equation

Now, we will study the Burgers equation which is a kind of Navier–Stokes equation without the pressure term.

4.2.1. Problem

The Burgers problem is stated as follows:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{f}, & \text{in } \Omega \times [0, T], \\ \mathbf{u}(t = 0) = \mathbf{u}_0, \\ \mathbf{u} = \mathbf{g}_D, & \text{on } \partial\Omega. \end{cases} \tag{27}$$

This problem is defined in two dimensions over a domain $\Omega = (0.25, 1.25) \times (0.5, 1.5)$. The considered source term f is given by the tensorial product $f_x \otimes f_y$:

$$\begin{aligned} f_x &= -\pi sin(4\pi x) * e^{-16\pi^2 \nu t}, \\ f_y &= -\pi sin(4\pi x) * e^{-16\pi^2 \nu t}. \end{aligned} \tag{28}$$

In this case, the problem has the following analytical solution:

$$\begin{aligned} u_1 &= -cos(2\pi x) sin(2\pi y) e^{-8\pi^2 \nu t}, \\ u_2 &= sin(2\pi x) cos(2\pi y) e^{-8\pi^2 \nu t}. \end{aligned} \tag{29}$$

As with the Stokes equations, the boundary conditions were chosen to verify the analytical solution. Fig. 9 illustrates both the domain and the boundary conditions used for this problem. Here we solved the Burgers equation with the Van-Kan algorithm. Eq. (27) was solved with a Crank–Nicholson scheme for the diffusive term, and an Adams–Bashforth scheme for the convective term, in order to have the same discretization as for the Van-Kan scheme (Eq. (17)). The simulations were performed with a viscosity ν equal to 10^{-2}, a time step Δt equal to 10^{-3} for a total real simulation time of 1 s.

4.2.2. Results

Similarly to the Stokes problem, the difference between the velocity vectors obtained by using the PGD method and the difference between the PGD and the analytical velocity fields can be seen in Fig. 10. These result prove that PGD is an alternative that can be used to solve the Burgers equation. Fig. 11 shows the change in the velocity error (defined in Eq. (26)) in relation to the number of nodes. The change in this error is the same for PGD and for the standard scheme, although the error of the standard scheme does appear to be slightly lower.

Fig. 12 shows the CPU time and it is clear that the PGD method is faster than the standard scheme. With 500 nodes in each direction, PGD was fifteen times faster than the standard scheme.

Finally, it should be noted that the solution was found in one enrichment for x and y for the pseudo-velocity (see Fig. 13). This corresponds to the estimated analytical solution, which can be decomposed into one function product and implies that PGD is an optimal representation of the solution to the 2D Burgers Problem.

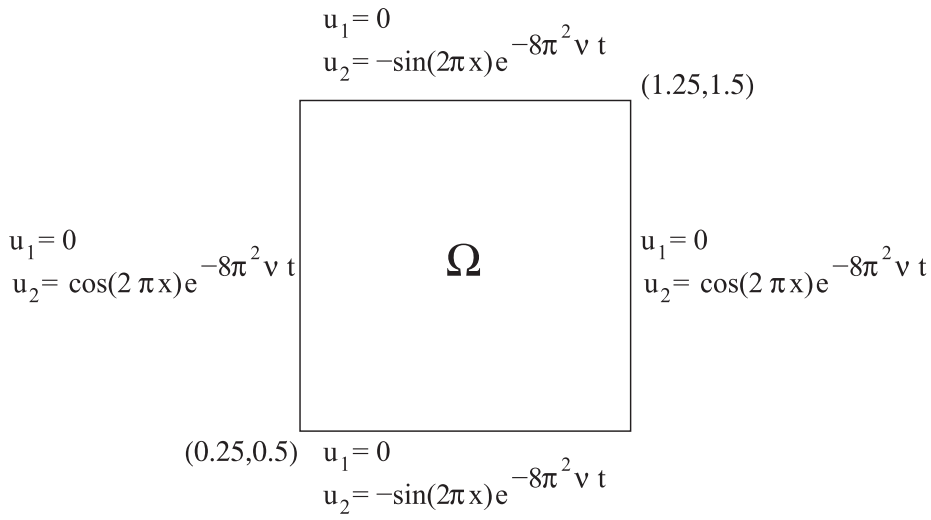
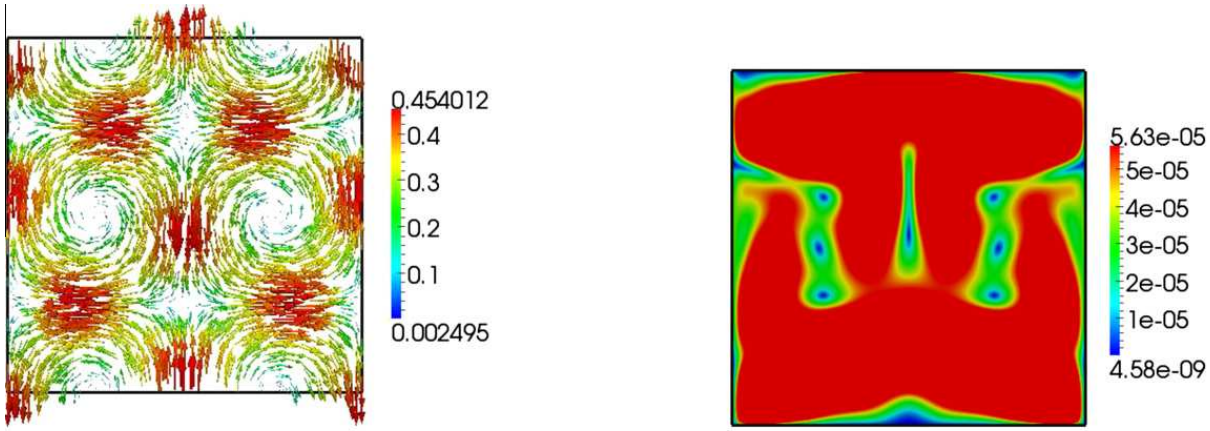


Fig. 9. Geometry and boundaries conditions of the Burgers problem.



(a) PGD velocity vectors (b) Difference between PGD and Analytical velocity fields

Fig. 10. Velocity vectors computed with PGD (left) and the difference between PGD and analytical velocity fields after 1 s real time simulation with $N_h = 250$.

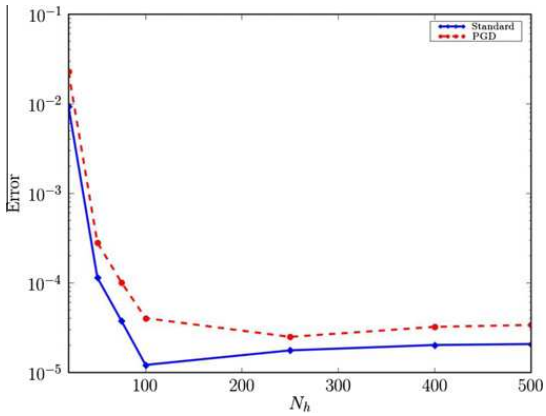


Fig. 11. Error in velocity as a function of the number of nodes for 1 s real time simulation.

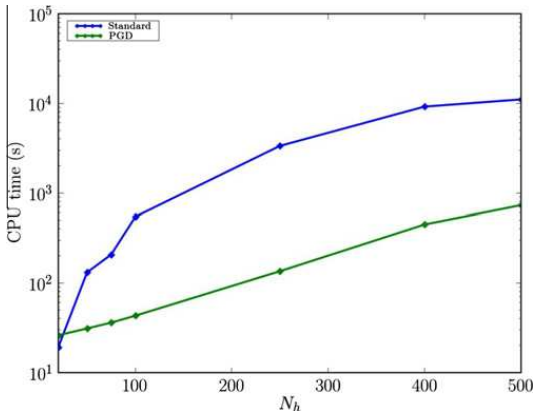


Fig. 12. CPU time for the PGD and standard solvers.

4.3. The Navier–Stokes problem

First, we will examine the lid-driven cavity in the stationary case ($Re = 100$ and $Re = 1000$) for which we can compare the results with those of Ghia et al. [34]. Then we will present our results for a post-critical Reynolds number (here $Re = 10{,}000$) and compare them with those obtained by Bruneau and Saad in [35].

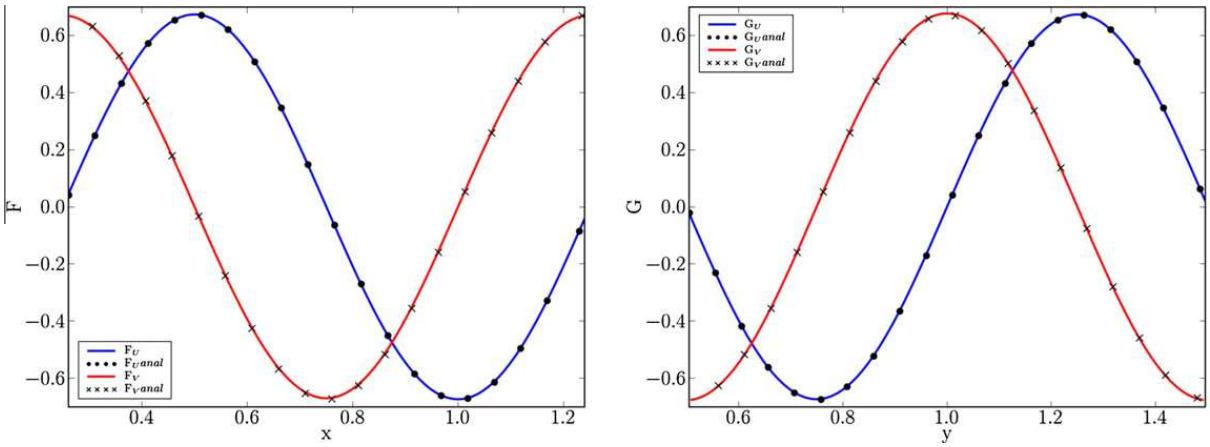


Fig. 13. Functions of the tensor product computed with PGD and from analytical solution that represent the fields $\tilde{U}_1$ and $\tilde{U}_2$ after 1 s real time simulation.

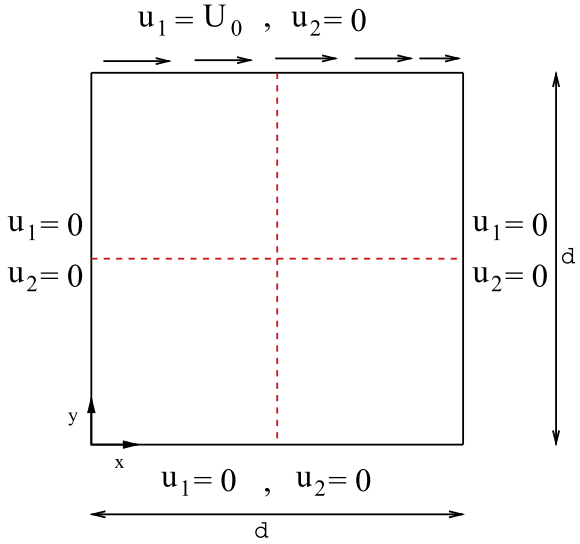


Fig. 14. Geometry of lid-driven cavity.

4.3.1. The steady lid-driven cavity

Let us consider the square domain $\Omega = (0,1) \times (0,1)$ for the resolution of the Navier–Stokes equations with Dirichlet conditions, as illustrated in Fig. 14. Here the two velocity components vanish on the boundary, except on the north face where the x-velocity is equal to U_0. The simulations were made with $\Delta t = 10^{-3}$, for two Reynolds numbers[4] ($Re = 100$ and $Re = 1000$) and with the source term f (see Eq. (16)) equal to zero.

Since we were looking for a stationary flow, we defined the following convergence criteria:

$$\frac{\|u_i^k - u_i^{k-1}\|}{\|u_i^k\|} \leqslant \epsilon_u \quad \text{with } i = 1,2, \tag{30}$$

$$\frac{\|\nabla p^k - \nabla p^{k-1}\|}{\|\nabla p^k\|} \leqslant \epsilon_p. \tag{31}$$

In the following cases the values $\epsilon_u = 5.10^{-8}$ and $\epsilon_p = 10^{-4}$ were considered.

Fig. 15 denotes the comparison between the streamlines computed from the PGD method and the standard method for $Re = 100$ and $Re = 1000$. Finally, Fig. 16 illustrates the comparison between the x-velocity at $x = 0.5$ and the y-velocity at $y = 0.5$ (see the dashed line on Fig. 14) computed with the PGD solver, and the results obtained by Ghia et al. [34] for the two Reynolds numbers considered. This latter figure was obtained with the same number of nodes in each direction ($N_x = 250$). The results obtained with the PGD method clearly correspond to those obtained by the standard method and Ghia's results.

[4] $Re = \frac{U_0 \times d}{\nu}$ where d is the width of the cavity.

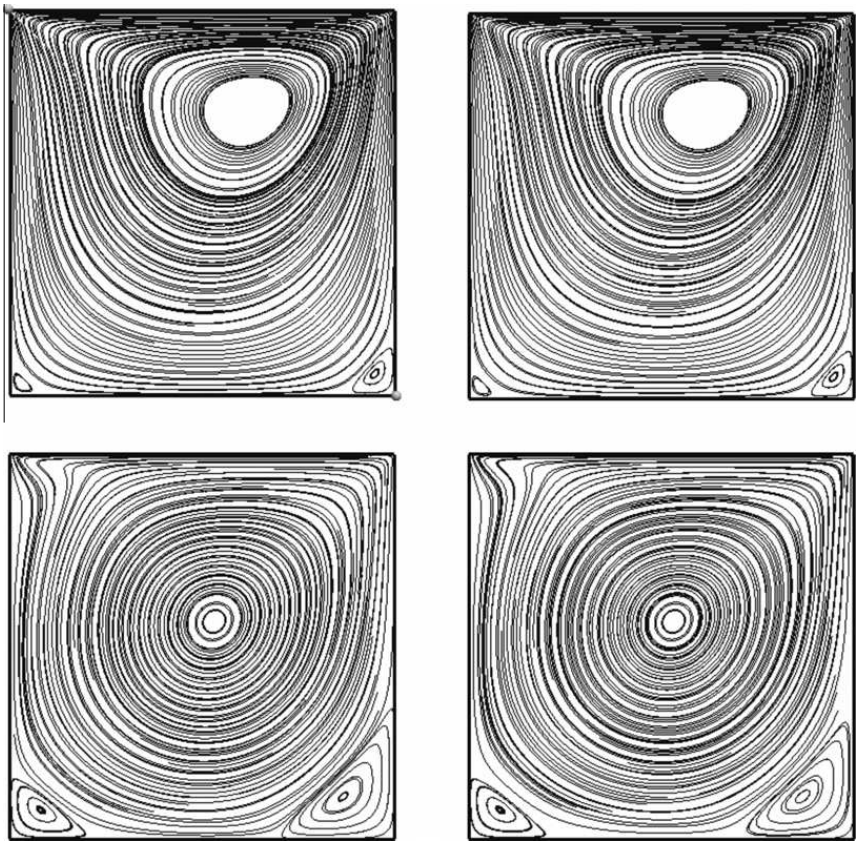


Fig. 15. Streamlines computed with PGD (left) and computed from the standard solver (right) with $N_h = 250$ for $Re = 100$ (top) and $Re = 1000$ (bottom).

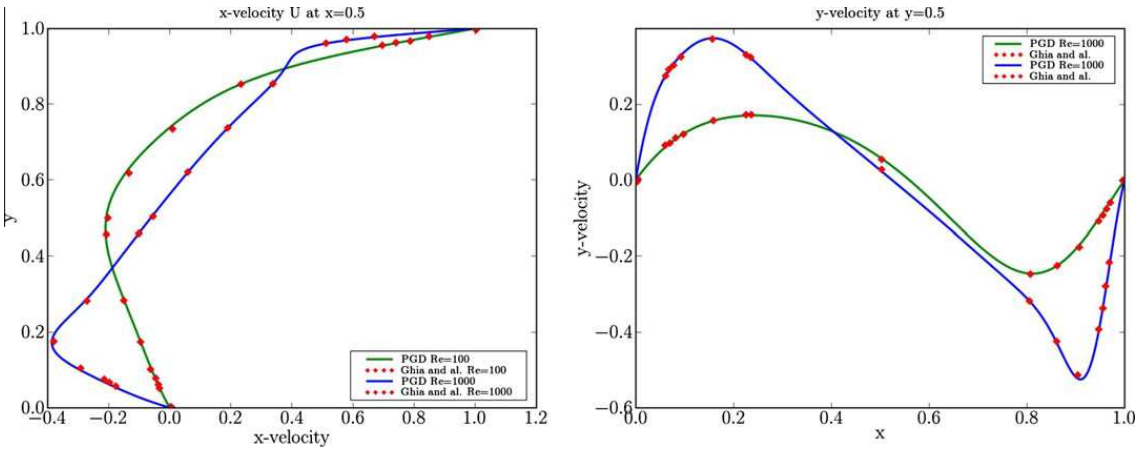


Fig. 16. Comparison of the x-velocity at $x = 0.5$ (left) and y-velocity at $y = 0.5$ (right) with the results of Ghia et al. for $Re = 100$ and $Re = 1000$.

It is interesting to compare the CPU time of these two solvers. Fig. 17 shows this comparison for $Re = 100$ and for $Re = 1000$. In this figure it can be seen that beyond a mesh size of 150×150, PGD becomes faster than the standard method. In fact, for a mesh size of 250×250, PGD was twice as fast as the standard method for each Reynolds number. Similarly, for a mesh size of 500×500, the CPU time was six times lower with the PGD solver for $Re = 100$. For $Re = 1000$, with the same grid (500×500), PGD was eight times faster than the standard solver.

4.3.2. The unsteady lid-driven cavity

Here we will consider the same lid-driven cavity as in the previous case, with the same boundary conditions. We will study the case of a post-critical Reynolds number where the flow is unsteady. We chose to fix this Reynolds number at 10,000. The simulations were made with $\Delta t = 10^{-4}$ and with a 250×250 mesh grid.

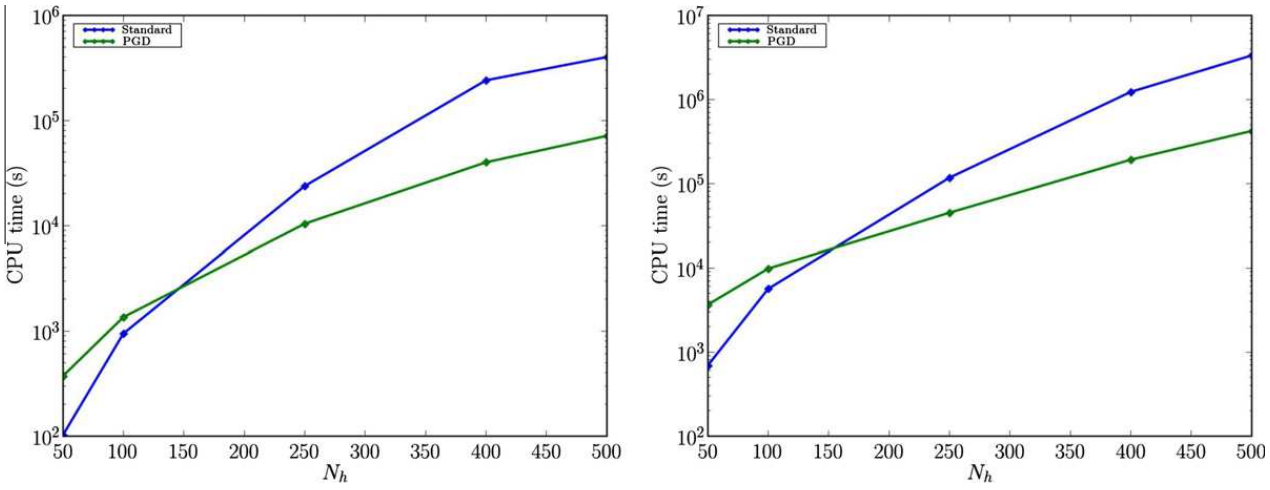


Fig. 17. Comparison of CPU time between the PGD solver and standard solver for the resolution in lid-driven cavity for Re = 100 (left) and Re = 1000 (right).

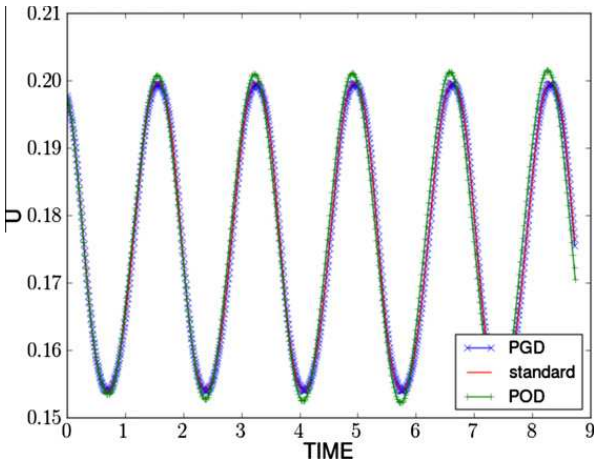


Fig. 18. Horizontal velocity (left) at monitoring point (2/16,13/16) for Re = 10,000.

Table 1
Eigenvalue associated with a mode n, fluctuating energy contained in the first n modes and the reconstruction error when n modes are maintained.

Mode n	1	2	3	4	5	6	7	8
Eigenvalue	$5.23e^{-5}$	$4.84e^{-5}$	$6.62e^{-7}$	$6.49e^{-7}$	$1.87e^{-8}$	$1.84e^{-8}$	$1.00e^{-9}$	$9.93e^{-10}$
Energy	51.285	98.676	99.325	99.961	99.979	99.997	99.998	99.999
Error	$9.7e^{-1}$	$1.38e10^{-2}$	$1.31e^{-2}$	$4.26e^{-4}$	$4.24e^{-4}$	$3.96e^{-5}$	$3.47e^{-5}$	$2.02e^{-5}$

The results obtained with PGD were again compared with those obtained using a standard simulation and also to results obtained using a POD-ROM resolution. The POD basis was obtained using the snapshots[5] applied to the fluctuating velocity $\mathbf{u}'(\mathbf{x},t) = \mathbf{u}(\mathbf{x},t) - \overline{\mathbf{u}}(\mathbf{x})$ where $\overline{\mathbf{u}}$ is the average velocity. In order to construct the eignevalue problem, 438 snapshots of the flow were taken every 2×10^{-2} s. Table 1 lists the proper values λ_n associated with each mode POD n, the energy contained in the first n modes defined by

$$\left\|\mathbf{u}'_{full} - \mathbf{u}'_n\right\|_{L2(\Omega)} \Big/ \left\|\mathbf{u}'_{full}\right\|_{L2(\Omega)}.$$

This table shows that with only a very few modes almost all of the energy can be captured and a good reconstruction of the fluctuating solution ensured. Thus with eight modes more than 99.99% of the energy is captured with a reconstruction error of the fluctuating velocity of the order of 2×10^{-5}.

[5] A short description of the Snapshots-POD method and the associated dynamic system is given in the annexe (see Appendix B).

The reduced model associated with POD was therefore built with eight modes and solved using a fourth order Runge–Kutta scheme. It should be noted that this system was stabilised by adding a numerical dissipation function of the mode number.

The horizontal velocity at the reference point (2/16,13/16) over time is shown on Fig. 18 for each of the solvers studied (PGD, POD and the standard). It can be seen from this figure that there is a good level of accuracy in the computation for the PGD method compared with the standard model. POD also reproduced the results obtained with the standard model with quite a good level of accuracy. This figure indicates a period of 1.69 s, which is close to the period of 1.64 s obtained by Bruneau and Saad in [35].

These results are different from these obtain by Sengupta et al. in [36] who find a period of 2.29 s. This difference could certainly be explain by the choice of the order's discretisation schemes used. In fact, Sengupta et al. used an high-order scheme (Combined Compact Difference scheme) while a second order scheme was used in this study. As Sengupta et al. mention in [36], the use of low order discretisation scheme and/or the use of an insufficiently refined grid in the case of a lid driven cavity with Reynolds numbers that are higher than a critical Reynolds, leads to instabilities that affect the dynamics of the flow. However, the aim of this work was to show the capacity of PGD to reproduce the results obtained with a standard model. The results obtained using the standard model and with PGD are similar to those obtained by Bruneau and

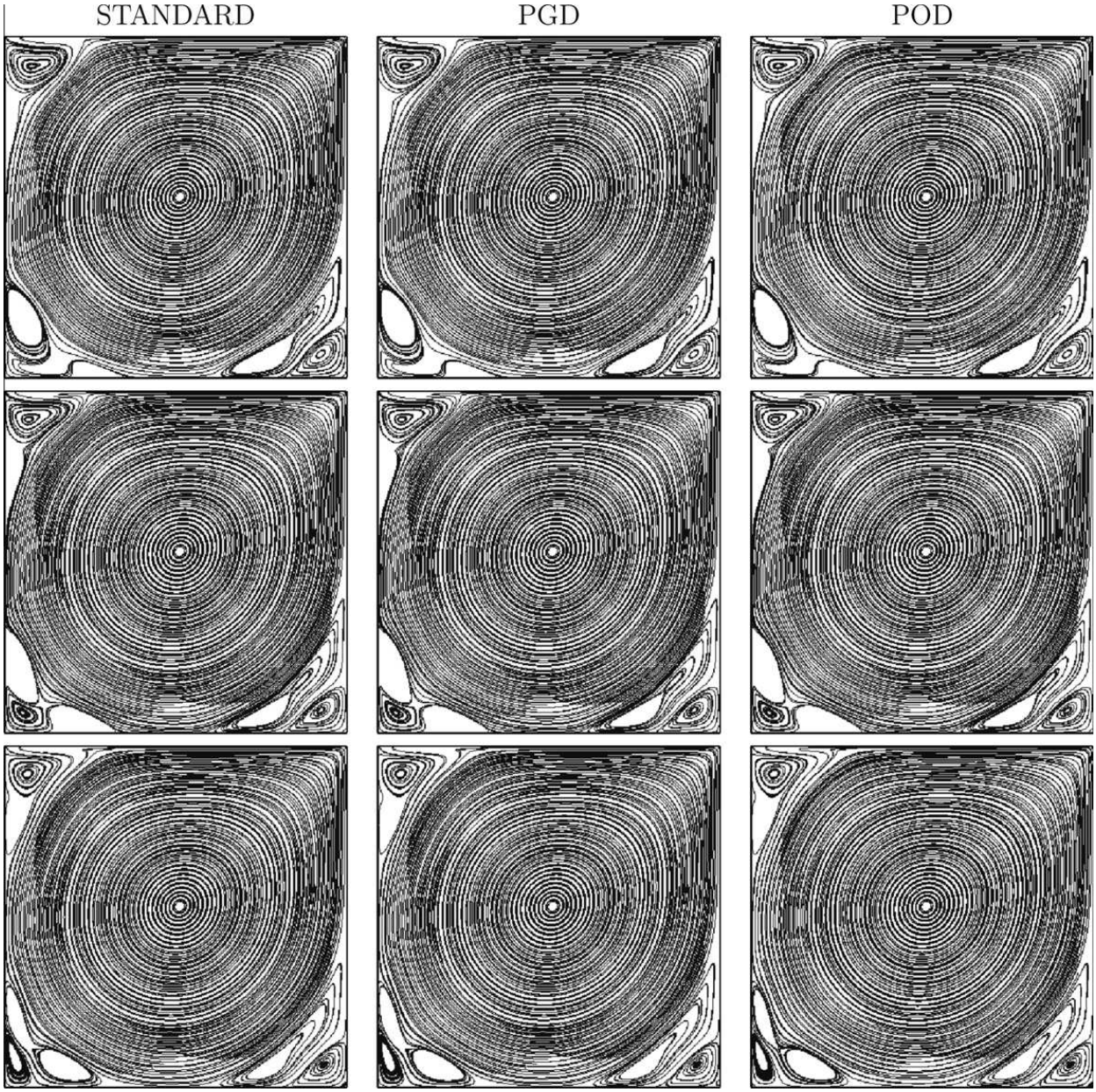


Fig. 19. Change in the stream function during one main period for Re = 10,000 on a 250 × 250 grid. From top to bottom for each solver (standard, PGD and POD) times t = 0, t = 0.338 and t = 0.676 are represented.

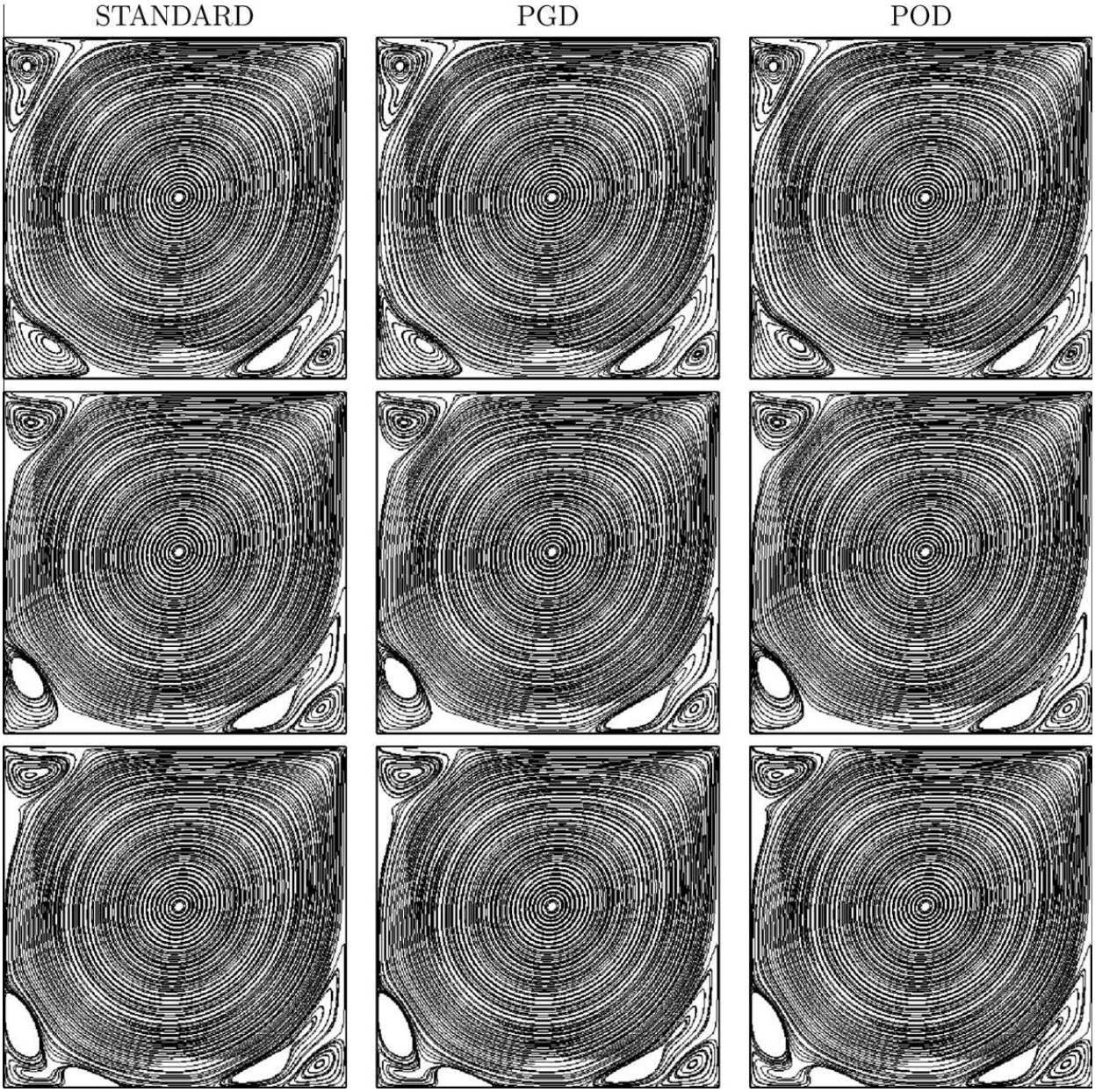


Fig. 20. Change in the stream function during one main period for Re = 10,000 on a 250 × 250 grid. From top to bottom for each solver (standard, PGD and POD) times t = 1.014, t = 1.352 and t = 1.69 are represented.

Saad in [35] using a second order model for the diffusion term and a third order upwind model for the convection term. The differences with the results of Sengupta et al. are not due to the PGD method but to the discretization models used since the results obtained with the standard model are very similar to those obtained using PGD.

The streamlines are also shown on Figs. 19 and 20 at different time points of the period and for each of the solvers studied. It is also noteworthy that there is a very good correspondence between the three models and the results obtained in [35]. For the PGD tensorial representation of solution, the firsts functions F_i and G_i for the x-velocity are shown on Fig. 21.

The results show that PGD provides results in aggrement with what expected, with a computation time divided by 6 compared with the standard model and that POD gives good results. The resolution of the reduced order POD is very fast, almost instantaneous. So, if the reduced order POD is sufficiently robust to predict flow over longer time periods than the sampling period, this method will be advantageous in terms of computation time. However, with certain applications, the POD-ROM methods requires an actualisation of the basis, and so, an adaptative control procedure. It should also be noted that the POD method requires stabilization, which is not the case for the PGD method.

The results of this part demonstrate that the computation of the lid-driven cavity with Re = 10,000 was performed efficiently with PGD. In fact, we were able to reproduce some results from the literature, like the behaviour of the vortex during one main period.

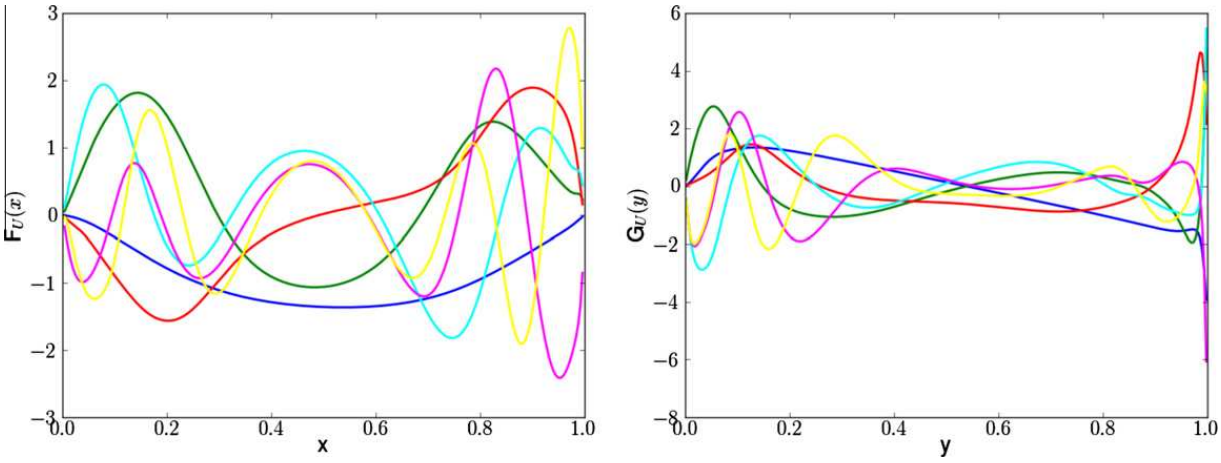


Fig. 21. Firsts functions of the tensor product computed with PGD that represent the x-velocity field for Re = 10,000.

5. Conclusion and further developments

This work is a first attempt to use the PGD method to solve many classical problems of fluid mechanics. We have demonstrated that PGD is able to solve the Burgers and Stokes equations accurately and with considerable time saving in CPU compared with the standard solver. Finally, for the classical 2D problem of the lid-driven cavity, we have shown that for three different Reynolds numbers (Re = 100, Re = 1000, and Re = 10,000), the PGD results are in agreement with those usually used to validate Navier–Stokes codes. For Re = 1000 (resp. Re = 10,000), PGD was eight (resp. six times) times faster than the full grid solver for standard discretization. To conclude, we have developed a Navier–Stokes solver with an optimum compromise between accuracy and CPU time. Increasing the Reynolds number beyond 10,000 is the subject of further development. It should be noted that this work is the first step in dealing with a 3D situation, where the problem is that 1000^3 degrees of freedom with a full description exceeds current computer storage possibilities. However, with PGD, successive enrichments involving 3 times 1000 degrees of freedom is entirely possible. In such a situation, the benefit of the method goes beyond simple saving of CPU and provides a feasible solution for otherwise intractable problems. For more standard discretisation, such as 200^3, we are now achieving better time-saving than for the 2D case, but this work is still in progress.

Acknowledgment

This work is supported by the French Poitou-Charentes region.

Appendix A. Algebraic formulation of progressive PGD with projection

A.1. Preliminaries

PGD is an iterative procedure which should be describe as follow:
At each iteration, the solution is enriched with an additional term

$$\alpha^{n+1}F^{n+1}(x)G^{n+1}(y).$$

The F^{n+1} and G^{n+1} functions are obtained by solving a small size non-linear problem ($N_x + N_y$), where N_x is the number of nodes in the direction x and N_y the number of nodes in the direction y. Then, $N + 1$ α^i coefficients are determined by solving a linear system of size ($N + 1$).

The PGD algorithm is summarized in three steps:

1. We assume we are at iteration $n + 1$: the first step consists in computing the new F^{n+1} and G^{n+1} functions, and will be referred to as the "***enrichment***" step (step $3 - 8$ in the algorithm described in Section 2.2).
2. Once these functions have been calculated, the $n + 1$ coefficients α^i have to be updated. This is the "***projection***" step (step 10 in the algorithm described in Section 2.2).
3. Finally, the convergence has to be checked. If the residual norm exceeds a given tolerance, enrichment is performed again, until convergence is achieved. This is the "***checking convergence***" step (step 11 in the algorithm described in Section 2.2).

The problem (3) can be written in a discrete form:

$$\mathcal{L}_h(U_h) = \mathcal{G}_h \tag{32}$$

with

$$\mathcal{L}_h = \sum_{j=1}^{n_{\mathcal{L}}} \mathbb{A}_x^j \otimes \mathbb{A}_y^j, \quad \mathcal{G}_h = \sum_{j=1}^{n_{\mathcal{G}}} \mathbf{f}_x^j \otimes \mathbf{f}_y^j, \quad U_h = \sum_{i=1}^{N} \alpha^i \mathbf{F}^i \otimes \mathbf{G}^i. \tag{33}$$

The operator $\mathcal{L}$ is discretized as a tensor product of operators $\mathbb{A}_x^j$ and $\mathbb{A}_y^j$ in the direction x and y, respectively. The discretized operator $\mathbb{A}_x^j$ (resp. $\mathbb{A}_y^j$) is a square matrix whose size is N_x (resp. N_y). The second term was decomposed as products of the sum of vectors $\mathbf{f}_x^j$ and $\mathbf{f}_y^j$ of size N_x and N_y. Finally the unknown U_h is calculated as a product sum of vectors $\mathbf{F}^i$ and $\mathbf{G}^i$ of size N_x and N_y using a weight coefficient α^i. $n_{\mathcal{L}}$ (resp. $n_{\mathcal{G}}$) represents the number of tensor products required to represent the separated form of the initial operator $\mathcal{L}$ (resp. the second member $\mathcal{G}$).

Taking into account the property of the tensor product, Eq. (32) can be written as:

$$\sum_{k=1}^{n_{\mathcal{L}}} \sum_{i=1}^{N} \alpha^i \left(\mathbb{A}_x^k \mathbf{F}^i \otimes \mathbb{A}_y^k \mathbf{G}^i \right) = \sum_{j=1}^{n_{\mathcal{G}}} \mathbf{f}_x^j \otimes \mathbf{f}_y^j. \tag{34}$$

The three steps of enrichment, projection and checking convergence will now be described with these notations.

A.2. The enrichment step

At this stage the unknown U_h can be written as follows with ($\alpha^{n+1} = 1$):

$$U_h = \sum_{i=1}^{n} \alpha^i \mathbf{F}^i \otimes \mathbf{G}^i + \mathbf{R} \otimes \mathbf{S}, \tag{35}$$

where $\mathbf{R}$ and $\mathbf{S}$ are unknowns. By introducing this new approximation of the solution into Eq. (34), we have to solve:

$$\sum_{k=1}^{n_{\mathcal{L}}} \left(\mathbb{A}_x^k \mathbf{R} \otimes \mathbb{A}_y^k \mathbf{S} \right) = \mathcal{G}_h - \sum_{k=1}^{n_{\mathcal{L}}} \sum_{i=1}^{N} \alpha^i \left(\mathbb{A}_x^k \mathbf{F}^i \otimes \mathbb{A}_y^k \mathbf{G}^i \right). \tag{36}$$

This non-linear system is solved within a fixed-point strategy.

In order to compute $\mathbf{R}$ we choose to fix $\mathbf{S}$ and we project Eq. (36) onto the vector $\mathbf{S}$. This gives the following problem, corresponding to Eq. (10) in continuous form:

$$\sum_{k=1}^{n_{\mathcal{L}}} \gamma_k^1 \mathbb{A}_x^k \mathbf{R} = \sum_{j=1}^{n_{\mathcal{G}}} \gamma_j^2 \mathbf{f}_x^j - \sum_{k=1}^{n_{\mathcal{L}}} \sum_{i=1}^{N} \gamma_{i,k}^3 \alpha^i \mathbb{A}_x^k \mathbf{F}^i \tag{37}$$

with

$$\gamma_k^1 = {}^t\mathbf{S} \mathbb{A}_y^k \mathbf{S} \in \mathbb{R}, \quad \gamma_j^2 = {}^t\mathbf{S} \mathbf{f}_y^j \in \mathbb{R}, \quad \gamma_{i,k}^3 = {}^t\mathbf{S} \mathbb{A}_y^k \mathbf{G}^i \in \mathbb{R}. \tag{38}$$

Similarly, in order to compute $\mathbf{S}$ we set $\mathbf{R}$ at the value just computed in Eq. (37) and we project Eq. (36) onto the vector $\mathbf{R}$. This gives the following problem, corresponding to Eq. (9) in continuous form:

$$\sum_{k=1}^{n_{\mathcal{L}}} \beta_k^1 \mathbb{A}_y^k \mathbf{S} = \sum_{j=1}^{n_{\mathcal{G}}} \beta_j^2 \mathbf{f}_y^j - \sum_{k=1}^{n_{\mathcal{L}}} \sum_{i=1}^{N} \beta_{i,k}^3 \alpha^i \mathbb{A}_y^k \mathbf{G}^i \tag{39}$$

with

$$\beta_k^1 = {}^t\mathbf{R} \mathbb{A}_x^k \mathbf{R} \in \mathbb{R}, \quad \beta_j^2 = {}^t\mathbf{R} \mathbf{f}_x^j \in \mathbb{R}, \quad \beta_{i,k}^3 = {}^t\mathbf{R} \mathbb{A}_x^k \mathbf{F}^i \in \mathbb{R}. \tag{40}$$

Problems (37) and (39) are solved iteratively. The fixed-point procedure stops when the kth iteration satisfies:

$$\| (\mathbf{R} \otimes \mathbf{S})_k - (\mathbf{R} \otimes \mathbf{S})_{k-1} \| \leqslant \epsilon, \tag{41}$$

where $\| \cdot \|$ is the L^2 norm and ϵ is a parameter chosen by the user. The new $\mathbf{F}^{N+1}$ and $\mathbf{G}^{N+1}$ are then given by the next normalization:

$$\mathbf{F}^{N+1} = \frac{\mathbf{R}}{\|\mathbf{R}\|}, \quad \mathbf{G}^{N+1} = \frac{\mathbf{S}}{\|\mathbf{S}\|} \tag{42}$$

A.3. Projection step

Assuming the $\mathbf{F}^i$ and $\mathbf{G}^i$ functions are known, α^i ($1 \leqslant i \leqslant N+1$) has to be computed. For this purpose, we project Eq. (36) onto $\mathbf{F}^j$ and $\mathbf{G}^j$. Thus the following linear problem, whose size is $(N+1)$, corresponding to Eq. (11) in continuous form, is obtained

$$\mathbb{H} \boldsymbol{\alpha} = \mathbf{J} \quad \text{with} \quad {}^t\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_{N+1}\}, \tag{43}$$

where the components of $\mathbb{H}$ and $\mathbf{J}$ are defined by:

$$\mathbb{H}_{ij} = \sum_{k=1}^{n_{\mathcal{L}}} {}^t\mathbf{F}^j \mathbb{A}_x^k \mathbf{F}^i \cdot {}^t\mathbf{G}^j \mathbb{A}_y^k \mathbf{G}^i \quad \text{and} \quad J_j = \sum_{k=1}^{n_{\mathcal{G}}} {}^t\mathbf{F}^j \mathbf{f}_x^k \cdot {}^t\mathbf{G}^j \mathbf{f}_y^k. \tag{44}$$

A.4. Check convergence

In order to estimate the convergence of the algorithm, a computation is performed of the residual *Res* of Eq. (3) defined by:

$$Res = \sum_{k=1}^{n_{\mathcal{L}}} \sum_{i=1}^{N+1} \alpha^i \left(\left[\mathbb{A}_x^k\right] \mathbf{F}^i \otimes \left[\mathbb{A}_y^k\right] \mathbf{G}^i \right) - \sum_{k=1}^{n_{\mathcal{G}}} \mathbf{f}_x^k \otimes \mathbf{f}_y^k. \tag{45}$$

When the L^2 norm of this residual becomes lower than a coefficient ϵ set by the user, the algorithm is considered to be at convergence, and the solution of the problem is expressed as:

$$U_h = \sum_{i=1}^{n+1} \alpha^i \mathbf{F}^i \otimes \mathbf{G}^i. \tag{46}$$

Appendix B. The proper orthogonal decomposition (POD)

The basic idea of POD consists in finding a "physical" basis which is optimal in terms of energy. Thus, a deterministic function ϕ is sought which gives the "best" representation of the set of flow fields u (assumed randomly and at real values) in the following sense:

$$\begin{cases} \overline{(u,\phi)^2} = \max\limits_{\psi \in L^2(\Omega)} \overline{(u,\psi)^2}, \\ (\phi,\phi) = 1, \end{cases} \tag{47}$$

where $(\bullet,\bullet)$ denotes the inner product of $L^2(\Omega)$ and $\overline{\bullet}$ denotes a statistic average operator. L^2 is the space of finite energy functions in the flow volume Ω.

From variational calculus it follows that the above expression is equivalent to the Fredholm integral

$$\text{Find } \lambda \in \mathbb{R} \text{ and } \phi \in L^2(\Omega) \text{ with } \int_\Omega R(x,x')\phi(x')dx' = \lambda\phi(x), \tag{48}$$

representing an eigenvalue problem for ϕ. R is the spatial correlation tensor defined, with the ergodicity hypothesis, by:

$$R(x,x') = \overline{u(x,t) \otimes u(x',t)}. \tag{49}$$

The eigenfunctions ϕ_n are orthogonal and all realizations of the flow u are written:

$$u(x,t) = \sum_{n=1}^{+\infty} a_n(t)\phi_n(x), \quad \text{in } L^2(\Omega) \text{ sense} \tag{50}$$

with $a_n(t) = (u(\bullet,t),\phi_n)$.

In practice, when the flow field is obtained by numerical simulation, the evaluation of the tensor R is a very large computational task. In order to reduce the calculation, we used the Snapshots method proposed in 1987 by Sirovich [37]. In this technique, based on the fact that the eigenfunctions can be expressed in terms of the original set of data:

$$\phi_n(x) = \sum_{k=1}^{M} u(x,t_k)A_{nk},$$

we must solve the matrix eigenvalue problem:

$$\sum_{k=1}^{M} C_{kj}A_{nk} = \lambda A_{nj}, \quad \text{for } j = 1,\ldots,M. \tag{51}$$

A_{nk} denotes the constants associated to the nth mode, M is the number of snapshots, and C is the temporal correlation tensor defined by:

$$C_{kj} = \frac{1}{M}\int_\Omega u_i(x,t_k)u_i(x,t_j)dx. \tag{52}$$

As C is symmetric and positive semi-definite, all eigenvalues are real and non-negative and can be ordered as $\lambda_1 \geqslant \lambda_2 \cdots \geqslant \lambda_M$. Each eigenvalue λ_n, taken individually, represents the energy contribution of the corresponding

eigenfunction. The eigenvectors ϕ_n are incompressible (e.g. their divergence is null) due to the way they are constructed, they satisfy the boundary conditions and can be normalized to form an orthonormal set.

B.1. Low-order dynamical system

Usually, the N first POD modes contain most of the energy. So we can expect that a low order dynamic system, obtained by the Galerkin projection of the Navier–Stokes equations onto the high energy ϕ_n eigenmodes, gives the flow dynamics accurately. The first step consists in approximating the flow field u by keeping the first N modes and ignoring the remaining modes. This approximation is written:

$$u(x,t) \simeq \overline{u} + \sum_{n=1}^{N} a_n(t)\phi_n(x). \tag{53}$$

The second step consists in performing a Galerkin projection. As the continuity equation for incompressible flow, $div\phi_n = 0$, is satisfied by each eigenfunction, here we considered only the dimensionless momentum equation of the flow. Introducing (Eq. (53)) into the dimensionless momentum equation of the flow, projected onto the spatial structures ϕ_n, and taking the ϕ orthonormal:

$$\frac{da_n}{dt} = \sum_{m=1}^{N}\sum_{k=1}^{N} C_{nmk}a_m a_k + \sum_{m=1}^{N} B_{nm}a_m + D_n + H_n \tag{54}$$

with $n = 1,\ldots,N$ and

$$\begin{cases} C_{nmk} = -(\phi_n, \nabla\phi_m \cdot \phi_k), & B_{nm} = \left(\phi_n, \frac{1}{Re}\Delta\phi_m - \nabla\overline{u}\cdot\phi_m - \nabla\phi_m\cdot\overline{u}\right) \\ D_n = -\int_\Gamma p\phi_n \mathbf{n}\, d\Gamma, & H_n = \left(\phi_n, -\nabla\overline{p} + \frac{1}{Re}\Delta\overline{u} - \nabla\overline{u}\cdot\overline{u}\right), \end{cases}$$

where $\mathbf{n}$ is the outward normal on the domain Ω considered of boundary Γ.

The previous system of equations includes a term D_n that is linked to the pressure P. However there is no simple expression of this term according to the expansion coefficients ϕ_n. Some techniques can be used to avoid this term. In the case of the lid driven cavity, this term is equal to zero.

References

[1] C. Allery, S. Guerin, A. Hamdouni, A. Sakout, Experimental and numerical pod study of the coanda effect used to reduce self-sustained tones, Mechanics Research Communications 31 (1) (2004) 105–120.
[2] C. Allery, C. Beghein, A. Hamdouni, On investigation of particle dispersion by a pod approach, International Applied Mechanics 44 (2008) 133–142.
[3] E. Liberge, A. Hamdouni, Reduced order modelling method via proper orthogonal decomposition (pod) for flow around an oscillating cylinder, Journal of Fluids and Structures 26 (2) (2010) 292–311.
[4] T.K. Sengupta, V. Suman, N. Singh, Solving Navier–Stokes equation for flow past cylinders using single-block structured and overset grids, Journal of Computational Physics 229 (1) (2010) 178–199.
[5] A. Rajabpour, F. Kowsary, V. Esfahanian, Reduction of the computational time and noise filtration in the ihcp by using the proper orthogonal decomposition (pod) method, International Communications in Heat and Mass Transfer 35 (8) (2008) 1024–1031.
[6] Z. Luo, J. Zhu, R. Wang, I. Navon, Proper orthogonal decomposition approach and error estimation of mixed finite element methods for the tropical pacific ocean reduced gravity model, Computer Methods in Applied Mechanics and Engineering 196 (41–44) (2007) 4184–4195.
[7] U. Galvanetto, G. Violaris, Numerical investigation of a new damage detection method based on proper orthogonal decomposition, Mechanical Systems and Signal Processing 21 (3) (2007) 1346–1361.
[8] X. Amandolese, C. Cremona, Analysing fluid loadings on moving bluff bodies using proper orthogonal decomposition, Journal of Fluids and Structures 20 (4) (2005) 577–587. bluff-Body/Flow Interactions.
[9] T. Sengupta, S. Dey, Proper orthogonal decomposition of direct numerical simulation data of by-pass transition, International Journal of Numerical Methods in Engineering 82 (2004) 2693–2703.
[10] J. Burkardt, M. Gunzburger, H.-C. Lee, Pod and cvt-based reduced-order modeling of Navier–Stokes flows, Computer Methods in Applied Mechanics and Engineering 196 (1–3) (2006) 337–355.
[11] Q. Du, M. Gunzburger, Grid generation and optimization based on centroidal Voronoi tessellations, Applied Mathematics and Computation 133 (2–3) (2002) 591–607.
[12] N. Verdon, C. Allery, C. Béghein, A. Hamdouni, D. Ryckelynck, Reduced-order modelling for solving linear and non-linear equations, International Journal for Numerical Methods in Biomedical Engineering (2009), doi:10.1002/cnm.1286.
[13] A. Ammar, D. Ryckelynck, F. Chinesta, R. Keunings, On the reduction of kinetic theory models related to finitely extensible dumbbells, Journal of Non-Newtonian Fluid Mechanics 134 (2006) 136–147.
[14] D. Ryckelynck, A priori hyperreduction method: an adaptive approach, Journal of Computational Physics 202 (2005) 346–366.
[15] D. Ryckelynck, Reduction a priori de modeles thermomecaniques, Comptes Rendus Mecanique 330 (7) (2002) 499–505.
[16] D. Ryckelynck, L. Hermanns, F. Chinesta, E. Alarcon, An efficient a priori model reduction for boundary element models, Engineering Analysis with Boundary Elements 29 (29) (2005) 796–801.
[17] A. Ammar, F. Chinesta, P. Diez, A. Huerta, An error estimator for separated representations of highly multidimensional models, Computer Methods in Applied Mechanics and Engineering 199 (2010) 1872–1880.
[18] A. Ammar, F. Chinesta, A. Falcó, On the convergence of a Greedy rank-one update algorithm for a class of linear systems, Archives of Computational Methods in Engineering 17 (4) (2010) 473–486.
[19] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids, Journal of Non-Newtonian Fluid Mechanics 139 (3) (2006) 153–176.
[20] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids: Part II: Transient simulation using space-time separated representations, Journal of Non-Newtonian Fluid Mechanics 144 (2–3) (2007) 98–121.

[21] F. Chinesta, A. Ammar, P. Joyot, The nanometric and micrometric scales of the structure and mechanics of materials revisited: an introduction to he challenges of fully deterministic numerical descriptions, International Journal for Multiscale Computational Engineering 6/3 (2008) 191–213.
[22] A. Nouy, O.P.L. Maitre, Generalized spectral decomposition for stochastic nonlinear problems, Journal of Computational Physics 228 (1) (2009) 202–235.
[23] A. Nouy, A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations, Computer Methods in Applied Mechanics and Engineering 196 (45–48) (2007) 4521–4537.
[24] P. Ladeveze, A. Nouy, On a multiscale computational strategy with time and space homogenization for structural mechanics, Computer Methods in Applied Mechanics and Engineering 192 (28–30) (2003) 3061–3087. multiscale Computational Mechanics for Materials and Structures.
[25] A. Ammar, M. Normandin, F. Daim, D. Gonzalez, E. Cueto, F. Chinesta, Non incremental strategies based on separated representations: applications in computational rheology, Communications in Mathematical Sciences 8 (3) (2010) 671–695.
[26] P. Ladeveze, J.-C. Passieux, D. Neron, The latin multiscale computational method and the proper generalized decomposition, Computer Methods in Applied Mechanics and Engineering 199 (21–22) (2010) 1287–1296. multiscale Models and Mathematical Aspects in Solid and Fluid Mechanics.
[27] A. Ammar, F. Chinesta, E. Cueto, Coupling finite elements and separated representation, International Journal of Multiscale Computational Engineering, in press.
[28] F. Chinesta, A. Ammar, E. Cueto, Proper generalized decomposition of multiscale models, International Journal of Numerical Methods in Engineering 83 (8–9) (2010) 1114–1132.
[29] D. Gonzalez, A. Ammar, F. Chinesta, E. Cueto, Recent advances in the use of separated representations, International Journal of Numerical Methods in Engineering 81 (5) (2010) 637–659.
[30] A. Nouy, A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations, Computer Methods in Applied Mechanics and Engineering 199 (23–24) (2010) 1603–1625.
[31] J. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Computer Methods in Applied Mechanics and Engineering 195 (44–47) (2006) 6011–6045.
[32] A. Quarteroni, F. Saleri, A. Veneziani, Factorization methods for the numerical approximation of Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 188 (1–3) (2000) 505–526.
[33] J. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, third ed., Springer, Berlin, 2002, 423p.
[34] U. Ghia, K. Ghia, C. Shin, High-re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method, Journal of Computational Physics 48 (1982) 387–411.
[35] C.-H. Bruneau, M. Saad, The 2d lid-driven cavity problem revisited, Computers and Fluids 35 (3) (2006) 326–348.
[36] T. Sengupta, V. Vijay, S. Bhaumik, Further improvement and analysis of ccd scheme: dissipation discretization and de-aliasing properties, Journal of Computational Physics 228 (17) (2009) 6150–6168.
[37] L. Sirovich, Turbulence and the dynamics of coherent structures. Part 1: Coherent structures. Part 2: Symmetries and transformations. Part 3: Dynamics and scaling, Quarterly of Applied Mechanics 45 (1987) 561–590.