Proper Generalized Decomposition method for incompressible Navier-Stokes equations with a spectral discretization

Antoine Dumon, Cyrille Allery, Amine Ammar

To cite this version:

HAL Id: hal-01062396
https://hal.archives-ouvertes.fr/hal-01062396
Submitted on 4 May 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Science Arts & Métiers (SAM) is an open access repository that collects the work of Arts et Métiers ParisTech researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: http://sam.ensam.eu
Handle ID: http://hdl.handle.net/10985/8487

To cite this version :

Any correspondence concerning this service should be sent to the repository Administrator: archiveouverte@ensam.eu
Proper Generalized Decomposition method for incompressible Navier–Stokes equations with a spectral discretization

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

LaSIE FRE-CNRS 3474, Pôle Sciences et Technologie, Avenue Michel Crepeau, 17042 La Rochelle Cedex 1, France
Arts et Metiers ParisTech, 2 Bvd du Ronceray, 49035 Angers Cedex 01, France

ARTICLE INFO

Keywords:
- Reduced order model
- Proper Generalized Decomposition
- Incompressible flow
- Spectral discretization

ABSTRACT

Proper Generalized Decomposition (PGD) is a method which consists in looking for the solution to a problem in a separate form. This approach has been increasingly used over the last few years to solve mathematical problems. The originality of this work consists in the association of PGD with a spectral collocation method to solve transfer equations as well as Navier–Stokes equations. In the first stage, the PGD method and its association with spectral discretization is detailed. This approach was tested for several problems: the Poisson equation, the Darcy problem, Navier–Stokes equations (the Taylor Green problem and the lid-driven cavity). In the Navier–Stokes problems, the coupling between velocity and pressure was performed using a fractional step scheme and a $P_N-P_{N+2}$ discretization. For all problems considered, the results from PGD simulations were compared with those obtained by a standard solver and/or with the results found in the literature. The simulations performed showed that PGD is as accurate as standard solvers. PGD preserves the spectral behavior of the errors in velocity and pressure when the time step or the space step decreases. Moreover, for a given number of discretization nodes, PGD is faster than the standard solvers.

1. Introduction

Over the last few decades, many approaches have been used to solve transfer equations. Some problems are related to very large systems that cannot be easily solved numerically. Therefore, traditional methods require very long computation times and a large storage capacity. One way to reduce computation time and memory storage when solving partial differential equations (PDE) consists in using reduced order models (ROM). ROM have been used extensively in recent decades. They consist in approximating the solution $f$ of the PDE as follows

$$f(x, t) \approx f_m(x, t) = \sum_{k=1}^{m} a_k(t) \Phi_k(x).$$

where $x$ is the spatial coordinates vector and $\Phi_k(x)$ is a low-dimensional reduced basis. $m$ is the reduced-basis size which is usually much smaller than the full grid size of the discretized solution. The coefficients $a_k(t)$ are solutions for a very low-order system obtained by a Galerkin projection of the initial equation of the problem over this basis. The difference with ROM lies in the way in which the reduced basis is computed.

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

http://dx.doi.org/10.1016/j.amc.2013.02.022
Various reduced bases can be used to reduce the order of models, including the Lagrange, Hermite, Taylor or POD basis. In fluid mechanics, the most popular reduced-order modeling technique is POD (Proper Orthogonal Decomposition). This method has been used to predict the dynamics of a flow [1–3], to control flows [4–7], or to compute particle dispersion in a flow [8,9]. Note that an “a posteriori” error estimation is available for the reduced basis solution of parametrized linear parabolic partial differential equations [10], and of parametrized Stokes [11] and Navier–Stokes [12–14] equations. To build these bases, these approaches often require some snapshots of the flow, which may require significant computation time. In order to circumvent this drawback, “a priori” model reduction techniques were developed. These methods consist in constructing a reduced basis without “a priori” knowledge of the solution.

The first is the A Priori model Reduction (APR) which has been the subject of several developments [15–18]. In 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 of the equation resolution.

The second, which was applied here, is Proper Generalized Decomposition (PGD). PGD consists in looking for the solution of a problem in the following separated form:

\[
f(x_1, \ldots, x_N) = \sum_{i=1}^{Q} \prod_{k=1}^{N_i} F_{ik}(x_k) = \sum_{i=1}^{Q} F_{i1}(x_1)F_{i2}(x_2) \cdots F_{iN}(x_N)
\]

\((x_i)\) 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 the unknowns involved in the solution is \(Q \cdot N \cdot M\) instead of the \(M^N\) degrees of freedom involved in standard techniques. In most cases, when the field is sufficiently regular, the number of terms \(Q\) in the finite sum is usually quite small (a few dozen) and in all cases the approximation converges towards the solution of the full grid description (see [19–21]). The method was originally proposed in the nineties by Ladeveze et al. as one of the main ingredients of the LATIN method (see for example [22,23]). In this framework, the approach is based on a space–time representation and is called “radial approximation”. This space–time representation was also used by Nouy [24,25] to solve stochastic problems (in this context, the method is called “Generalized Spectral Decomposition”) a few years ago. Ammar et al. extended PGD to multidimensional decomposition to solve models of polymeric systems [26–28]. Since this work, the PGD was also applied to solve quantum chemistry problems [29], 3D elastic problems [30], fluid problems [31,32] and structural optimization problems [33]. Finally, a review of the method can be found in [34,35]. PGD is usually restricted to a rectangular domain for a best performance of the tensorial decomposition. However it can be used in a non rectangular domain [36]. In this case the non rectangular domain is immersed in a rectangular one and the initial operator is associated with a conjugate operator describing the boundaries.

In order to treat transient problems, PGD could be applied using one of the three following decompositions.

- The first consists in a time space decomposition: \(f(t, x) = \sum_{i=1}^{Q} F_i(t)F_i(x)\). We are directly looking for a time/space solution. The main drawback encountered here is that a full grid description is required to define the functions over the physical space.

- To circumvent this difficulty, the second possibility consists in writing a full decomposition involving the two dimensions of the physical space: \(f(t, x, y) = \sum_{i=1}^{Q} F_i(t)F_i(x)F_i(y)\). 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 this work, consists in keeping the incremental approach and performing the separation over the physical space at each time step: \(f(t, x, y) = \sum_{i=1}^{Q} F_i(t)F_i(x)F_i(y)\).

The PGD approach is usually associated with a finite-difference, finite-element or finite volume discretization. In some situations, such as turbulence flows, these discretization methods are not accurate enough to simulate the relevant physical phenomena. This drawback cannot be easily solved using a mesh refinement due to the memory storage limit. To avoid this, it is possible to use spectral methods. Therefore, in this paper, we coupled the PGD method with a spectral discretization to solve the Navier–Stokes equations. The aim was to show numerically that the PGD method, while being faster than the standard method, does not affect the accuracy of the solution and preserves the benefits of spectral discretization. It should be noted that Nouy [37] has already used a spectral discretization (Spectral Element Method) with PGD to solve an advection–diffusion problem. The originality of this work is the extension of the approach to solve the Navier–Stokes equations, which leads to some difficulties. In fact, the major difficulty in solving incompressible flows is that the velocity and the pressure are coupled with the incompressibility constraints. This is achieved by using a fractional step scheme. A second difficulty is related to the choice of the approximation spaces associated with velocity and pressure. These spaces must satisfy the inf-sup condition [38]. One possibility consists in approximating the pressure with polynomials of a degree lower than those approximating the velocity (with a difference equal to two between the two degrees). This method is called the \(P_N-P_{N-2}\) formulation [39–41].

It should be remembered that the originality of this work is the extension of the PGD approach to solve the Navier–Stokes equations. In fact, in this work, the spectral formalism associated with the PGD method is detailed and the method is used to solve the Navier–Stokes equations through a fractional step scheme combined with the \(P_N-P_{N-2}\) formulation.

This paper is organized as follows. First, the PGD method will be presented in a general framework. Secondly, spectral discretization will be detailed and its association with PGD performed on a Poisson equation. Then, the discretization of
the Navier–Stokes equations with a \( P_N-P_{N-2} \) method associated with a fractional step scheme and its PGD formulation is described. Finally, the results on the test cases of the Darcy and Taylor–Green problems and the 2D lid-driven cavity in stationary case are detailed.

2. PGD and spectral formulation

2.1. Description of PGD

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

\[
\begin{align*}
\text{Find } U(x, y) & \text{ as: } \\
\mathcal{L}(U) & = G \\
\text{in } \Omega \\
\text{+ Boundary Conditions}
\end{align*}
\]

where \( \mathcal{L} \) is a linear differential operator and \( G \) 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:

\[
U(x, y) \approx U_m(x, y) = \sum_{i=1}^{m} \tilde{a}^i F^i(x) G^i(y),
\]

where \( U_m(x, y) \) is the approximation of the solution of order \( m \). The coefficients \( \tilde{a}^i \in \mathbb{R} \) are the weighting factors. At each iteration, the solution is enriched with an additional term \( a^{m+1} F^{m+1}(x) G^{m+1}(y) \). In the following, an iteration of the PGD algorithm will consist in constructing the solution of order \( m + 1 \) from the solution of order \( m \) by applying the following three steps:

1. Enrichment step: At the \( m \) stage, the solution approximation of order \( m - 1 \) is taken as known. In this step we look for the new product of functions \( F^m(x) G^m(y) \) such as:

\[
U_m(x, y) = \sum_{i=1}^{m-1} \tilde{a}^i F^i(x) G^i(y) + F^m(x) G^m(y).
\]

Introducing this expression into (2), results in:

\[
\mathcal{L}(U_{m-1} + F^m G^m) = G + \text{Res}^m,
\]

where \( \text{Res}^m \), defined by

\[
\text{Res}^m = \mathcal{L}(U_m) - G
\]

is a residual, due to the fact that Eq. (4) is an approximation of the solution. To determine \( F^m \) and \( G^m \), Eq. (5) is projected onto each of the unknowns \( F^m \) and \( G^m \). Furthermore, the residual must be orthogonal to the \( F^m \) and \( G^m \) functions. Thus, we obtain the following applications:

- The application \( S_m : X \rightarrow Y \) that associates a function dependent on \( x, F^m \in X \), to a function dependent on \( y, G^m \in Y \) which is given by:

\[
\langle \mathcal{L}(F^m(x) G^m(y)), F^m \rangle_{L^2(X)} = \langle G - \mathcal{L}(U_{m-1}(x, y)), F^m \rangle_{L^2(X)},
\]

- The application \( T_m : Y \rightarrow X \) that associates a function dependent on \( y, G^m \in Y \), to a function dependent on \( x, F^m \in X \) which is given by:

\[
\langle \mathcal{L}(F^m(x) G^m(y)), G^m \rangle_{L^2(Y)} = \langle G - \mathcal{L}(U_{m-1}(x, y)), G^m \rangle_{L^2(Y)}
\]

where \( \langle \cdot, \cdot \rangle_{L^2(X)} \) and \( \langle \cdot, \cdot \rangle_{L^2(Y)} \) are the scalar products on \( L^2 \), in the \( x \) and \( y \) directions.

In order to obtain the new functions \( F^m \) and \( G^m \), Eq. (7) and (8) must be solved simultaneously. This was achieved using the standard fixed point algorithm detailed in Table 1. There are some restrictions for the convergence of the fixed point method. If the operators are symmetrical, convergence is guaranteed. If not, a preconditioned symmetrical problem is built up by multiplying the initial operator (and the second member) by its transposed form. The initialisation of the fixed point algorithm is also an important factor for the convergence of the method. In this study, the functions were randomly initialized and satisfied the boundary conditions.

2. Projection step: In order to increase the accuracy of the decomposition, the \( m \) \( \tilde{a}^i \) coefficients are now sought in such a way that the residual \( \text{Res}^m \) is orthogonal to each product of functions \( F^i G^i \). The \( \tilde{a}^i \) coefficients are then solutions of the following systems of equations:

\[
\left\langle \mathcal{L} \left( \sum_{i=1}^{m} \tilde{a}^i F^i(x) G^i(y) \right), F^k G^k \right\rangle_{L^2(\Omega)} = \langle G, F^k G^k \rangle_{L^2(\Omega)} \text{ for } 1 \leq k \leq m
\]
We denote $W : (X)^m \times (Y)^m \rightarrow \mathbb{R}^m$ as the application that associates functions $F^m = \{F_i\}_{i=1}^m$ and $G^m = \{G_i\}_{i=1}^m$ to the vector $\Lambda_m = \{x_i\}_{i=1}^m \in \mathbb{R}^m$. Its expression is given by (9).

3. Checking the convergence step:

If the $L^2$ norm of the residual defined by Eq. (6) is lower than the coefficient $\epsilon$ set by the user, the PGD algorithm is converged. Otherwise, one more iteration is needed, and the enrichment and projection steps are repeated taking $m = m + 1$ until convergence is obtained.

The PGD algorithm\(^1\) is summarized in Table 2. In this work, the PGD algorithm was achieved by performing a discretization of the three previous steps on Tchebychev–Gauss–Lobato collocation points, which is detailed in the following section.

2.2. Spectral discretization

We defined the resolution domain $\Omega = [-1, 1] \times [-1, 1]$. It is noteworthy that a simple affine transformation can deal with any rectangular 2D domain $[a, b] \times [a', b']$. The domain $\Omega$ is discretized using the Tchebychev–Gauss–Lobato grid defined as:

$$
x_i = \cos(i \frac{\pi}{N_x}) \quad \text{for} \quad i = 0, \ldots, N_x
\quad y_j = \cos(j \frac{\pi}{N_y}) \quad \text{for} \quad j = 0, \ldots, N_y
$$

(10)

where $N_x$ and $N_y$ are the maximum indexes of collocation points in the $x$ and $y$ direction.

The PGD approximation of the solution of order $m$ is given by:

$$
U_m(x, y) = \sum_{k=1}^{m} \alpha^k F^k(x) G^k(y)
$$

(11)

where $F^k(x)$ (respectively $G^k(y)$) are approximated by polynomials of order $N_x$ on the $(N_x + 1)$ nodes $x_i$ (respectively, polynomials of order $N_y$ on the $(N_y + 1)$ nodes $y_j$) of the grid. Then,

$$
F^k(x) = \sum_{l=0}^{N_x} F_l(x_i) h_l(x) \quad \text{and} \quad G^k(y) = \sum_{l=0}^{N_y} G_l(y_j) f_l(y)
$$

(12)

$h_l(x)$ (respectively $f_l(y)$) is the Lagrange interpolation polynomial of degree $N_x$ (resp. $N_y$) associated with the collocation point $x_i$ (resp. $y_j$). This is given by:

$$
h_l(x) = \prod_{j=0, j \neq l}^{N_x} \frac{x - x_j}{x_i - x_j} = \frac{(-1)^{l-1} (1 - x^2) T_{N_x}(x)}{2N_x^2(x - x_i)}
$$

(13)

where $T_{N_x}(x) = \cos(N_x \arccos(x))$ is the Tchebychev polynomial of degree $N_x$ and $\tau_l = \frac{1}{2}$ if $l = 0$ or $l = N_x$, and $\tau_l = 1$ otherwise. $f_l(y)$ is given by the same expression by replacing $x$ by $y$.

The differentiation of $U_m(x, y)$ in $x$-direction at any collocation point $(x_i, y_j)$ can be written

$$
\frac{\partial U_m}{\partial x}(x_i, y_j) = \sum_{k=1}^{m} \alpha^k \left( \sum_{l=0}^{N_x} \partial_{x_i} F^l(x_i) \right) G^k(y_j)
$$

(14)

The components $\partial_{x_i}^N$ of the $\partial^N$ matrix correspond to $h_l(x_i)$ and are given by:

\(^1\) This formulation of the PGD is called “progressive PGD with projection”. Other formulations exist (for more details, see [37]).
Table 2
PGD algorithm ($\epsilon$ is a parameter fixed by the user).

1. for $m = 1$, $m_{\text{max}}$ do
   2. Perform step 1. to 7. of fixed point algorithm (see Table 1)
   3. Let $f^m = (f^{m-1}, f^m)$ and $g^m = (g^{m-1}, g^m)$
   4. Computation of $A_m = \{x|f^m\} = W(F^m, G^m)$
   5. Let $U_m = \sum_{i=1}^{N} x^i y^i G^m(y)$ and check convergence for $U$
      \(|\|\text{Res}^m||_{(x_0, y_0)} \leq \epsilon\)
   6. end for

$$
\begin{align}
\nabla^2 U_{m} &= -\frac{\bar{c}_l}{2N} \frac{(-1)^{i+j}}{\sin(\frac{\pi}{2N})} \sin(\frac{\pi}{2N} l) \quad 0 \leq i, j \leq N_x, \ i \neq j \\
\nabla^2 U_{m} &= -\frac{\cos \frac{\pi}{N} y}{2 \sin^2 \frac{\pi}{N}} \quad 1 \leq i \leq N_x - 1 \\

\Delta_{ij} &= -\Delta_{ij} X_{ij} = \frac{2N_x^2 + 1}{6} \\
\end{align}
$$

(15)

The second derivative of $U_m$ is performed by

$$
\frac{\partial^2 U_m}{\partial x^2}(x_i, y_j) = \sum_{k=1}^{m} \gamma_k \left( \sum_{i=0}^{N_y} \Delta_{ij} X_{ij} G^m(y_j) \right)
$$

(16)

where $\Delta_{ij} X_{ij} = (\nabla^2 U_{m})^2$.

The differentiation of $U_m(x, y)$ in $y$-direction is obtained in the same way.

3. Application: 2D Poisson problem

3.1. The PGD problem

The following section provides an illustration of PGD associated with a spectral discretization on the 2D Poisson problem. We looked for $U(x, y) \in \Omega$ such that,

\[
\begin{align}
\nabla^2 U(x, y) &= g_1(x, y) \quad \text{in } \Omega = [0, 1] \times [0, 1] \\
U(x, y) &= g_2(x, y) \quad \text{on } \partial \Omega \\
\end{align}
\]

(17)

where $\partial \Omega$ is the boundary of the domain $\Omega$.

After discretization by a spectral formulation, Eq. (17) could be written for $i = 0, \ldots, N_x$ and $j = 0, \ldots, N_y$:

$$
\sum_{i=1}^{N_x-1} \Delta_{ij} X_{ij} U(x_i, y_j) + \sum_{i=1}^{N_y-1} \Delta_{ij} X_{ij} U(x_i, y_j) = SM_y
$$

(18)

where

$$
SM_y = g_1(x_i, y_j) - \Delta_{ij} X_{ij} g_2(x_i, y_j) - \Delta_{ij} X_{ij} g_2(x_i, y_j) - \Delta_{ij} X_{ij} g_2(x_i, y_j) - \Delta_{ij} X_{ij} g_2(x_i, y_j)
$$

(19)

Using the PGD method, the solution was sought in the separated form described in Eq. (3) at the collocation points $(x_i, y_j)$.

In this problem, the three steps of the PGD method described in Section (2.1) are written:

1. Enrichment step: The new terms $F^m(x_i)$ and $G^m(y_j)$ (for $1 \leq i \leq N_x - 1$ and $1 \leq j \leq N_y - 1$) are the solutions to the non linear problem defined by:

$$
\langle A_0(F^m, G^m), F^m_{l}(x) \rangle = \langle SM_y, F^m_{l}(x) \rangle \\
\langle A_0(F^m, G^m), G^m_{l}(y) \rangle = \langle SM_y, G^m_{l}(y) \rangle
$$

(20)

where

$$
A_0(F^m, G^m) = \left( \sum_{l=1}^{N_x-1} \Delta_{ij} X_{ij} F^m(x_i) \right) G^m(y_j) + F^m(x_i) \left( \sum_{l=1}^{N_y-1} \Delta_{ij} X_{ij} G^m(y_j) \right)
$$

(21)

This system was solved using a fixed point method.

2. Projection step: The $m$ first functions $F^k$ and $G^k$, and the $m$ weighting factors $\alpha^k$ are given by solving the following linear system:
\[ \mathbb{J} = \mathbf{J} \quad \text{with} \quad ^{t}\mathbf{z} = (z_{1}, \ldots, z_{m}) \] (22)

The components of \( \mathbb{J} \) and \( \mathbf{J} \) are defined (for \( 1 \leq k, \quad q \leq m \) and for \( i = 0, \ldots, N_{x} \) and \( j = 0, \ldots, N_{y} \)) by:

\[
\mathbb{J}_{eq} = \sum_{i=1}^{N_{x}} \sum_{j=1}^{N_{y}} A_{ij}(F^{k}, G^{q})F^{i}(x_{i})G^{q}(y_{j})
\] (23)

and

\[
J_{q} = \sum_{i=1}^{N_{x}} \sum_{j=1}^{N_{y}} SM_{ij}F^{i}(x_{i})G^{q}(y_{j})
\] (24)

3. Checking the convergence: If the \( L^{2} \) norm of the residual defined (for \( i = 0, \ldots, N_{x} \) and \( j = 0, \ldots, N_{y} \)) by

\[
\sum_{k=1}^{m} \psi_{k} A_{ij}(F^{k}, G^{q}) - SM_{ij}
\] (25)

is lower than the coefficient \( \epsilon \) set by the user, the algorithm is converged. Otherwise, one more iteration at least is needed, and the enrichment and projection steps are repeated taking \( m = m + 1 \) until convergence is achieved.

3.2. Numerical example

Here, we will consider two different test cases with a given exact solution.

**Problem 1.** Find \( U(x, y) \in \Omega = [0, 1] \times [0, 1] \) solution of the problem (17) such that

\[
g_{1}(x, y) = -8\pi^{2} \cos(2\pi x) \sin(2\pi y) \quad \text{and} \quad \begin{cases} g_{2}(x, 0) = g_{2}(x, 1) = 0 \\
g_{2}(0, y) = g_{2}(1, y) = \sin(2\pi y) \end{cases}
\]

The analytical solution is then given by: \( U_{\text{ana}}(x, y) = \cos(2\pi x) \sin(2\pi y) \).

**Problem 2.** Find \( U(x, y) \in \Omega = [0, 1] \times [0, 1] \) solution of the problem (17) such that

\[
g_{1}(x, y)(x, y) = 12(x^{2}y^{4} + x^{4}y^{2}) - 8\pi^{2} \cos(2\pi x) \sin(2\pi y) \quad \text{and} \quad \begin{cases} g_{2}(x, 0) = g_{2}(x, 1) = 0 \\
g_{2}(0, y) = g_{2}(1, y) = \sin(2\pi y) \\
g_{2}(x, 1) = x^{4} \\
g_{2}(1, y) = y^{4} + \sin(2\pi y) \end{cases}
\]

The analytical solution is then given by: \( U_{\text{ana}}(x, y) = x^{4}y^{4} + \cos(2\pi x) \sin(2\pi y) \).

These problems were solved with the PGD algorithm detailed previously as well as with the full grid spectral standard solver in order to compare accuracy and performance. The number of nodes in each direction is denoted by \( N \). The convergence coefficient of PGD algorithm \( \epsilon \) was taken to be equal to \( 10^{-13} \) and the convergence parameter for the fixed point algorithm \( \eta \) was set at \( 10^{-9} \) with a maximum iteration value of \( k_{\text{max}} = 50 \). The error between the analytical solution and the computed solution obtained by PGD or by the full grid solver, plotted on Fig. 1, was defined by

\[ E = \frac{\|U - U_{\text{ana}}\|_{L^{2}(\Omega)}}{\|U_{\text{ana}}\|_{L^{2}(\Omega)}} \] (26)

We observed that the PGD solver was as accurate as the full grid solver and that the machine error was reached for \( N = 20 \). In addition, the error had an exponential convergence depending on the number of nodes, which is typical in spectral discretization.

For Problem 1, PGD required only one function (\( m = 1 \)) to converge, while sixty functions (\( m = 60 \)) were required for Problem 2. In both cases, PGD was much faster than the full grid solver. For Problem 1, with a mesh size of \( 64 \times 64 \), PGD was around thousand times faster than the full grid solver. This is because the error of the full grid solver is reached with only one PGD function product. For Problem 2, although the PGD method required sixty function products, the gain in computation time was also significant. Indeed, for a mesh size of \( 64 \times 64 \), PGD was around one hundred times faster than the full grid solver. For PGD, the cost of preparing the decomposition and computing the bases (steps 1–3) is included in the comparison of the computational costs.

In order to study the influence of the various PGD parameters (\( \epsilon, \eta, k_{\text{max}} \)) on the solution obtained, the following error was introduced:

\[ E_{m} = \frac{\|U_{m} - U_{F}\|_{L^{2}(\Omega)}}{\|U_{F}\|_{L^{2}(\Omega)}} \] (27)

where \( U_{F} \) denotes the reference solution computed with the full grid solver, and \( U_{m} \) is the PGD solution obtained with \( m \) products of functions (see Eq. (3)). Only Problem 2 was considered.²

² This study was not possible for Problem 1 because only one couple of functions provides the solution.
For each enrichment \( m \) and for various values of \( k_{\text{max}} \), the number of iterations required for the convergence of the fixed point algorithm is illustrated in Fig. (2a). It should be remembered that the parameter \( k_{\text{max}} \) corresponds to the maximum number of iterations set by the user in the fixed point algorithm. This figure shows that for \( k_{\text{max}} = 2, 5, 10 \), the fixed point algorithm never converged to the prescribed tolerance \( \eta = 10^{-15} \). To obtain this level of accuracy, it is generally necessary to perform at least twenty iterations (see results for \( k_{\text{max}} = 50 \)). Fig. (2b) shows the change in error \( E_m \) for each PGD enrichment \( m \). To obtain an error lower than \( 10^{-13} \), the PGD method required around seventy functions for \( k_{\text{max}} = 5, 10, 50 \), compared with one hundred and ten for \( k_{\text{max}} = 2 \). Thus, even when \( k_{\text{max}} = 50 \) and the fixed point algorithm converges, the number of PGD functions required to obtain a given accuracy is very close to that required when \( k_{\text{max}} = 10 \). To conclude, even when \( k_{\text{max}} = 10 \) and the fixed point algorithm does not converge, this is still sufficient to obtain an accurate solution quickly.

The influence of parameter \( \eta \), which corresponds to the accuracy of the fixed point algorithm, on PGD convergences, is illustrated in Fig. (3a). It should be noted that for values smaller than \( 10^{-4} \), parameter \( \eta \) does not affect PGD convergence. Finally, Fig. (3b) shows that when the number of discretization nodes increases, the number of functions needed for PGD convergence increases too.

4. Navier–Stokes problem

4.1. \( P_N-P_{N,2} \) method associated with a fractional step scheme

4.1.1. A third-order projection scheme

We considered the dimensionless unsteady Navier–Stokes equations for Newtonian incompressible fluids in a primal velocity–pressure formulation. For any open domain \( \Omega \subset \mathbb{R}^2 \) and a positive \( T \), we looked for the velocity field \( \mathbf{u} \) and the pressure field \( p \) solutions of

\[
\begin{align*}
  \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p &= \mathbf{f} \quad \text{in } \Omega \times [0,T] \\
  \nabla \cdot \mathbf{u} &= 0 \\
  \mathbf{u}(t = 0) &= \mathbf{u}_0 \\
  \mathbf{u} &= \mathbf{g} \quad \text{on } \partial \Omega
\end{align*}
\]

(28)

where \( \mathbf{f}, \mathbf{u}_0, \) and \( \mathbf{g} \) are given vectors and \( \text{Re} \) is the Reynolds number.

The fractional step schemes involve computing the velocity and the pressure fields separately, through the computation of an intermediate velocity, which is then projected onto the subspace of divergence-free functions (see [42–44]). In this work, an Adams–Bashforth/third-order backward Euler projection scheme introduced by Botella in [41] was used. At the \( n + 1 \) time step, we were looking for \( \mathbf{u}^{n+1} \) and \( p^{n+1} \). The steps which compose this algorithm are detailed in the following:

(1) The estimated velocity \( \mathbf{u}^{n+1} \) is the solution to the correction step defined by:

\[
\begin{align*}
  \sigma \mathbf{u}^{n+1} - \frac{1}{\text{Re}} \nabla^2 \mathbf{u}^{n+1} &= \mathbf{f}^{n+1} \\
  \mathbf{u}^{n+1} &= \mathbf{g}^{n+1} \quad \text{on } \partial \Omega
\end{align*}
\]

(29)

where \( \sigma = \frac{1}{\text{Re}} \) and \( \Delta t \) is the time step. Vector \( \mathbf{f}^{n+1} \) is given by:
The entries for the pressure derivative matrix in the $x$-direction, called $\tilde{D}^{N_x}$, are given by:

\[
\tilde{D}^{N_x}_{ij} = \frac{3 \cos \beta}{2 \sin \beta} \frac{\sin \frac{\pi j}{2}}{\sin \frac{\pi i}{2}} \frac{(-1)^{i+j}}{\sin \frac{\pi j}{2} \sin \frac{\pi i}{2}} \quad 1 \leq i, j \leq N_x - 1, \ i \neq j
\]

The entries for the pressure derivative matrix in the $y$-direction, called $\tilde{D}^{N_y}$, are defined in the same way.

The prediction step (29) constitutes a Helmholtz problem for the estimated velocity $\tilde{u}^{n+1}$ with Dirichlet boundary conditions which could be solved using standard methods. The projection step corresponds to a Darcy problem which is solved using an Uzawa algorithm.

\[
\sigma U + \tilde{D}^{N_x} p = \tilde{f}_x
\]
4.1.4. PGD for Navier–Stokes equations

The operators \( D^{1/2}_g \) and \( D^{1/2}_f \) are defined in problem 1 and Eq. (35) into (36), the following matrix equation for pressure is obtained:

\[
\begin{align*}
\left( D^{1/2}_g \right) P + \left( D^{1/2}_f \right) G &= G \\
G &= D^{1/2}_g \bar{F}_x + D^{1/2}_f D^{1/2}_g - \sigma S
\end{align*}
\]

Substituting Eqs. (34) and (35) into (36), the following matrix equation for pressure is obtained:

\[
\begin{align*}
\left( D^{1/2}_g \right) P + \left( D^{1/2}_f \right) G &= G \\
G &= D^{1/2}_g \bar{F}_x + D^{1/2}_f D^{1/2}_g - \sigma S
\end{align*}
\]

with

\[
G = D^{1/2}_g \bar{F}_x + D^{1/2}_f D^{1/2}_g - \sigma S
\]

The operators \( D^{1/2}_g \) and \( D^{1/2}_f \) are called Uzawa operators. The pressure is then obtained by solving Eq. (38).

4.1.4. PGD for Navier–Stokes equations

The problem is solved using the PGD algorithm. Assuming we are at \( n + 1 \) time step, \( p^{n + 1}, u^{n + 1}, \bar{u}^{n - 1} \) and \( \bar{u}^{n - 2} \) are known and \( p^{n + 1} \) and \( \bar{u}^{n + 1} \) are then calculated. For that purpose, the following steps were carried out:

1. The x-component \( \bar{u}^{n + 1} \) of the estimated velocity \( \bar{u}^{n + 1} \) is sought in the form

\[
\bar{u}^{n + 1}(x, y) = \sum_{k=1}^{m_n} \chi_k^{n + 1}(x) C_k^{n + 1}(y)
\]

The PGD strategy is similar what was presented in Section 3.1 with

\[
A_g(F^{n + 1}_g, C^{n + 1}_g) = \sigma T^{n + 1}_g \left( \sum_{l=1}^{N_n} D^{n + 1}_g F^{n}_g(x_l) C^{n + 1}_g(y_l) \right)
\]

and

\[
S_{g} = \bar{F}_x^{n + 1}(x, y) - D^{n + 1}_g g_x^{n + 1}(x, y) - D^{n + 1}_g g_x^{n + 2}(x, y) - D^{n + 1}_g g_x^{n + 1}(x, y)
\]

where \( g_x \) is the x-component of the vector \( g \) defined in problem (28) and \( \bar{F}_x^{n + 1} \) is the x-component of the source term vector \( \bar{F}^{n + 1} \).

\( \bar{u}^{n + 1} \), the y-component of the estimated velocity \( \bar{u}^{n + 1} \), is computed in the same way such that:
\[ \tilde{p}^{n+1}(x,y) = \sum_{k=1}^{m_p} \delta^k_x F^k_p(x) G^k_p(y) \]  

(42)

(2) The pressure \( p^{n+1} \) was calculated in the form
\[ p^{n+1} = \sum_{k=1}^{m_p} \delta^k_x F^k_p(x) G^k_p(y) \]  

(43)

In this case, Eq. (38) was solved in a similar manner to that shown in Section 3.1 with the pressure operators defined in section 4.1.3.

(3) Velocity \( u^{n+1} \) was updated using Eq. (32).

4.2. Applications

We decided to solve Navier–Stokes equations using PGD associated with a \( P_N-P_{N-2} \) spectral discretization. In order to validate the approach, we first considered a Darcy problem, which corresponds to the projection step of the fractional step scheme. Then, two different Taylor Green problems were examined to assess the accuracy of the method when the time step and the grid size vary. Finally, the method was applied to solve the flow in a 2D lid-driven cavity.

4.2.1. Darcy problem

In this section the following problem is solved:
\[
\begin{cases}
\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega \\
\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega \\
\mathbf{u} \cdot \mathbf{n} = g \text{ on } \partial \Omega
\end{cases}
\]  

(44)

where \( \mathbf{u} = (u, v) \) is the velocity and \( p \) is the pressure. This problem corresponds to the projection step (31). We chose the case studied by Botella [40] which admits the following analytical solutions:

\[
\begin{align*}
u_{\text{ana}}(x,y) &= 4\pi \cos(4\pi x) \sin(4\pi y), \\
p_{\text{ana}}(x,y) &= \cos(4\pi x) \cos(4\pi y)
\end{align*}
\]

By carrying these solutions into problem (44), the source terms and boundary conditions are well defined. An arbitrary pressure constant was chosen in order to make the calculated pressure at the central point of the mesh equal to the exact one.

The problem was solved using PGD and with a full grid spectral standard solver. The convergence coefficient of the PGD algorithm was taken to be equal to \( \epsilon_p = 10^{-8} \). Simulations were performed for a mesh characterized by \( N \) collocation points in each direction and ranging from 8 to 64. The relative error of the analytical solution, defined by Eq. (26), was plotted as a function of the number of nodes in each direction \( N \) (see Fig. 4).

This figure shows that the PGD solver is as accurate as the full grid solver. Moreover, from 32 nodes in each direction, the error is no longer significant. We observed an exponential convergence with the number of nodes. These results are similar
In this case, with that obtained with the horizontal velocity.

considered the case studied by Botella in [40]. The error associated with the vertical velocity has not been drawn because it is similar to that obtained with the horizontal velocity.

The CPU time required to solve this problem by PGD was much lower than the CPU time necessary with a full grid solver. In this case, with \( N = 64 \), the PGD solver was 100 times faster than the standard solver. This result is due to the fact that PGD only requires one couple of functions to reconstruct the solution properly.

4.2.2. Taylor–Green problem

The Taylor Green problem corresponds to the Navier–Stokes problem where an analytical solution is provided. We considered the case studied by Botella in [41]. This consists in solving Navier–Stokes equations in a domain \( \Omega = [-1, 1] \times [-1, 1] \) and on the time interval \([0, T]\), where the solutions are given by:

\[
\begin{align*}
    u_{\text{ana}}(x, y, t) &= \cos(\gamma t) \sin \left( \frac{\pi x}{2} \right) \cos \left( \frac{\pi y}{2} \right), \\
    v_{\text{ana}}(x, y, t) &= -\cos(\gamma t) \cos \left( \frac{\pi x}{2} \right) \sin \left( \frac{\pi y}{2} \right), \\
    p_{\text{ana}}(x, y, t) &= \frac{1}{4} \cos^2(\gamma t) [\cos(\pi x) + \cos(\pi y)] + 10(x + y) \cos(\gamma t).
\end{align*}
\]

The boundary conditions and the source term \( f \) were defined in order to match the analytical solution. Reynolds number was set at \( Re = 100 \). To initialize the time integration scheme, the fields \( u \) and \( p \) at \(-2\Delta t, -\Delta t \) and 0 were taken to be equal to the exact solution. The coefficients needed to check PGD convergence were set at \( \epsilon_0 = 10^{-13} \) for intermediate velocities and \( \epsilon_p = 10^{-8} \) for pressure. In both cases, the fixed point criterion was set at \( \eta = 10^{-5} \).

To check the spatial accuracy of PGD, we first studied the stationary Navier–Stokes equation by taking \( \gamma = 0 \). The discrete errors \( E_u = ||\Phi - \Phi_{\text{ana}}|| \) on the collocation inner points for fields \( \Phi = u, v \) or \( p \) are displayed in Table 3. These results show that PGD is as accurate as the full grid standard solver. In this case too, PGD requires only one couple of functions to reconstruct the solution exactly.

The study of temporal accuracy was performed on the unsteady, time periodic analytical solutions with \( \gamma = 5 \). The time step was set to \( \Delta t = 10^{-3} \) and a grid with 16 collocation points in each direction was considered. Fig. 5 shows the changeover time of errors \( E_u \) and \( E_p \) obtained by PGD. We observed no amplification of the errors with time, which suggests the numerical solutions are stable. It is noteworthy that these results are close to those obtained by Botella and confirm the ability of PGD to treat Navier–Stokes equations with a spectral discretization. Errors obtained with the full grid solver have not been plotted because they are exactly the same as those obtained by PGD.
Another test consists in studying the relative error with respect to the analytical solution, defined by Eq. (26), according to the time step. Fig. 6 shows that the behaviour of the PGD errors for velocity and pressure are very close to those obtained by a full grid solver. Moreover, the errors obtained by PGD have an exponential decay with the time step, which is characteristic of spectral methods.

As in Section 3.2, the influence of the various parameters of the fixed point algorithm \((\eta, k_{\text{max}})\) on the accuracy of the solution was studied. As a first step, we were interested in the influence of the \(k_{\text{max}}\) parameter on the prediction of the intermediate velocity. Figs. (7a) and (7b) show that, in this case, the convergence of the fixed point algorithm does not affect the prediction of the velocity. Indeed, the algorithm did not converge, except for \(k_{\text{max}} = 50\), and the error \(E_m\) was the same for each \(k_{\text{max}}\). Note that the results obtained for the intermediate \(y\)-velocity are similar to those obtained here. Fig. (8a) shows that the parameter \(\eta\) does not influence the prediction of the velocity. Consequently, in the following, the parameters of the fixed point algorithm required to compute the intermediate velocities was set at: \(\eta = 10^{-5}\) and \(k_{\text{max}} = 10\).

As a second step, we were interested in the influence of the \(k_{\text{max}}\) parameter on the prediction of the pressure. Fig. (7c) shows that independently from the choice of the \(k_{\text{max}}\) parameter, the fixed point algorithm did not converge. In this case, the number of iterations in the fixed point algorithm affects the behaviour of the error (see Fig. (7d)). We observed that the smaller the \(k_{\text{rho}}\), the higher the number of PGD functions necessary for the convergence of the PGD algorithm. To obtain an error of \(E_m = 10^{-13}\), the choice of a \(k_{\text{rho}} = 2\) led to \(m = 80\), the choice of a \(k_{\text{rho}} = 10\) led to \(m = 32\) and the choice of a \(k_{\text{rho}} = 50\) led to \(m = 20\). Since the purpose of this work was to save computational time, the choice of \(k_{\text{rho}} = 50\) is not convenient because the time spent in the fixed point algorithm is too long. The best compromise between CPU time saving in the fixed point algorithm and the desired number of PGD function products for convergence seems to correspond to a value of \(k_{\text{rho}}\) equal to 10. Fig. (8a) shows that the value of \(\eta\) does not affect the accuracy of the pressure prediction. Consequently, in the section below, for the lid driven cavity, the parameters were set to: \(\eta = 10^{-5}\) and \(k_{\text{max}} = 10\).

The comparison of PGD CPU time and standard solver CPU time is illustrated in Fig. 9. It is clear that beyond a mesh size of 26 × 26, PGD was faster than the standard method. In fact, for a mesh size of 48 × 48, PGD was ten times faster than the standard method.

4.2.3. Lid-driven cavity flow problem

We will now discuss the case of a flow in a square lid-driven cavity, as illustrated in Fig. 10. The two velocity components vanish on the boundary, except on the north face where the \(x\)-velocity is equal to \(U_0\). The source term \(f\) is taken equal to zero.

Simulations were performed for \(Re = \frac{U_0 D}{\nu}\) equal to 100 with a time step equal to \(\Delta t = 10^{-3}\) and a non regular grid with 20 nodes in each direction. For this Reynolds number, simulations converged towards stationary solutions. So the following convergence criteria were defined:

\[
\frac{t}{||\Phi^{k} - \Phi^{k-1}||} ||\Phi^{k}|| \leq \gamma_u \quad \text{with} \quad \Phi = u, v
\]

\[
\frac{t}{||\nabla p^{k} - \nabla p^{k-1}||} ||\nabla p^{k}|| \leq \gamma_p
\]

\(\gamma_u\) (resp. \(\gamma_p\)) was taken as being equal to \(10^{-10}\) (resp. \(10^{-5}\)). Finally, the simulations were performed with a full grid solver and with the PGD method. Coefficients needed to check the convergence of PGD algorithm were set at \(\epsilon_u = 10^{-10}\) for intermediate velocities and \(\epsilon_p = 10^{-5}\) for pressure. Concerning the fixed point parameters, based on the results obtained in the previous section, we chose \(\eta = 10^{-5}\) and \(k_{\text{max}} = 10\) to compute velocities and pressure.
Fig. 7. Convergence of PGD for the Taylor–Green problem at \( t = 1 \) s with \( N = 16 \). (a and b) Intermediate x-velocity for \( \eta_a = 10^{-15} \), (c and d) pressure with \( \eta_a = 10^{-15} \).

Fig. 8. Convergence of PGD solutions at \( t = 1 \) s for the Taylor–Green problem with \( k_{\text{max}} = 10 \) and \( N = 16 \).
Fig. 9. Comparison between full model CPU time and PGD model CPU time.

Fig. 10. Geometry of a lid-driven cavity.

Fig. 11. First functions of the tensor product computed with PGD that represent the $x$-velocity field.
The first PGD functions $F_i$ and $G_i$ for the x-velocity are shown in Fig. 11. The streamlines and the isocontours of pressure obtained by PGD and by the full grid solver are shown on Figs. 12 and 13. The PGD results are in agreement with those of the standard solver.

Table 4 shows the extreme velocities along the centerlines and their corresponding positions. The minimum horizontal velocity on the vertical centerline is denoted as $u_{\text{min}}$ and its location as $y_{\text{min}}$. The maximum and minimum vertical velocities on the horizontal centerline are denoted by $v_{\text{max}}$ and $v_{\text{min}}$, respectively, and their locations by $x_{\text{max}}$ and $x_{\text{min}}$. These results show that PGD associated with a spectral discretization is able to provide a highly accurate solution.

The change in the error defined by Eq. (27) in relation to the number of PGD functions is plotted in Fig. 14. We observed that, for the velocity in the lid-driven cavity, sixty function products were necessary to reconstruct the full grid solution. For the pressure, approximately eighty products were necessary to achieve convergence.

The change in the ratio between the CPU time of the full solver and the PGD solver with the number of nodes is shown in Fig. 15. At 25 nodes, and above, the PGD method was faster than the standard solver. Moreover, with a $36 \times 36$ grid, PGD was four times faster than the standard solver.

**Table 4**

<table>
<thead>
<tr>
<th>Source</th>
<th>Method and grid</th>
<th>$u_{\text{min}}$</th>
<th>$v_{\text{max}}$</th>
<th>$v_{\text{max}}$</th>
<th>$x_{\text{max}}$</th>
<th>$v_{\text{min}}$</th>
<th>$x_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ghia et al. [45]</td>
<td>Finite difference, $N = 129$</td>
<td>-0.21090</td>
<td>-0.0469</td>
<td>0.17527</td>
<td>-0.2656</td>
<td>-0.24533</td>
<td>0.3047</td>
</tr>
<tr>
<td>Deng et al. [46]</td>
<td>Finite Volume</td>
<td>-0.21405</td>
<td>-</td>
<td>0.17896</td>
<td>-</td>
<td>-0.25399</td>
<td>-</td>
</tr>
<tr>
<td>Zhang et al. [47]</td>
<td>Pseudospectral, $N = 48$</td>
<td>-0.2140482</td>
<td>-0.0419</td>
<td>0.1795763</td>
<td>-0.2630</td>
<td>-0.2538134</td>
<td>0.3104</td>
</tr>
<tr>
<td>Present</td>
<td>Pseudospectral, $N = 20$</td>
<td>-0.2141353</td>
<td>-0.0419</td>
<td>0.1797115</td>
<td>-0.2630</td>
<td>-0.2541496</td>
<td>0.3104</td>
</tr>
</tbody>
</table>
We will now discuss the optimality of the PGD method. For this purpose, the number of PGD functions required to obtain a solution was analyzed. A SVD (Singular Value Decomposition) [48,49] was applied. If the PGD method is optimal, the number of PGD functions should be almost the same as the number of functions obtained by the SVD. To achieve this, the change in the norm of the residual of each equation solved depending on the number of PGD functions or SVD functions was considered. The residual is defined as:

$$\text{Res}^N_w = \mathcal{L}_w \left[ \sum_{i=1}^{N_w} F_w^i G_w^i \right] - SM_w$$

where $w$ corresponds to the different unknowns of the problem, namely the horizontal velocity and the pressure. The results obtained for vertical velocity were very close to those obtained for horizontal velocity, which is why this velocity will not be considered in the following. $\mathcal{L}_w$ (respectively $SM_w$) is the operator (resp. the second member) associated with the equation which was used to calculate the variable $w$. Functions $F_w^i, G_w^i$ are the PGD functions or the SVD functions of the unknown $w$ at the last time step. Below, FULL-SVD corresponds to the results obtained with a SVD on the full grid spectral standard solver, and PGD-SVD refers to the results obtained with a SVD on the PGD solution.

The norm of the residual associated with the equation satisfied by the velocity depending on the number of retained functions $N_w$ is plotted on Fig. 16 (left). The residual associated with the horizontal velocity obtained by PGD, PGD-SVD and FULL-SVD gave the same curve of residuals up to a value of $10^{-5}$. This proves that, for this variable, PGD gives the same optimal representation as could be provided by a SVD of the full solution. From this value, the PGD curve decreased more slowly to a norm of the residual close to zero in approximately sixty functions, while SVDs suddenly decreased to zero in twenty functions. The same results are plotted for pressure in Fig. 16 (right). Here, it is more difficult to reach a conclusion. In fact, both SVDs converged to a value of the residual norm that was less than $10^{-5}$ in twenty function products, while PGD needed more...
than seventy products. It is clear that the convergence in pressure is slower than the convergence in velocity. PGD is not optimal in this case either.

5. Conclusions

In this work, the PGD method was coupled with a spectral discretization to solve various transfer equations. For all the 2D cases considered (the stationary diffusion equation, the Darcy problem and the Navier–Stokes equations), the PGD results were as accurate as those obtained with a full model. Moreover, above a given number of nodes, PGD was faster than the full solver. For the 2D stationary diffusion equation, using PGD was about 100 times faster on a $64 \times 64$ mesh size. For Navier Stokes equations, it was around five or ten times faster, depending on the problem considered. The study of the influence of the fixed point algorithm on the accuracy of the solution showed that the convergence of the fixed point algorithm does not affect the solution significantly.

Increasing the Reynolds number will be the subject of further studies. This work is also the first step toward dealing with a 3D situation. Indeed it would be interesting to include the time coordinate in the separated representation of the PGD solution or to consider 3D spatial problems. It might be hoped that the time saved will be greater than in the 2D case.

References
