



**Institut Supérieur de l'Aéronautique et de l'Espace**
**ONERA, the French Aerospace Lab**

# Model Reduction in Aeroelasticity for Preliminary Design

---

**DOCUMENT:** Research Project's Report

**DEGREE:** Diplôme d'Ingénieur ISAE SUPAERO (MSc)

**STUDENT:** CHANDRE VILA, Oriol

**SUPERVISORS:**

MORLIER, Joseph (ISAE SUPAERO)
DUBREUIL, Sylvain (ONERA)

**DELIVERY DATE:** 13/03/2019

# Contents

# List of Tables

# List of Figures

# 1 Introduction

## 1.1 Presentation

This report aims to present the current work developed during my one-year research project within the umbrella of the *Parcours Recherche* proposed by ISAE-SUPAERO. This project has been monitored by Joseph MORLIER (ISAE) and Sylvain DUBREUIL (ONERA). Divided into three stages, first of all, some techniques have been tried in a 1-equation problem. Then the best methodology has been applied to a 2-equation problem and, finally, introducing the technique to more general problems has been approached.

## 1.2 Aim

This project aims to implement Reduced Order Models in fluid-structure problem simulations in order to lighten the computational requirements of the preliminary design phase.

## 1.3 Projects Overview and Objectives

This project has three main stages that have been performed during the last 12 months:

1. General approach to Reduced Order Models (ROM):
   - To develop bibliographic research of the topic.
   - To test several methodologies.
   - To select one methodology for future fluid-structure problems reduction.
2. Application of the chosen methodology to an own code of fluid-structure interactions (internship in ONERA).
3. Application of the chosen methodology to *OpenAeroStruct*, solver provided by the University of Michigan.

## 1.4 Motivation

In front of a more and more demanding industrial world, it is expected to compute more precise and wider designs. Consequently, the computational power required is increasingly important. In this connection, Model Order Reduction (hereafter MOR) is a technique for reducing the computational complexity of mathematical models in numerical simulations. By a reduction of the model's associated state space dimension or degrees of freedom,

an approximation to the original model is computed, which normally is less expensive [7]. Consecutively, two approaches exist to reduce the number of random variables under consideration by obtaining a set of principal variables: ***(a)*** feature selection and ***(b)*** feature extraction. In this analysis, we are interested in the latter which involves reducing the number of resources required to describe a large set of data. General dimensionality reduction techniques are used in different methodologies [8].

# 2 State of the Art

## 2.1 Principal Component Analysis (PCA)

The first category of ROM analysed -which is widely used- is the *Principal Component Analysis* (PCA). PCA is a statistical procedure which converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables -called principal components- by using an orthogonal transformation. This transformation is defined such the first principal component has the largest variance (how much the data could variate), and the succeeding components must be orthogonal to their precedent while having the highest variance [9]. Furthermore, depending on the field of application, this technique has received different names. Just the fields in which this paper is interested are listed: ***(a)*** singular value decomposition (SVD) in linear algebra [10], ***(b)*** proper orthogonal decomposition (POD) in mechanical engineering [11] [12], ***(c)*** discrete Karhunen–Loève transform (KLT) in signal processing [13], ***(d)*** robust PCA -which uses the $L^1$ instead of the norm $L^2$- [10] and, ***(e)*** factor analysis in psychometrics [14]. The different fields usually aspire to evolve their problem to an algebraic issue in order to use SVD technique. The aforementioned ROM is defined as a factorisation of a matrix into a number of constitutive components all of which have a specific meaning in applications [10]: $\mathbf{A} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^*$, with $\mathbf{U} \in \mathbb{C}(mxm)$, $\Sigma \in \mathbb{R}(mxn)$ where $(\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{p=min(m,n)} \geq 0)$, and $\mathbf{V} \in \mathbb{C}(nxn)$. In modern partial SVD algorithms, the properties of random vectors are exploited to efficiently build a subspace that captures the column space of a matrix [15].

Some limitations can be appreciated in PCA. Firstly, the results depend on the scaling of the variables. Secondly, the applicability of PCA is limited by certain assumptions made in its derivation. Thirdly, in fields where all the signals are non-negative and the mean-removal process forces the mean of some exposures to be zero (which consequently creates nonphysical negative fluxes), it is necessary a forward modelisation to recover the true magnitude of the signals. Finally, since PCA does not explicitly attempt to model the difference between classes of data, *Linear Discriminant Analysis* (LDA) was developed [16]. The above-mentioned works by creating one or more linear combinations of predictors, creating a new latent variable for each discriminant function. In general terms, the first function created maximises the differences between groups on that function; and the following functions must not be correlated with the previous functions while maximising the differences [17].

## 2.2 Independent Component Analysis (ICA)

The second category of ROM analysed is the *Independent Component Analysis* (ICA). This computational method separates a multivariate signal into additive subcomponents, and it is a special case of blind source separation. On one hand, ICA is based on two assumptions: ***(a)*** the source signals are independent of each other and ***(b)*** the values in each source signal have non-Gaussian distributions. On the other hand, three effects may appear due to mixing the source signals: ***(a)*** *independence*: the source signals are independent but their signal mixtures are not; ***(b)*** *normality*: the distribution of a sum of independent random variables with finite variance yields to a Gaussian distribution, and, ***(c)*** *complexity*: the complexity in time of any signal mixture is greater than that of its simplest constituent source signal [18].

## 2.3 Dynamic Mode Decomposition (DMD)

The third category is not a ROM itself. *Dynamic Mode Decomposition* (DMD) is a matrix decomposition technique that is highly versatile and builds upon the power of SVD. DMD appears as an alternative ***(a)*** when the model equations are beyond their range of validity or ***(b)*** when the governing equations simply are not known or well-formulated [10]. By computing the eigenvalues and eigenvectors of a linear model that approximated the underlying dynamics -even if the system is nonlinear-, experimental data itself drives the understanding of the system [10]. DMD could be perceived as minimizing the norm of the difference between the stored experimental data in $k+1$ and the approximative dynamics obtained by the linear expression $\mathbf{x}_{k+1} = \mathbf{A} \cdot \mathbf{x}_k$; over $k = 1, 2, 3, ..., m-1$ snapshots. The rank $q$ of a POD-projected matrix $\tilde{\mathbf{A}}$ would be iterated until achieving the required precision while accomplishing the lowest $q$ [19]. Despite the fact that it fails immediately if the data matrix is full rank and the data has no suitable low dimensional structure, it is possible to highlight three advantages of DMD: ***(a)*** no equations are needed and ***(b)*** the future state is always known.

## 2.4 Proper Generalized Decomposition (PGD)

The fourth (and last) category analyzed is not a ROM it-self too. Since solving decoupled problems is computationally much less expensive than solving multidimensional problems, *Proper Generalized Decomposition* (PGD) is usually considered a dimensionality reduction algorithm. Thus, this numerical method for solving boundary value problems assumes that the solution of a multidimensional (or multiparametric) problem can be expressed in a separated

representation, of the form:

$$\mathbf{u}(x_1...x_d) = \sum_{i=1}^{N} \mathbf{F}_i^1(x_1) \times \cdots \times \mathbf{F}_i^D(x_d) \tag{2.1}$$

where the number of terms $N$ and the functions $\mathbf{F}_i$ are a priori unknown. At a particular enrichment step $n+1$ of (2.1), the functions are known for $i \leq n$ from the previous steps. So it should be only computed the new product involving the $D$ unknown functions $\mathbf{F}_{n+1}^j(x_j)$ [20]. In a problem defined as $\mathbf{u}(x,y) = \sum_{i=1}^{N} \mathbf{X}_i(x) \cdot \mathbf{Y}_i(y)$, PGD uses the weak problem formulation (see Algorithm 1).

**Algorithm 1** PGD. [20]

**Input:** Data stored in $u^{n,p}(x,y)$.

1. Calculating $\mathbf{X}_n^p(x)$ from $\mathbf{Y}_n^{p-1}(y)$:

$$u^{n,p}(x,y) = \sum_{i=1}^{n-1} \mathbf{X}_i(x) \cdot \mathbf{Y}_i(y) + \mathbf{X}_n^p(x) \cdot \mathbf{Y}_n^{p-1}(y) \tag{2.2}$$

2. Calculating $\mathbf{Y}_n^p(y)$ from $\mathbf{X}_n^p(x)$.

$$u^{n,p}(x,y) = \sum_{i=1}^{n-1} \mathbf{X}_i(x) \cdot \mathbf{Y}_i(y) + \mathbf{X}_n^p(x) \cdot \mathbf{Y}_n^p(y) \tag{2.3}$$

**Return:** $\mathbf{X}_n^p(x)$ and $\mathbf{Y}_n^p(y)$

Within PGD strategy, $(d \cdot M \cdot N)$ unknowns factors are concerned instead of $M^d$ unknowns of a traditional finite element (FE) method ($M$ is the number of nodes, $d$ is the dimension of the problem and $N$ is the number of summands). In the Table 2.1, it can be seen the saving in computational costs of PGD. The column $M^d$ shows the cost of a FEM, the column $(d \cdot M \cdot N)$ of PGD and the column $max(N)$ represents the number of summands required for achieving $M^d \equiv (d \cdot M \cdot N)$ [1].

Table 2.1: Cost computational comparison [1].

| **d** | $M^d$ **(M=100)** | $(d \cdot M \cdot N)$ | $max(N)$ |
|---|---|---|---|
| 3 | $10^6$ | $300 \cdot N$ | 3333 |
| 5 | $10^{10}$ | $500 \cdot N$ | $2{\times}10^4$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 40 | $10^{80}$ | $4{\times}10^3 \cdot N$ | $2,5{\times}10^{76}$ |

The most advantageous application of PGD is the evaluation of design parameters as new coordinates of the problem. So that, in real-time, it will be possible to evaluate the effect of

any modification. This approach is possible thanks to robust offline calculations: a general, static and expensive solution is computed and stored in $\{\mathbf{R}, \mathbf{S}, \mathbf{T}, \cdots, \mathbf{Z}\}$ vectors of the Greedy algorithm -shown in (2.4).

$$\mathbf{u}^{N+1}(x) = \mathbf{u}^{N}(x) + \mathbf{R}(x_1)\mathbf{S}(x_2)\mathbf{T}(x_3)\cdots\mathbf{Z}(x_d) \tag{2.4}$$

Then, by multiplying and adding the components of these vectors, an online solution is reconstructed.

## 2.5 Computing Performance Analysis of ROM

*Reduced Basis* (RB) methods, which represent a remarkable instance of ROM techniques, were used to exploit the parametric dependence of the PDE solution by combining a handful of high-fidelity (HF) solutions computed for a set of parameter values. By this approach, a very large algebraic system is replaced by a much smaller one. If $N$ and affine operator components $Q_a$ are small enough, very fast response can be achieved for real-time problems using a Galerkin reduced basis (G-RB). Quaternoni et al., took advantage of a suitable offline/online computational splitting formulation to compute error bounds efficiently: ***(a)*** computing the norm of the residual (see Algorithm 3.4, p. 62 in [2]); and, ***(b)*** computing the stability factor $\beta_h$. In the latter, a *Interpolatory Radial Basis Functions* was selected. In order to guarantee the positivity of the interpolant, it was interpolated the logarithm of $\beta_h$ rather than factor itself. This leads to a symmetric linear system which was solved in the offline phase to yield the interpolation weights (see Algorithm 3.7, p. 67 in [2]).

Thereupon, the G-RB was tested on a steady heat conduction-convection problem satisfied by the non-dimensional temperature $u$ [2]. In addition, the problem was discretized by piecewise linear FE using a mesh made of $\approx 2,2{\times}10^5$ tetrahedral elements. Thanks to the offline/online decomposition, for any given parameter value $\mu$ at the online stage, the solution can be obtained very rapidly (as can be observed in the Table 2.2). The offline/online decomposition is a direct consequence of the affine parametric dependence property, which allows splitting the assembly of the reduced matrices and vectors.

Table 2.2: Computational details for the HF and RO models of the advection-diffusion problem. [2]

| High-fidelity model | | Reduced-order model | |
|---|---|---|---|
| Number of FE dofs $N_h$ | 44171 | Number of RB dofs | 29 |
| Affine operator components $Q_a$ | 2 | Dofs reduction | 1520:1 |
| Affine RHS components $Q_f$ | 6 | Offline CPU time | $\approx 5\ min$ |
| FE solution time (assembly + solution) | $\approx 3,5\ s$ | Online CPU time | $1\ ms$ |

The analysis is concluded with several comments. Firstly, it is said that the solution $u_h(\mu)$ is highly dependent of $\mu$. Secondly, as can be seen in Figure 2.1, the estimator $\Delta_N(\mu)$

is very sharp and its effectivity $\eta_N(\mu)$ is very closed to 1. And thirdly, the error presents an exponential decay, which is a remarkable property of the RB approximations. Consequently, they are very accurate and efficient.

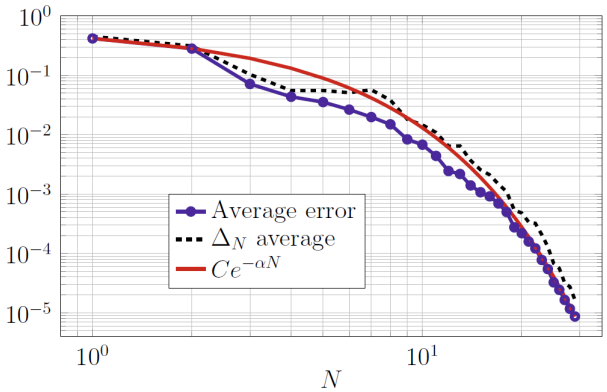


Figure 2.1: Comparison of the average error $u_h(\mu) - u_N(\mu)_V$ and bound $\Delta_N(\mu)$ computed on a set of 350 random parameter values.[2]

Continuing analyzing the current problem, a POD was used to construct a RB approximation. POD approach uses the error estimator $\Delta_N(\mu)$ to acquire a more accurate result. Quarteroni et al., used *Latin Hypercube Sampling* (LHS) because it is a very good sampling technique for initial points. It was reported in Figure 2.2 the spectrum obtained using LHS with $n_s = 25, 50, 100, 200, 5000$. The snapshot of $n_s = 100$ was used to extract a RB because the spectrum obtained is already sufficiently accurate. The RB extracted required a dimension of $N = 26$ to achieve $\varepsilon_{POD} = 10^{-5}$.

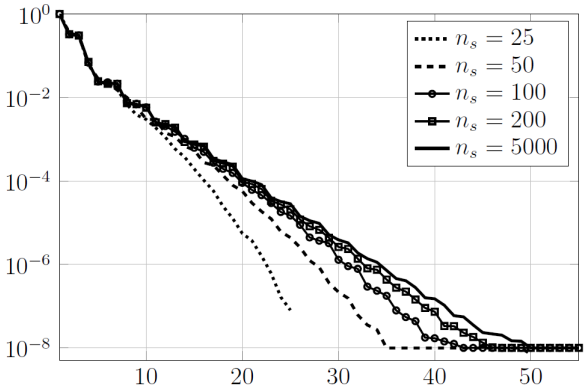


Figure 2.2: First 55 singular values of the correlation matrix obtained by LHS.[2]

To construct RB spaces, greedy algorithms might be also used. This option can be evaluated if the POD method entails a severe computational cost. Greedy algorithms represent an efficient alternative as they allow the construction of the reduced space by minimizing the number of snapshots to be evaluated (by seeking at each step the local optimum). Nevertheless,

two potentially critical aspects have to be considered. A greedy algorithm: ***(a)*** must be supported by a posteriori error estimated with $u_h(\mu) - u_N(\mu)_V$ in a very inexpensive way (which has demonstrated to be less accurate); and, ***(b)*** is not necessarily cheaper than a POD. In that point, it is essential to make greedy algorithms competitive with POD.

Remaining in the advection-convection problem, a greedy algorithm was introduced to construct a RB approximation. Firstly, it was performed a preprocessing step to build the RBF interpolant of the stability factor $\beta_h(\mu)$. Since the problem is coercive and it depends only on the $\mu_4$ (number of sampling points) direction, equidistributed interpolation points along the above-mentioned direction were chosen. Using the Algorithm 7.3 (p. 146 in [2]), the convergence history of the greedy algorithm had been reported (see Figure 2.3). This historical evolution had been performed with different train sample ($\Xi_{train}$) made of $n_{train} = 10^2, 10^3, 10^4$ points selected by LHS; stopping at $N = 27$ in the first case and at $N = 29$ in the other cases.

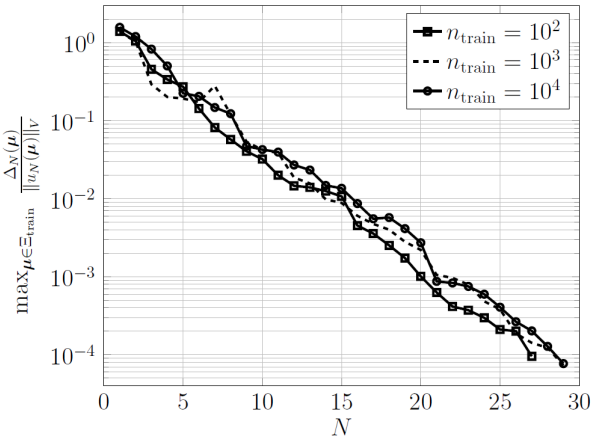


Figure 2.3: Convergence history of the greedy algorithm.[2]

## 2.6 ROM in Aeroelasticity

The final objective of the study concerned within this paper, is the definition of a strategy based on ROM to attack Aeroelastic problems. In the paper published by Lucia et al., a POD produced an accurate model for the coupled system of an aeroelastic panel in crossflow [3]. In concert with the POD, a domain decomposition (DD) was considered necessary in order to commit the overcoming difficulties in translating discontinuities in the flowfield.

For the **Structural dynamics equations**, the panel dynamics were computed with von Karman's large-deflection plate equation (for the theoretical development: [3]); which yields to

the structural state $\mathbf{Y}_s = \begin{bmatrix} w(x) \\ s(x) \end{bmatrix}$:

$$\dot{\mathbf{Y}}_s = \begin{bmatrix} 0 & L_N \\ I & 0 \end{bmatrix} \mathbf{Y}_S + \begin{bmatrix} \mu \mathbf{P} \\ 0 \end{bmatrix} \tag{2.5}$$

$$L_N = -\frac{\mu}{\lambda} \frac{\partial^4}{\partial x^4} + N_x \frac{\partial^2}{\partial x^2} \tag{2.6}$$

Where: $w(x)$ are the panels deflections, $s(x) = \dot{w}(x)$ are the velocities, $\mathbf{I}$ is the identity matrix, $L_N$ is a nonlinear matrix operator (where $\partial\partial x$ represents central-difference, spatial discretization), the array $\mathbf{P}$ collecting the values of $\left(\frac{1}{\gamma M_\infty^2} - P\right)$ at the panel grid points, $\mu$ is the mass ratio, $\lambda$ is the non-dimensional dynamic pressure, $\gamma$ is the ratio of specific heats and $M_\infty$ is the Mach number.

Equations (2.5) and (2.6) were integrated in time using an Euler implicit method (1st order accurate in time). Besides, using a large number of structural grid points reduced the effects of translating fluid values to structural nodes.

For the **fluid equations**, the dynamics are governed by the 2-D Euler equations for inviscid flow. Assuming no grid deformation and the integral form of the Euler equations reduce to the following, an overall vector of flow variables $\mathbf{U}(t)$ was produced by collocating variables at each spatial cell location to the corresponding column vector location, using row-by-row ordering (for the theoretical development: [3]). After separating the spatial and the time derivatives, spatial terms were grouped to form a nonlinear operator $R$ acting on the set of fluid variables:

$$\frac{d\mathbf{U}(t)}{dt} = R[\mathbf{U}(t)] \tag{2.7}$$

While the structural grid used 141 nodes along the solid surface, 116 nodes were extended to the freestream. The spacing of the grid in the normal direction was determined to increase geometrically from the solid wall, while in the streamwise direction was determined as constant ($\Delta_{wall} = 0,0125$).

Through the transpiration boundary condition, the structural system of (2.5) was loosely coupled to the fluid system (2.7). In order to avoid time-lagging errors that could arise in the coupling scheme, an extrapolated panel pressure value was used ($\mathbf{P}_{ex}^{n+1}$):

$$\mathbf{P}_{ex}^{n+1} = \mathbf{P}^n + \Delta\mathbf{P}^n = 2\mathbf{P}^n - \mathbf{P}^{n-1} \tag{2.8}$$

Here, the fluid system dominated the computational time required to produce solutions for the coupled system. Thus, a ROM of the flowfield was coupled to the full-order structural model in order to produce an accurate aeroelastic panel response with less computational expense.

As dynamic cases moving grids require special treatment in the POD/ROM formulation, the solution domain was divided into three sections to facilitate the use of POD with the moving transonic shock. The solution domain was divided into three regions: *section I* or far field, *section II* or near field, and, *section III* or shock region. The section II had been solved more frequently than the far field to update the internal boundary shared with section III. In its turn, the shock region contained the flow over the panel and confines entirely the moving shock of interest. No overlap is included in this decomposition.

Thus, to analyze the POD/ROM/DD three smaller fluid problems were solved (see Figure 2.4). These problems were linked by internal boundaries. For this fluid variable, the full-system

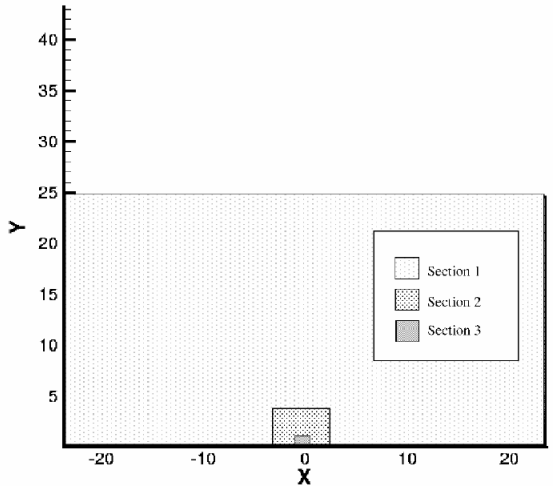


Figure 2.4: Domain Decomposition (DD) used in the POD/ROM/DD approach. [3]

dynamics in (2.7) are depicted hereafter: $\frac{d\mathbf{w}}{dt} = R(\mathbf{w})$. POD produced a linear transformation $\Psi$ between the full-system solution $\mathbf{w}$ and the reduced-order solution $\hat{\mathbf{w}}$:

$$\mathbf{w}(t) = \mathbf{W}_0 + \Psi\hat{\mathbf{w}}(t) \tag{2.9}$$

Where: $\Psi = SV \rightarrow S^T SV = V\Lambda$ ($\Lambda$ is the diagonal matrix of eigenvalues). In the study presented, both implicit and explicit time-accurate methodologies were used: ***(a)*** *implicit* for the nonshocked regions (sections I and II), and, ***(b)*** *explicit* for the shock region because the moving discontinuity was a too strong nonlinearity for the first scheme. Nevertheless, some constraints were modeled with a implicit POD/ROM scheme. They were introduced to enforce smoothness on the internal boundary between adjacent domains ($\mathbf{U}_{S1}$ and $\mathbf{U}_{S2}$ with a shared internal boundary $\Gamma$). Two constraints options were considered:

$$C^1(\mathbf{U}) = \sum_{k\in\Gamma} \mathbf{U}(k)_{S1} - \mathbf{U}(k)_{S2} = 0 \tag{2.10}$$

$$C^2(\mathbf{U}) = \sum_{k \in \Gamma} [\mathbf{U}(k)_{S1} - \mathbf{U}(k)_{S2}]^2 = 0 \tag{2.11}$$

On one hand, the first constraint was similar to requiring the mean difference to be zero on the boundary (approximating the $L^1$ norm). On the other hand, the second constraint was similar to requiring the variance to be zero on the boundary (approximating the $L^2$ norm). While the first constraint could be precomputed and does not require updates during the time integrations, the second constraint required that the Jacobian was reformed for each iteration.

This article is considered of interest because it exemplifies a possible treatment of an Aeroelastic problem through ROM. Although, this is an old article which means that the topic has probably seen multiple updates, it can easily be seen that the strategy of ROM does not need to apply the whole problem.

# 3 Methodology

## 3.1 Selection of a Methodology

In order to analyse the different methodologies, we have worked in an Advection-Diffusion problem. So a one-equation problem which it is faster solved, so it allows us to focus on the ROM analysis. In this section, only the conclusions of the analysis are presented. If the reader wants to follow all the process, please find it presented in the Annex A.

After the study performed during this article workout, one can conclude that the success of the POD can be easily proved. Besides, since the solver used to compute a problem by POD is not modified, it is just needed to add some lines of code to reduce the problem. Thus, it is calmly introduced in an already coded problem. In addition, Azaïez et al. in [21] have shown with some relevant numerical experiments that the recursive POD is computationally more accurate than the PGD for multivariate functions, recursively expanding the modes retained in the previous step.

On the contrary, some more efforts are needed to modify a problem to use the PGD. Furthermore, although it is really fast to achieve solutions in the online stage, it can be considered necessary only if a lot of cases of one problem are expected to be computed. Another really interesting application could be the use of the PGD to perform interactive computations or interactive training [17]. If the problem is not considered to be highly demanded, it could be better to consider a POD approach.[22]
Extending the PGD analysis, it is really interesting its application by a Parametric approach. The above-mentioned approach with the inclusion of the boundary conditions as extra-coordinates of the problem (either Neumann and Dirichlet) let the user to study a whole generic problem with only one robust offline computation.
Nevertheless, the study of the effectiveness of the Stopping Criterion must be deeply analyzed in order to ensure the convergence.

Talking about the approach DMD, it is been seen that it is necessary a film of snapshots to create an effectively reduced model. So, the DMD can be an interesting alternative to POD or PGD when the solution has to be solved also in a time domain. The biggest advantage of the DMD is its ability to define the dimensions of the reduced model to achieve a predefined tolerance.

After the analysis, a final question can be considered: a strategy for Aeroelasticity

problems using ROM is possible. Particularly, introducing a general ROM to a multidisciplinary problem could not be the most effective methodology. But, introducing ROM to each (or selected) one-discipline subproblem could worth the extra work of the engineer.
An interesting approach has been presented by Xia et al., in [23], producing a high-quality representation of the indicator function of different phases of the material using a combination of POD and PGD.

To conclude, after this first analysis, the POD methodology has been selected to be introduced in an Aeroelastic solver. This technique has been combined with a Greedy algorithm and a Kriging interpolation to the final implementation in a solver from ONERA. The proposed methodology is presented in the following sections.

## 3.2 Methodology Applied to a Single-Equation Problem

In this section, the notation has been changed in comparison to the one used in the Ch. 2. This change is implemented in order to facilitate the comprehension of the Aeroelastic problem.
For the construction of the final algorithm, it is necessary to present three techniques that have been used in this project: the POD concept (just a reminder), the Greedy Algorithm (to construct the RB) and the Kriging Interpolation (to fast interpolate in the online stage).

### 3.2.1 POD

Just to ease the comprehension of the implementation of the methodology explained in the Ch. 2, the POD (formerly called *PCA*) is reformulated in the example case of a linear parametric problem. Firstly, the following linear parametric problem is considered:

$$\mathbf{M}(\xi) \cdot X(\xi) = \mathbf{f}(\xi) \tag{3.1}$$

Where $\xi$ is a known parameter (or combination of parameters). The problem is solved for different values of $\xi$ and the solution $X(\xi)$ is saved in the so-called Observation Matrix. In other words, the columns of the observation matrix ($\mathbf{O}$) are the solution of the linear problem for each $\xi^{(i)}$ ($i$ the current observation (or sample). Schematically:

$$\mathbf{O} = \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ X(\xi^{(1)}) & X(\xi^{(2)}) & \dots & X(\xi^{(nObs)}) \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}$$

Then, by an orthogonal transformation (POD), we can decompose this matrix in three:

- **L**: Left Singular vectors of **O**.
- $\Sigma$: Singular values of **O**.
- **R**: Right Singular vectors of **O**.

$$\mathbf{O} = \mathbf{L} \cdot \Sigma \cdot \mathbf{R}^T \tag{3.2}$$

In order to compute the reduced matrix, we are just taking into account until the $k$ order (the number of reduction modes):

$$\mathbf{O}_r = \mathbf{L}[:\ ,\ k] \cdot \Sigma[k\ ,\ k] \cdot \mathbf{R}^T[k\ ,\ :] \tag{3.3}$$

In our problem, we are interested in the matrix **L** (Left Singular Vectors).

### 3.2.2 Greedy Algorithm

The construction of the Reduced Base (RB) is the crucial point of the Offline Phase. This phase is normally so-costly so a reliable algorithm must be implemented.
Different methods can be used in order to compute the matrix **L** of (3.2). In our approach, we have selected the Greedy Algorithm.

It is necessary to remind that the problem of interest is always $\mathbf{M}(\xi) \cdot X(\xi) = \mathbf{f}(\xi)$. In this example, a two-component parameter $\xi$ is considered: $\xi = (\xi_x \quad \xi_y)$ (cf. Figure 3.1).

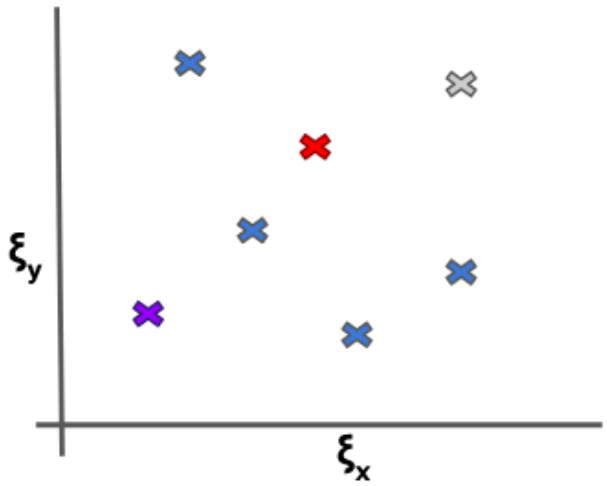


Figure 3.1: Scheme of the selected candidates during a Greedy Algorithm process.

In the Figure 3.1, it is possible to observe the mapping of candidates that are treated.

A Greedy Algorithm is based in the analysis of each candidate each iteration, choosing the worst point and adding its exact solution to the base.
The proceeding scheme can be seen in the algorithm 2. First of all, a random candidate is selected ($\xi^{(1)}$). This first point (purple point in Figure 3.1) is the first candidate where the real solution is calculated: $X(\xi^{(1)}) = \mathbf{M}^{-1} \cdot \mathbf{f}$. Then, the *SVD* is applied and we save the left singular vectors (matrix $\mathbf{L}$) of this orthogonal transformation. In order to do the latest action, the following observation matrix is considered:

$$\mathbf{O}[:, \text{columns} = k] = \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ X(\xi^{(1)}) & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}$$

Then the orthogonal transformation can be expressed as: $[\mathbf{L}, \_, \_] = svd(\mathbf{O})$. This matrix $\mathbf{L}$ is the same dimension as $\mathbf{O}$ -where $k$ is always the number of Reduction modes selected. After the computation of this first version of the Reduced Base, the loop begins: for $i = 2 : (k-1)$, the residual (cf. equation (3.4)) of the each candidate $j$ (each cross in the map) is evaluated:

$$\mathbf{r}(\xi^{(j)}) = \mathbf{M}(\xi^{(j)}) \cdot \left[ \sum_{i=1}^{nModes} \alpha_i(\xi^{(j)}) \cdot L_i \right] - \mathbf{f}(\xi^{(j)}) \tag{3.4}$$

The candidate $j$ where this result is maximum is considered as the point $\xi^{(2)}$ (*grey* as we are in the second iteration). This point is added to the observation matrix

$$\mathbf{O}[:, \text{columns} = k] = \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ X(\xi^{(1)}) & X(\xi^{(2)}) & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}$$

and then the matrix $\mathbf{L}$ is updated using always $[\mathbf{L}, \_, \_] = svd(\mathbf{O})$.

Since the real solution is added always to the RB, the residual of the candidates used to create this base will be always zero. Thus, if $nCandidates > k$ the same point will never be used to times. To complete the explanation of the candidates shown in the Figure 3.1, the point in red represents the selected point in any iteration $i$ while the blue points are the candidates evaluated but which residual has been never the worst of the set.

### 3.2.3 Kriging Interpolation

Before treating what is a Kriging Interpolation, it is necessary to follow the arguments that lead us to this solution.

**Algorithm 2** POD training algorithm with Greedy Algorithm. [6]

**Input:** Mapping of $(\xi_x \quad \xi_y)$..

1. Select randomly a first sample: $\xi^{(1)}$
2. Solve the problem: $X(\xi^{(1)})$
3. Build the corresponding Reduced Base (RB): $\mathbf{L}$
4. For $i = 2, \cdots, k$
   (a) Solve: $\xi^{(i)} = \underset{\xi \in [nCandidates]}{\arg\max} \left|\left|\mathbf{r}(\xi^{(j)})\right|\right|$
   (b) Solve the problem: $X(\xi^{(i)})$
   (c) Rebuild $\mathbf{L}$ based on the samples: $\{X(\xi^{(1)}), X(\xi^{(2)}), \cdots, X(\xi^{(i)})\}$

**Return: L**

First of all, once the Reduced Base is created it is necessary to question how can the Online phase Algorithm can be implemented. From the bibliography [10], it is shown that the reconstruction of a linear parametric problem can be considered as follows:

$$\Phi^{-1} \cdot \mathbf{M}(\xi^{new}) \cdot \Phi \cdot X(\xi^{new}) = \Phi^{-1} \cdot f(\xi^{new}) \tag{3.5}$$

And after this formulation two questions emerge:

1. What is $X(\xi^{new})$?
2. What is $\Phi$?

The first question can be solved expressing $X(\xi^{new})$ as a lineal combination:

$$X(\xi^{new}) = \sum_{i=1}^{nModes} \alpha_i(\xi^{new}) \cdot \Phi_i \tag{3.6}$$

Then, the second question it is simple. $\Phi$ is the so-mentioned RB $\mathbf{L}$ created with the combination of POD and the Greedy Algorithm.

Finally, it is possible to notice that all the doubts have been clarified except for one that has appeared in the last formulation: how can $\alpha_i(\xi^{new})$ be computed?

The latest is not a simple question. Two options are considered: ***(a)*** using a Reduction Problem Resolution (so basically solving the equation 3.5) and, ***(b)*** interpolating the coefficient at the points $\left(\xi^{(1)}, \xi^{(2)}, \cdots, \xi^{(i)}\right)$. Since for the first option a modification of the code should be performed, this possibility is rejected.

Thus, at this point the option of interpolation the solution is explored. One possibility is to use the so-called *Substitution Models* [24]. Within, two big families can be found: the linear regression and the interpolating models. The first family does not allow us to compute the real point so it is also rejected.

Since we are working with interpolation models, the notation of our linear combination has slightly changed:

$$X(\xi^{new}) = \sum_{i=1}^{nModes} \hat{\alpha}_i(\xi^{new}) \cdot \mathbf{L}_i \tag{3.7}$$

Finally, the selected method is Kriging [25]. This methodology is largely used in Aerospace Engineering because it uses a Gaussian formulation that allows us to model the values by the mean value and prior variance. Thus, if it is possible to estimate the variance of a solution, it feasible to know the error of interpolation. And, since $\hat{\alpha}_i$ is a Gaussian Vector, $X$ it is so too:

$$X(\xi^{new}, \sigma) = \sum_{i=1}^{nModes} \hat{\alpha}_i(\xi^{new}, \sigma) \cdot \mathbf{L}_i$$

In Figure 3.2, a Kriging interpolation is shown. The red line represents the interpolated values, the blue dashed line calls for the real solution and the dark shadowed zone represents the interval of confidence of the interpolation. Thus, the shadowed zone can be seen as the interpolation error committed at each point. As expected, where a real solution is added, no error is committed.

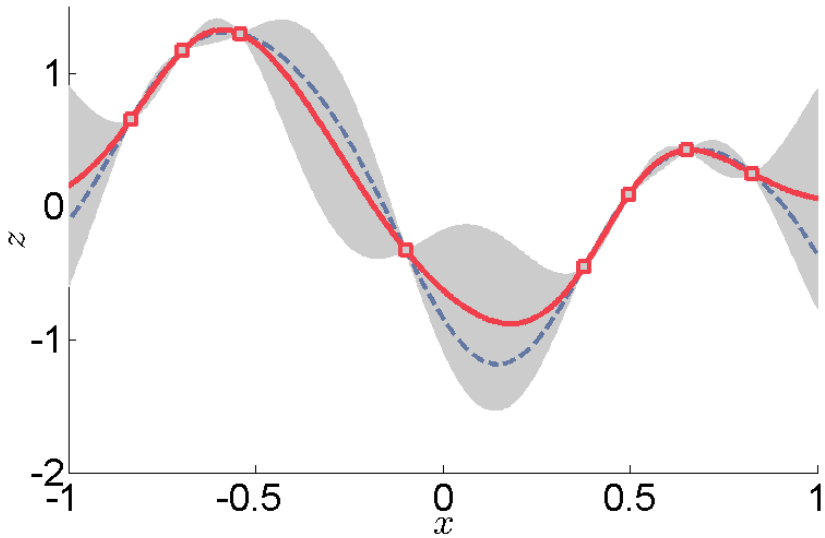


Figure 3.2: Example of 1D data interpolation by Kriging, with confidence intervals. [4]

Then, as it has been introduced before, the solution of the Kriging follows a Gaussian distribution: $\hat{\alpha} \sim \mathcal{N}(\mu_{\hat{\alpha}}, \sigma^2_{\hat{\alpha}})$. In order to compute the mean value and the variance of this

variable respecting the projection to the RB, we should consider the following construction of the information:

$$\begin{cases} \mu_{\hat{\alpha}} = \sum_{i=1}^{nModes} \mu_{\hat{\alpha}_i} \cdot \mathbf{L}_i \\ \sigma^2_{\hat{\alpha}} = \sum_{i=1}^{nModes} \sigma^2_{\hat{\alpha}_i} \cdot (\mathbf{L}_i)^2 \end{cases} \tag{3.8}$$

## 3.3 Methodology Applied to Two-Equation Problem

In this section, we are treating the problem of the Static Aeroelastic problem. Since it is a Static problem, only the coupling in space is considered. So, for the coupling with both problems (or equations) only the mapping in space is considered.
In order to introduce the topic, *Aeroelasticity* is a field that couples inertial, elastic, and aerodynamic forces. Obviously, in the industry there are already solvers used in a Preliminary Design that involves a reduced problem, but, here, we are interested in some new approaches in order to do this exploration of the domain in real-time (Kriging and Greedy Algorithm).
It is really convenient to specify that it is not an optimisation problem (there are no differential codes). The idea is a code for exploring the design space in real time.
In the Table 3.1, we define the new two problem thanks to the problem used during the whole methodology process: $\mathbf{M}(\xi) \cdot X(\xi) = \mathbf{f}(\xi)$.

Table 3.1: Linking the theory problem components to the new problems.

| **Problem** | **Aerodynamic** | **Structural** |
|---|---|---|
| **M** | **A** | **K** |
| X | $\Gamma$ | U |
| **f** | **b** | $\mathbf{F}_S$ |

Developing the whole equations, it results in the following two expressions:

**Aerodynamics**

$$\mathbf{A}(\xi) \cdot \Gamma(\xi) = \mathbf{b}(\xi) \tag{3.9}$$

**Structures**

$$\mathbf{K}(\xi) \cdot U(\xi) = \mathbf{F}_S(\xi, \Gamma) \tag{3.10}$$

The Aerodynamic System is the AIC matrix multiplies the circulation vector, equals to the RHS vector. The Circulation vector follows a Gaussian distribution since a Kriging model is used to interpolate the system. For the Structures System, the Stiffness matrix multiplies the Displacement vector and they equal to the force term that depends on the Aerodynamic problem and the point analysed. The Displacement vector is also a random variable.
Here, it is simple to observe that the Structural System depends on the Aerodynamic System

since the Structural forces applied to the system are related to the fact that the global system (the aircraft) is moving through the air.

To introduce the methodology to both equations, the $\alpha$ parameter used all the time to represent the Gaussian variable that we interpolate with the Kriging, for Aerodynamics is also $\alpha$ and for Structures is $\beta$. Then, the Reduced Base have created thanks to the $\Gamma$ results for Aerodynamics resulting in the base $\mathbf{G}$ and, for Structures, the $U$ results are used to compute the RB $\mathbf{U}$.

$$\begin{cases} \Gamma(\xi,\sigma) = \sum_{i=1}^{nModes} \hat{\alpha}_i(\xi,\sigma)\cdot \mathbf{G}_i \\ U(\xi,\sigma) = \sum_{i=1}^{nModes} \hat{\beta}_i(\xi,\sigma)\cdot \mathbf{D}_i \end{cases} \tag{3.11}$$

In order to compute the Aeroelastic relation, a transfer matrix is used in order to be able to determine the external structural forces from the Aerodynamic forces. This stage couples the Aerodynamics with the Structures.
This relation can be expressed as:

$$\mathbf{F}_S = \mathbf{H}^{-1}\cdot \mathbf{F}_A \tag{3.12}$$

For the coupling in space, we interpolate the mesh for converging the Reduced Problem inside the Greedy Algorithm, so when creating the Reduced Base.
So, for each candidate evaluation, the algorithm 3 is performed. It is necessary to insist that it is performed for the reduced problem.

After all these processes, the final algorithm for the Offline Phase is presented in the algorithm 4.

For the Online Phase, it is also necessary to interpolate in the Kriging functions following the equations 3.8 after defining the interesting point $\xi^{new}$.

**Error identification**

Here, it is sought to identify possible sources of errors in the code proposed. On one hand, it exists the error due to the base truncation in the POD approach. On the other hand, one can identify the error of interpolation occurring in the Kriging.

Consequently, one can drive these errors by modifying the number of samples: higher the number of samples used, more important influence of the Kriging process into the error. But, lower the number of samples, more influence of the POD process.
Nevertheless, one could improve the error due to Kriging by adding more points to the functions once the Offline process would be already calculated. This should last with a reduction on the error without deteriorating the computation time.

**Algorithm 3** POD training algorithm with Greedy Algorithm.

**Input:** Aerodynamic mesh file and the point $\xi$.

1. Compute the value of $\mathbf{A}(\xi)$ and $\mathbf{b}(\xi)$.
2. Solve the reduced Aerodynamic problem.
3. Transform the reduced problem to a full-rank problem, using the Aerodynamic RB $\mathbf{G}^{-1}$.
4. Transfer the Aerodynamic forces applied to each node, to forces applied to each Structure nodes.
5. Transform the full-rank problem to a reduced problem, using the Structural RB $\mathbf{U}$.
6. Solve the reduced Structural problem.
7. Evaluate the error: the maximum value between
$$\frac{||\Gamma - \Gamma_0||}{||\Gamma_0||} \tag{3.13}$$
and
$$\frac{||U - U_0||}{||U_0||} \tag{3.14}$$
8. If the error is bigger than the tolerance, modification of the Aerodynamic nodes according to the displacements and the rotation computed in $U$. Re-do the whole algorithm.

**Return:** $\Gamma$ and $U$

**Algorithm 4** Offline Global Algorithm

**Input:** Mapping of $\xi$..

1. Select randomly a first sample.
2. Solve the Aerostructural Problem: $\Gamma$ and $U$.
3. Build the corresponding Reduced Base (RB): **G** and **U**.
4. For $i = 2, \cdots, k$
   (a) For $j = 1, \cdots, nCandidadates$
      i. Algorithm 3
      ii. Solve: $\xi^{(i)} = \underset{\xi \in [nCandidates]}{\arg\max} \left|\left|\mathbf{r}(\xi^{(j)})\right|\right|$
   (b) Solve the problem: $\Gamma(\xi)$ and $U(\xi)$.
   (c) Rebuild **G** and **U**.
5. Create two Kriging of $k$ functions each: $\alpha$ for Aerodynamics and $\beta$ for Structures.

**Return:** Kriging $\alpha$ and $\beta$

# 4 Results

## 4.1 Solver disaggregation

The Aerodynamic analysis is performed by using a Vortex Lattice Method (VLM). The VLM models the lifting surfaces of an aircraft as an infinitely thin sheet of discrete vortices to compute lift and induced drag. The influence of the thickness, the viscosity is neglected.
Consequently, giving a geometry of a wing, the VLM solver computes the circulation, the Aerodynamics forces and the Pressure distribution.
After the Aerodynamic analysis, the Aerodynamic load at each (Aerodynamic) node are transferred to the Structural nodes. Once these external loads are transferred, the third stage is run. This stage uses a Finite Element Method (FEM) for computing the structural behaviour of the wing. The FEM approach used in this problem is the Discrete Kirchoff Triangles (DKT) of plates following an elastic linear homogeneous isotropic behaviour. If the hypothesis of plane stress is applied ($\sigma_{zz} = 0$), -in the local stiffness matrix- the membrane strain and bending strain are decoupled. The approach taken into account in this study uses the theory of the virtual work principle. For more information, see [26].
To finish with the current stage, the mesh is deformed with the results of the FEM and the Aerodynamic code is launched again. Thus, a loop process is originated until a determined tolerance is achieved.

The hypothesis made to each solver are:

- For VLM:
  - The flow field is considered incompressible, inviscid and irrotational.
  - The influence of thickness on aerodynamic forces are neglected.
- For FEM:
  - Small deformations.
  - Small displacements.
  - The plans do not consider the geometrical curve of the mean structures.

## 4.2 Case of Study

The results have been achieved for an Airbus wing (see Figure 4.1), modifying the first approach of a joined-wing.

The code given by the ONERA that parametrizes the wing (see Figure 4.1) uses a lot of parameters. The parameters fixed (their value does not change) are the angle of attack, the

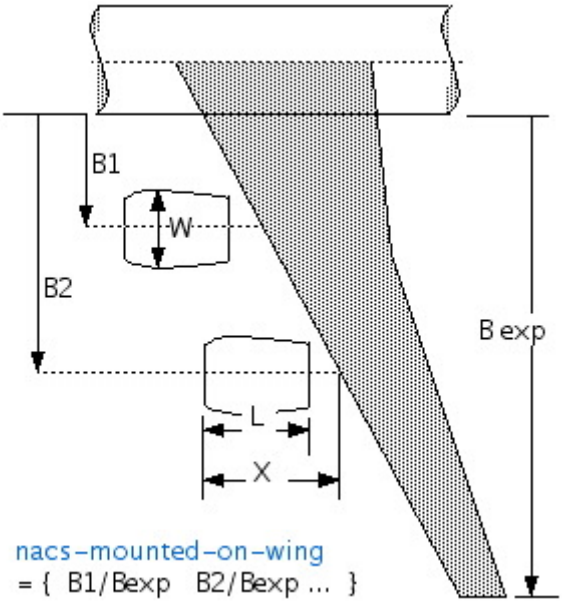


Figure 4.1: Scheme of the wing used in the current parametric study. [5]

Mach number, the Young modulus, the Poisson's ratio, the distance of the fuselage over the wing, the sweep angle, the dihedral angle, the number of ribs and the altitude (11000 meters). The parameters that can change their values between given bounds are the skin thickness, the ribs thickness, the leading edge and trailing edge spar thickness, the span and the wing surface. So, a point of interest (PoI) of our problem is defined by 6 values.

## 4.3 First study: Reduced domain

At first approach, the domain chosen is reduced in order to verify the capability of the solution to reproduce the tendencies of the result.
These results are applied in a solver in which the convergence of the Reduced Solution is not implemented.

In order to verify and to validate the POD procedure, different cases for the offline code (see Table 4.1) and different configurations for the online code (see Table 4.2) have been analysed.

In order to do the analysis, for each configuration and case, several aspects must be ensured. The idea is to verify that the real value is comprised between the two variance values (see Figures 4.2, and 4.3). Furthermore, it has been verified that the node where Von Mises stress is maximum is the same for all the cases.
In Figure 4.3, an artificial deformation ratio has been applied to give the deformed wing image.

From these figures, some comments must be done. Firstly, while the real value of the

Table 4.1: Cases analysed for the offline procedure.

| ID | # Samples | # Candidates |
|---|---|---|
| *1* | 5 | 10 |
| *2* | 5 | 50 |
| *3* | 5 | 100 |
| *4* | 15 | 30 |
| *5* | 15 | 50 |
| *6* | 15 | 100 |
| *7* | 25 | 30 |
| *8* | 25 | 50 |
| *9* | 25 | 100 |
| *0* | 65 | 70 |

Table 4.2: Configurations analysed for the online procedure.

| Configuration | *1* | *2* | *3* | *4* |
|---|---|---|---|---|
| **Skin thickness (m)** | 0.028 | 0.029 | 0.030 | 0.031 |
| **Ribs thickness (m)** | 0.009 | 0.014 | 0.010 | 0.015 |
| **LE spar thickness (m)** | 0.017 | 0.019 | 0.022 | 0.025 |
| **TE spar thickness (m)** | 0.009 | 0.013 | 0.011 | 0.015 |
| **b (m)** | 64,75 | 66.83 | 73.20 | 75.65 |
| **S** ($m^2$) | 443.56 | 449.70 | 444.44 | 500.21 |

circulation is comprised between the value of the two variances, for the displacement this does not happen in any case evaluated. Secondly, if a comparison is performed between the POD solution and the *real* solution, the error computed is usually less than a 15%. The only value that the POD cannot predict in an acceptable accuracy is the minimum value of the equivalent Von Mises stress.

To conclude the study of the POD-RB, five random configurations have been tested for each case and compared with the original code results. This study drives to the same conclusions: ***(a)*** using only 5 iterations in the Greedy algorithm is not enough despite the fact that the results are not truly inaccurate; ***(b)*** the method is not able to well approximate the inferior limit in the Von Mises stress, but it is able to approximate highly reliable both maximum and average stress; ***(c)*** the results are a lot better than the ones achieved with the previous analysis (which were limit cases); and, ***(d)*** in the displacement calculation, the real value is normally out of the variance bounds, but the values are not distanced.

Finally, the Kriging procedure is giving acceptable variance values, but the real solution is not localised between the limits (in the case of the displacement). This can only be explained by a lag of accuracy of the POD-RB. A source of error could be the construction of the RB.

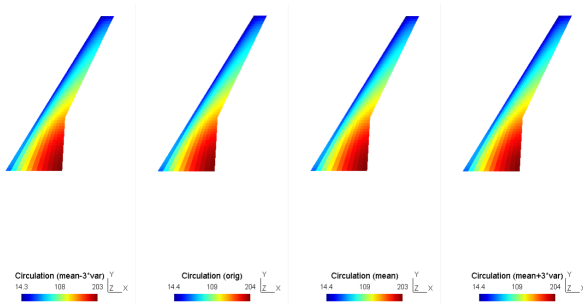


Figure 4.2: Circulation comparison for Conf 1 and Case 10.

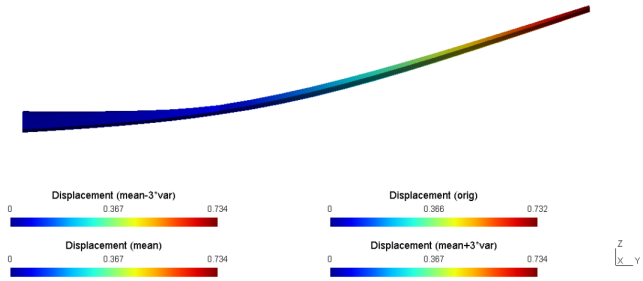


Figure 4.3: Displacement comparison for Conf 1 and Case 10.

Since it is a coupled problem, the real solution added to the RB could not reduce the error to zero. For sure, the error in that point is reduced, but the coupling of the problem can produce a non-zero error indicator. In the original Greedy algorithm, this problem does not exist because the same point can be added several times.

## 4.4 Second study: Enlarged domain

Drawing on the need for reformulating the problem due to the implementation of the convergence in the Greedy Algorithm, the domain of study has been enlarged. At this point, two limit wings have been considered: the A320 wing and the A380 wing (cf. Table 4.3).

Table 4.3: Limits of the parameters.

| $\xi$ | *Min* | *Max* |
|---|---|---|
| **Skin thickness (mm)** | 1.45 | 30 |
| **Ribs thickness (mm)** | 1.45 | 15 |
| **LE spar thickness (mm)** | 4.35 | 90 |
| **TE spar thickness (mm)** | 4.35 | 90 |
| **b (m)** | 34 | 80 |
| **S** ($m^2$) | 122 | 845 |

In this study, it is also considered a method to augment the number of points of the Kriging. Once the RB are computed, they are used to compute more approximate points to have more points in the Kriging and reducing the potential error of the interpolation.
Then, a study defined by $nSamples = 10$, $nCandidates = 50$ and $nKriging = 150$ is performed.
In Figures 4.4 and 4.5, the solution for the circulation and displacement are presented (respectively). They have presented 4 solutions: both solution for variances (limit superior and inferior), the mean solution (so the solution of the Kriging Interpolation) and the solution of the original code. It can be seen that both solutions are well classified within the limits of the error of interpolation. The point used for this simulation is presented in Table 4.4.

Table 4.4: $\xi$ of interest.

| Skin thickness (mm) | Ribs thickness (mm) | LE spar thickness (mm) | TE spar thickness (mm) | b (m) | S ($m^2$) |
|---|---|---|---|---|---|
| 20 | 10 | 50 | 40 | 55 | 480 |

After this study, it is seen that the savings in time are really considerable: after an offline computation of RB that runs for 7h30min, the real-time solution needs only 0.3 $s$ to estimate a solution while the original code needs 60 $s$. Nevertheless, the precision of the interpolation is not accurate. Probably the domain it was too large for the number of modes ($nSamples$) used to reproduce the estimate solution.

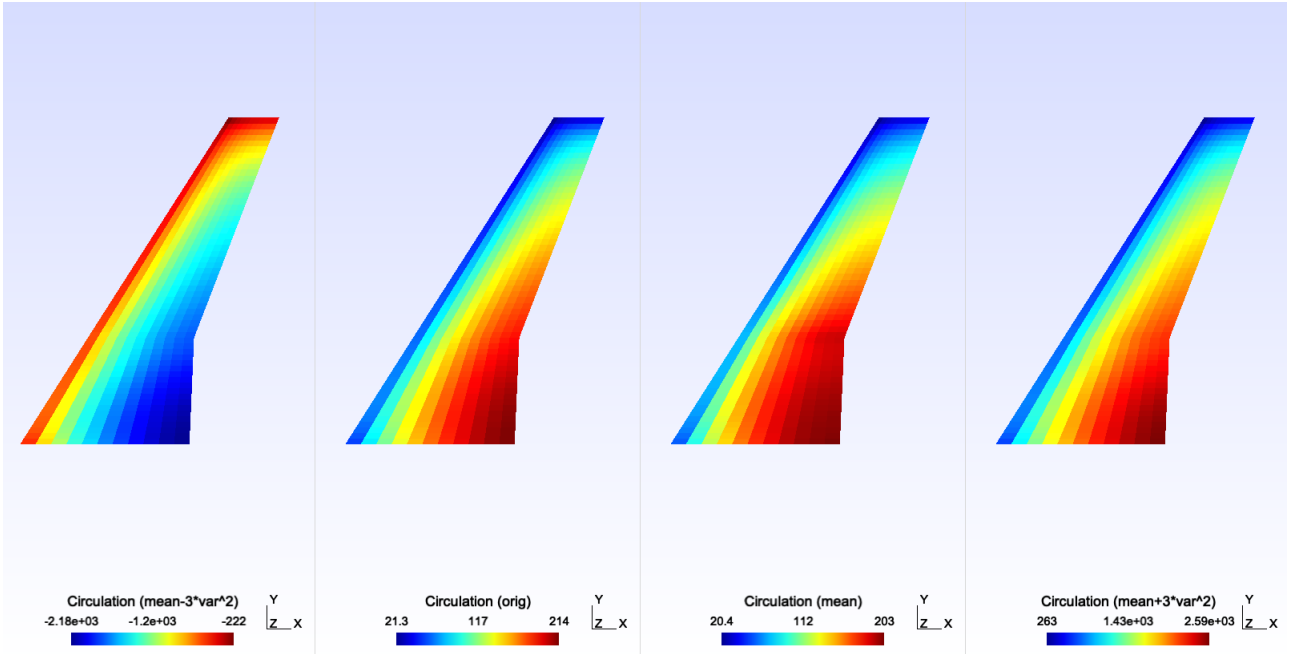


Figure 4.4: Circulation.

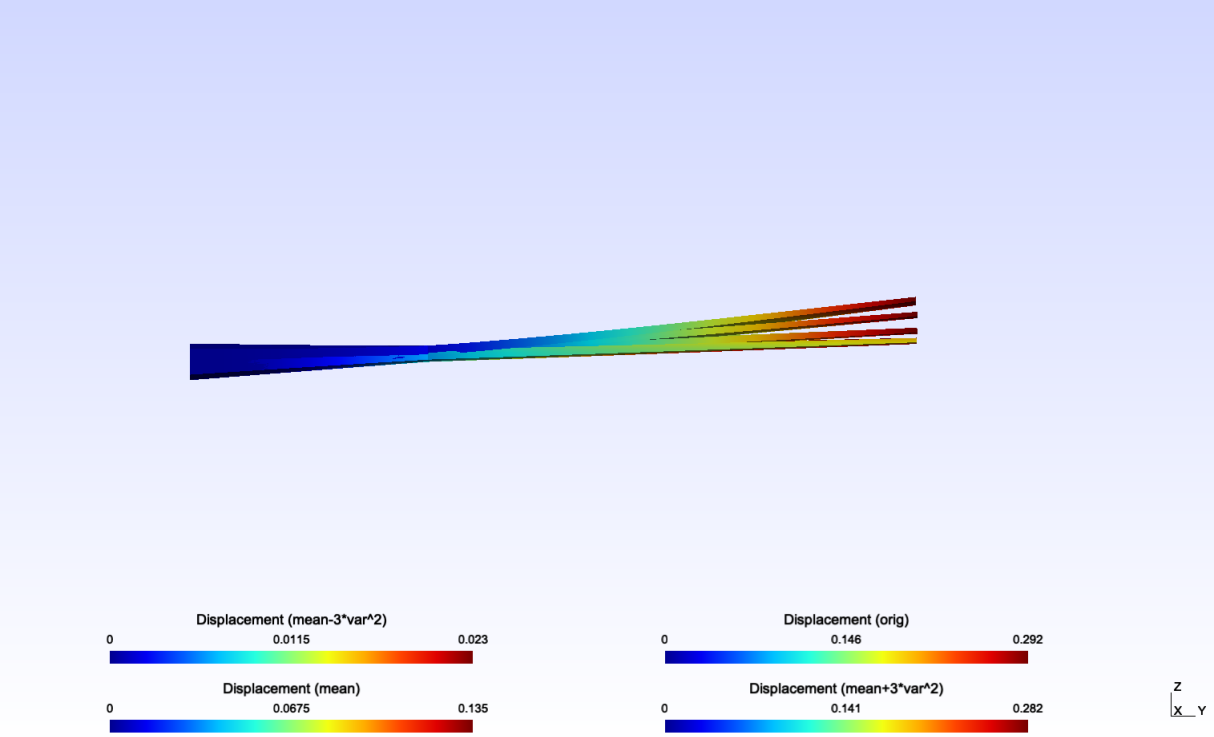


Figure 4.5: Displacement.

# 5 Conclusions

To conclude, we can disaggregate two error nature: one due to the action of Reduction the Problem (we are not including information that is part of the solution) and another one due to the interpolation method (which thanks to the Kriging Model can be estimated).

To improve the first source of errors, we identify the importance of the number of samples. For improving the results, we could augment the order of the Reduced Problem, we could reduce the problem (which is not really interesting in a Preliminary Stage) or substituting the fixed-point iteration method of the Greedy Algorithm.

To improve the Interpolation Error, we could augment the number of candidates. This added to a higher number of samples will results in more point to the Kriging and more points analysed while building the Reduced Base. Obviously, we could also enlarge the number of Kriging points.

Finally, the gain in time, as said, it is clear. And we could imagine a situation that this application could really be useful. If we imagine a room with Aerodynamic and Mechanical Engineers, the solutions and possible discussions could be evaluated in real time, allowing the Preliminary to progress faster. Thus, it will be interesting to be applied in a problem if only a lot of cases must be simulated. If not, it does not worth to perform such an expensive offline calculus.

***GITHUB LINKS***

The PIR's GitHub or go to the next *url*: `http://github.com/mid2SUPAERO/PIR_CHANDRE_ROM`.
The internship's GitHub or go to the next *url*: `http://github.com/mid2SUPAERO/Stage_CHANDRE_POD`.

# 6 Future Steps

For future steps, two jobs should be done.

Firstly, a parametric study of the influence of $nSamples$, $nCandidates$ and $nKriging$ should be performed. With this analysis in the enlarged domain, the influence on the quality of the RB driven by these parameters is searched.

Secondly, this methodology should be applied to more industrial solver. This implementation could not be treated finally in this project, but it should be really interesting to perform a study in *OpenAeroStruct* (OAS).
*OAS* is an open-source low-fidelity Aerostructural analysis and Optimisation tool made by the University of Michigan. The analysis is based on a couple of system between a VLM code and a simplified wingbox FEM. For the Optimisation, the solver uses the NASA software *OpenMDAO*. In this version, in which the wingbox is introduced and, for a basic case (only cruise configuration), it is possible to define up to 17 interesting parameters: (the parameters classified as *Mission Requirements* could be classified as one of the other 3 types)

- 4 for Aerodynamic
  1. $C_{L0}$
  2. $C_{D0}$
  3. Percentage of laminar flow: $k_{lam}$
  4. Point the maximum camber: $C_{maxt}$
- 4 for Structures
  1. Young modulus: $E$
  2. Shear modulus: $G$
  3. Poisson's ratio: $\nu$
  4. Material density: $\rho_m$
- 3 for Fuel data
  1. $C_T$
  2. Fuel Mass
  3. Fuel density: $\rho_f$

- 6 for Mission Requirements

    1. Speed: $v$
    2. Altitude: $h$
    3. Angle of attack: $\alpha$
    4. OEW: $W_0$
    5. Reserve fuel requirement: $W_{freserve}$
    6. Load factor

Just as a comment, with the speed and the altitude it is possible to compute the Mach number, the Reynolds number and the speed of sound.

In order to perform this upgrade, it is necessary to understand both codes and to search for interesting information. In other words, within all the analysis (both software organise the information inside *Python* dictionaries concerning the different parts. There are dictionaries for Aerodynamics, for Structures, for Aerodynamic Performance, for Structures Performance, for Global Performance or for Coupled Parameters, among a wide range of others. During this project, the parameter considered as the most important have been already located:

```
# A = prob['AS_point_0.coupled.aero_states.mtx_rhs.mtx']
# b = prob['AS_point_0.coupled.aero_states.mtx_rhs.rhs']

# G = prob['AS_point_0.coupled.aero_states.horseshoe_circulations.horseshoe_circulations']
# Fpanel = prob['AS_point_0.coupled.aero_states.panel_forces.panel_forces']
# rho = prob['prob_vars.rho']
# Fpoints = prob['AS_point_0.coupled.aero_states.panel_forces.force_pts_velocities']
# bound_vecs = prob['AS_point_0.coupled.aero_states.panel_forces.bound_vecs']
# Fa = prob['AS_point_0.coupled.aero_states.panel_forces_surf.wing_sec_forces']

# H = prob['AS_point_0.coupled.wing.def_mesh.displacement_transfer.transformation_matrix']
# Fs = prob['AS_point_0.coupled.wing.forces']
# K = prob['AS_point_0.coupled.wing.local_stiff_transformed']
# disp = prob['AS_point_0.coupled.wing.def_mesh.displacement_transfer.disp']
```

# 7 References

[1] E. Cueto, "Lecture notes in Model Order Reduction. Proper Generalized Decomposition (PGD) methods," April 2015.

[2] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations. An Introduction*. SpringerBriefs in Applied Science and Technology, 2016.

[3] D. Lucia, P. Beran, and P. King, "Reduced-Order Modeling of an Elastic Panel in Transonic Flow," *Journal of Aircraft*, vol. 40, no. 2, 2003.

[4] G. Wahba, *Spline Models for Observational Data*. SIAM, 1990.

[5] Lissys.Demon, "Geometric Specifications (in English)."

[6] D. Amsallem, "An Adaptive and Efficient Greedy Procedure for the Optimal Training of Parametric Reduced-Order Models," *International Journal for Numerical Methods in Engineering*, vol. 102, 2014.

[7] U. Baur, P. Benner, and L. Feng, "Model Order Reduction for Linear and Nonlinear Systems: a system-theoretic Perspective," *Archives of Computational Methods in Engineering*, vol. 21, pp. 331–358, 2014.

[8] R. A. Blog, "Its all about the future."

[9] M. Einasto, L. J. Liivamägi, E. Saar, J. Einasto, E. Tempel, E. Tago, and V. J. Martínez, "Principal Component Analysis," *Astronomy & Astrophysics Manuscript*, vol. 535, no. A36, 2011.

[10] J. N. Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford: Oxford University Press, 2013.

[11] A. Chattorjee, "An Introduction to the Proper Orthogonal Decomposition," *Current Science*, vol. 78, no. 7, pp. 808–817, 2000.

[12] A. Megretski, "Proper Orthogonal Decomposition," *Massachusetts Institute of Technology - Department of Electrical Engineering and Computer Science*, 2004.

[13] S. S. Ghoman, Z. Wang, P. C. Chen, and R. K. Kapania, "A POD-based Reduced Order Design Scheme for Shape Optimization of Air Vehicles," *AIAA - Structural Dynamics and Materials Conference*, no. 1808, 2012.

[14] D. Child, *The Essentials of Factor Analysis*. Bloomsbury Academic Press, 2006.

[15] B. Erichson, S. L. Brunton, J. N. Kutz, and S. Voronin, "Randomized Matrix Decompositions using R," *Journal of Statistical Software*, 2018.

[16] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[17] H. Abdi, *Discriminant Correspondence Analysis*. The University of Texas at Dallas: Encyclopedia of Measurement and Statistics, 2007.

[18] T. Isomura and T. Toyoizumi, "A local learning rule for independent component analysis," *Scientific Reports*, vol. 6, no. 28073, 2016.

[19] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Philadelphia: SIAM News Blog, 2016.

[20] F. Chinesta, R. Keunings, and A. Leygue, *The Proper Generalized Decomposition for Advanced Numerical Simulations: A primer*. SpringerBriefs in Applied Science and Technology, 2014.

[21] M. Azaiez, F. B. Belgacem, and T. C. Rebollo, "Recursive POD expansion for reaction-diffusion equation," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 3, no. 3, 2016.

[22] L. M. Valsecchi, *Reduced Order Methods for PDEs: a comparison between Proper Orthogonal Decomposition and Proper Generalized Decomposition*. PhD dissertation, Politecnico de Milano, 2015.

[23] L. Xia, B. Raghavan, P. Brietkopf, and W. Zhang, "A POD/PGD Reduction Approach for an Efficient Parameterization of Data-driven Material Microstructure Models," *Computer Methods in Material Science*, vol. 13, no. 2, pp. 219–225, 2013.

[24] G. Matheron, *Les variables régionalisées et leur estimation: Une application de la théorie des fonctions aléatoires aux sciences de la nature*. Masson, 1965.

[25] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: The MIT Press, 2006.

[26] A.-D. Kudawoo, "Eléments de plaque: modélisations DKT, DST, DKTG et Q4G," May 2011.

# A Evaluation of the ROM techniques

## Methodology

In order to exemplify this section, the problem presented in A.

### Reference Solution

Extracted from Amsallem et al., a code for this problem has been acquired. This code is necessary in order to validate and compare the processes that are tested (see algorithm 5).

**Algorithm 5** Steady Advection-Diffusion reference case algorithm. [6]

**Input:** Value of the non-homogeneous Dirichlet boundary conditions, value of the parameters and mesh data.

1. Compute matrix $\mathbf{A}_1$. It is a matrix of the second-order derivatives using a centered scheme.
2. Compute matrix $\mathbf{A}_2$. It is a matrix of the first-order derivatives using an upwind scheme.
3. Compute $\mathbf{A}$: $\kappa \cdot \mathbf{A}_1 - \mathbf{u} \cdot \mathbf{A}_2 = \mathbf{A}$.
4. Compute $\mathbf{b}$. It is a vector where the Dirichlet boundary conditions are forced. The other nodes are defined as zero.
5. Solve the problem: $\mathbf{w} = \mathbf{A}^{-1} \cdot \mathbf{b}$

**Return:** Sample vector $\mathbf{w}$

As can be seen in the algorithm just presented, the problem is solved at once. No iterations are needed.

### POD case

To perform the approach of the POD in our test case, it has been used the algorithm 6 where a Greedy algorithm is added to the POD treatment. This algorithm can be found in Amsallem et al.

The algorithm presented, it does not apply a completely random first sample. In fact, each initial parameter value has been fixed to the mean value of its range. Besides, the Neumann boundary conditions have been forced within the matrix $\mathbf{A}_2$.
Since it is a general algorithm, algorithm 6 has been applied to each candidate solution after

**Algorithm 6** POD training algorithm. [6]

**Input:** Parameters range. Here, $\mu$ represents the parameters vector.

1. Select randomly a first sample: $\mu^{(1)}$
2. Solve the HDM: $f(\mathbf{w}(\mu^{(1)}, \mu^{(1)}) = 0$
3. Build a corresponding ROB: $\mathbf{V}$
4. For $i = 2, \cdots, s$
   (a) Solve: $\mu^{(1)} = \underset{\mu \in \{\mu_1, \cdots, \mu_C\}}{\arg\max} ||\mathbf{r}(\mu)||$
   (b) Solve the HDM: $f(\mathbf{w}(\mu^{(i)}, \mu^{(i)}) = 0$
   (c) Build a ROB $\mathbf{V}$ based on the samples: $\{\mathbf{w}(\mu^{(1)}), \cdots, \mathbf{w}(\mu^{(i)})\}$

**Return: V**

solving the problem using the algorithm 5 at each candidate.

The goal of this algorithm is to train the model. Here, *train* means to build the best ROB possible in order to, then, create approximative (but faster) solutions. To achieve this objective, 1000 candidates have been used to create a huge amount of data (cases). In our case, the POD has been trained using 10 iterations (the variable $s$ in step 4 of the algorithm 6 is defined as 10). In Figure A.1 this enrichment process is presented. The idea, as explained in algorithm 6, is to create the first sample *randomly* and from the second sample until the $s$ sample, just add a column to the RB $\boldsymbol{V}$. At each iteration $s$, a SVD needs to be applied to the system. For instance, the fourth iteration is adding a column to the left-singular vectors resulting from the third POD. Thus, the Greedy algorithm is optimizing the data each iteration using a ROM.

Once the model has been trained, the desired case can be run. The way to proceed is basically transforming the solution of each case (solved with algorithm 5) by multiplying the $\boldsymbol{V}$ resulting from the algorithm 6.

### PGD case

The second method tested has been the PGD method.

### Offline stage

In this case, two approaches have been applied. The first one is the algorithm 1, already presented. Nevertheless, it is a too intrusive method and it would force us to code and to define

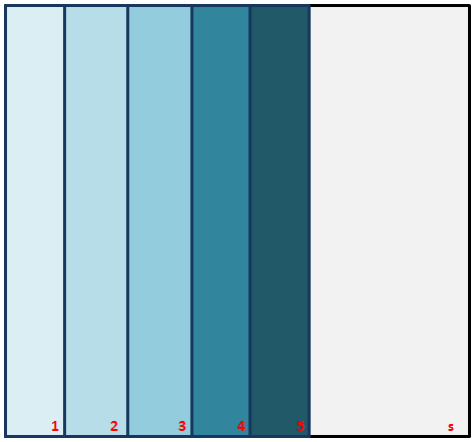


Figure A.1: Scheme for the enrichment process for POD presented in the algorithm 6.

a new PGD script for each new problem. Besides, in our specific problem, the convective terms introduce non-symmetry to the solution, which obscure the methodology.
Thus, a new approach has been used in order to achieve a more general method of the PGD methodology. Presented in the sixth chapter of Chinesta et al. book [20], the *Residual Minimization for Non-Symmetric Operators* accelerates the convergence of the problem. Briefly, in a problem defined as $\mathbf{A} \cdot \mathbf{X} = \mathbf{F}$, it is sought the solution $\mathbf{X}$ that minimizes the residual norm $||\mathbf{A} \cdot \mathbf{X} - \mathbf{F}||^2$. Notably, an Alternating Direction Scheme is also applied, see Algorithm 7. Here, $\mathbf{Re}$ represents the residual.

**Algorithm 7** Alternating Direction Scheme Algorithm for a two-parameters PGD. [20]

**Input:** The two vectors $\mathbf{R}$ and $\mathbf{S}$ for the previous iteration $n-1$.

1. Using $\mathbf{S}^{n-1}$, $\mathbf{R}^n$ is computed using a symmetric linear algebraic system $\frac{\partial ||\mathbf{Re}||^2}{\partial \mathbf{R}} = 0$.
2. Using $\mathbf{R}^n$, $\mathbf{S}^n$ is computed using a symmetric linear algebraic system $\frac{\partial ||\mathbf{Re}||^2}{\partial \mathbf{S}} = 0$.

**Return:** Vectors $\mathbf{R}$ and $\mathbf{S}$ for the current iteration $n$.

The current offline methodology requires defining the problem in a matrix way, and then a solver -which must never be modified- is used. This solver needs five constructors to work and it is provided by Chinesta et al.

Firstly, in a Matlab environment, these 5 constructors are defined as *cell structures* and are the following:

- **AA**: it is called *the operator* and its dimension are $[5 \times 4]$ matrices. The dimensions can be easily justified. On one hand, in the current Advection-Diffusion Problem, five coordinates are defined $(x, y, u_1, u_2, \kappa)$. On the other hand, two different phenomena

are found (advection and diffusion) for two different coordinates ($x$ and $y$); so that, the problem is divided in x-advection, y-advection, x-diffusion and y-diffusion.

- **BB**: here, the source term is defined. As the source term is redefined in a separated form, the extra-coordinates terms (in our case $u_1, u_2, \kappa$) must not modify the solution. Hence, the cell structure understands five vectors, three of which are filled with ones. Finally, as in our problem, no source term is defined in the space domain, the source terms which apply for $x$ and $y$ are represented by an array filled with zeros.

- **N_NT**: for a more general problem, the mass matrices need to be considered within the solver. In our particular case, this cell structure has 5 eye matrices which dimensions depends on the number of each parameter discretization.

- **Dirichlet**: this structure defines the index where the Dirichlet boundary conditions are found. Normally, it has as cells as the number of coordinates of the problem. If no Dirichlet BC are found in a coordinate, the array is left as empty.

- **GG**: here, 5 vectors are defined to force Dirichlet BC. As in the cell **BB**, arrays corresponding to the direction where no Dirichlet BC are defined are filled with ones. Nevertheless, if some coordinate needs Dirichlet BC, they must be specified here. For instance, in our problem, the second cell must contain the non-homogeneous boundary temperature.

Then, the whole problem is solved and stored in a cell structure named **FF**. The application of Dirichlet BC in the final results can be seen when representing each column of the different cell's matrix (see Annex: Figures A.8, A.9, A.10, A.11 and A.12). If there is any Dirichlet BC in any parameter, the first column of the ***FF*** correspondent cell is the correspondent ***GG*** array.

### Online stage

With the offline stage computed, it is only needed to launch as many solutions as desired following the algorithm 8.

### Numerical aspects

On one hand, POD runs the algorithm a predefined number of times. Thus, no stopping criterion is needed. The predefined number of POD computation is 1000.

**Algorithm 8** Online PGD.

**Input:** The solved offline problem saved within **FF**.

1. Definition of the value of the parameters $u_1, u_2, \kappa$. This values can be found in the indexes $iU, iV, iK$ of the corresponding **FF**'s cell.
2. Computation the weight of the solution: $\alpha = \mathbf{FF}\{3\}(iU) \cdot \mathbf{FF}\{4\}(iV) \cdot \mathbf{FF}\{5\}(iK)$.
3. Computation of the solution for the whole domain: $\mathbf{T} = \mathbf{FF}\{2\}' \cdot \alpha \cdot \mathbf{FF}\{1\}$.

**Return:** The temperature field **T**.

On the other hand, PGD process applies stopping criterion in two different loops. The loop ranked lower is the one for the fixed-point algorithm (inside the enrichment process). Here, a tolerance of $10^{-8}$ is sought for the following Stopping Criterion[20] (**FF** answers for the solution):

$$\frac{\left|\left|\mathbf{FF}_n^p - \mathbf{FF}_n^{p-1}\right|\right|}{\left|\left|\mathbf{FF}_n^{p-1}\right|\right|} \tag{A.1}$$

Then, for the Enrichment loop, a simplified Stopping Criterion[20] has been applied with a sought tolerance of $10^{-5}$. The simplified criterion is defined as the norm of the current enrichment iteration $n$ divided by the norm of the first enrichment iteration:

$$\frac{||\mathbf{FF}_n||}{||\mathbf{FF}_1||} \tag{A.2}$$

In both criterion, a limit number of iterations is defined in order to force the exit.

## Results

### Steady Advection-Diffusion Problem

One and only problem has been treated with the ROM. This problem has been extracted from Amsallem's notes [6]. The problem is a Steady Advection-Diffusion problem (A.3) applied in a closed domain with non-homogeneous Dirichlet boundary conditions (see Figure A.2).

$$\mathbf{u} \cdot \nabla T - \kappa \cdot \Delta T = 0 \tag{A.3}$$

Where the range of the parameters are defined below:

$$\begin{cases} \mathbf{x} = (x, y) \in [0, 1] \times 0, 1] \\ \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}; u_1, u_2 \in [0, 0.5] \\ \kappa \in [0, 0.025] \end{cases} \tag{A.4}$$

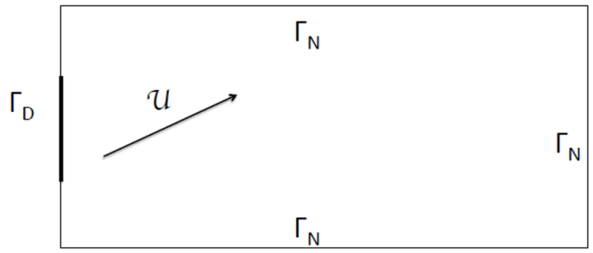

Figure A.2: Steady Advection-Diffusion Problem.[6]

The boundary conditions of the problem are expressed in (A.5).

$$\begin{cases} T(\{x,y\} \in \Gamma_D) = T_D(y,\bar{y}) \\ \nabla T(\{x,y\} \in \Gamma_N) \cdot \mathbf{n}(\mathbf{x}) = 0 \end{cases} \tag{A.5}$$

With the non-homogeneous boundary temperature defined as below:

$$0.9T_D(y,\bar{y}) = \begin{cases} 300, & \text{if } y \in [0,13] \\ 300 + 325 \cdot (sin(3\pi \cdot |y-\bar{y}| + 1), & \text{if } y \in [13,23] \\ 300, & \text{if } y \in [23,1] \end{cases} \tag{A.6}$$

Where $\bar{y} = 0.4$.

### Reference Solution

Due to the need of validating the solution while solving both POD and PGD strategies, one case has been used. To define it, the values selected are $u_1 = 0.11\ ms$, $u_2 = 0\ ms$ and $\kappa = 0.025\ ms^2$.
As it has been explained in algorithm 5, firstly a huge matrix were all the cases are solved is computed. For the current solution, it has been solved for a mesh of $61 \times 61$ elements, the dimensions of the matrix used to compute the solution are $3481 \times 3481$.
In Figure A.3, the case solution is presented. In order to understand the physics of the problem, in the Figure A.6 in the Appendix, the reader can find several cases where different parameters have been changed: $u_1$, $\kappa$ and $\bar{y}$. This solution has been extracted from [6].

### POD Solution

As in the reference solution, all the cases have been computed. Nevertheless, as it is the goal of the POD, the matrix used to compute the solution has been reduced. In the POD case, the matrix has a dimension of $3481 \times 10$. The size reduction is consistently translated to a reduction in the computation time and data storing space, but it also implies that an error due to the loss of information.

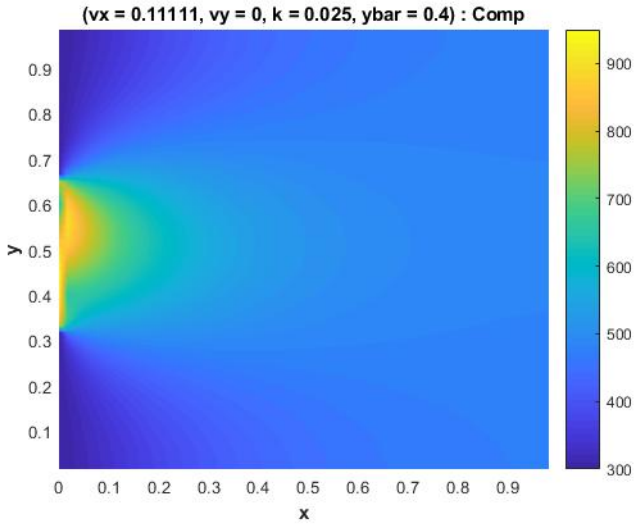


Figure A.3: Computational reference solution for $u_1 = 0.11\ ms$, $u_2 = 0\ ms$ and $\kappa = 0.025\ ms^2$ with a mesh of $61 \times 61$ elements.

In pursuance of considering an average behaviour, the POD process has been performed 20 times and an average of the maximal error, an average of the average error and an average of the computation time have been determined. The results have revealed the following results: $E_{max}^{avg} = 7.13\%$ and $E_{avg}^{avg} = 0.94\%$. Besides, the time used to compute the results has been reduced by a 85%. Thus, the results justify the use of a POD strategy if a maximum error of 7.13% is affordable.
In Figure A.4, the case solution is presented.

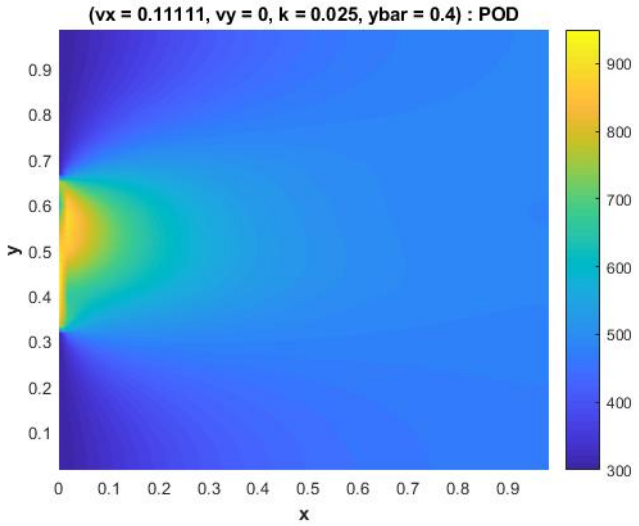


Figure A.4: POD solution for $u_1 = 0.11\ ms$, $u_2 = 0\ ms$ and $\kappa = 0.025\ ms^2$ with a mesh of $61 \times 61$ elements.

To enlarge the study, the reader can find represented in Figure A.7 of the Appendix the different modes of the POD based RB $\boldsymbol{V}$.

### PGD Solution

The PGD methodology must be analyzed twice (offline and online stage). Firstly, the phase online is evaluated.

Once the problem is solved offline for a wide range of values for all the extra-coordinates it is really fast to get results. Particularly, each solution can be solved with less than half of a second, which represents a reduction of nearly 95%.

Secondly, the offline phase must be also analyzed. First of all the computation time, which has increased the time from nearly 10 seconds to 45 minutes[1]. Furthermore, the dimensions of the data for this case varies depending on the number of iterations needed to reach the tolerance of the enrichment process. However, it is easily seen that the amount of data is bigger than in the other cases since it must be added to the dimensions of all the solution matrices. There are 5 matrices of dimension $[N_{elements}^{parameter} \times n_{iterations}]$.

Nevertheless, the results of the offline phase cannot be accepted because of an issue in the Stopping Criterion of the enrichment process: the tolerance is not reached. Although the evaluation of the whole methodology could be conducted, the validation of the results could not. Hence, while the fixed-point algorithm reaches the desired tolerance, the enrichment process does never reach the specified tolerance before 1500 iterations -the trigger value for stopping the enrichment iterations if the tolerance is not achieved. Since lowering the exigency in the tolerance has not reached the required level of accuracy of the solution, a proper Stopping Criterion for the critical loop should be defined in order to achieve success.
Just for exemplifying this issue, in Figure A.5 the PGD case solution is presented.

### Some solutions for the Problem

In order to understand the physics of the problem, using the reference code from Amsallem, several cases have been tested and are presented in Figure A.6.

[1]Please note that these values are referred to the computer used. This is not an absolute result.

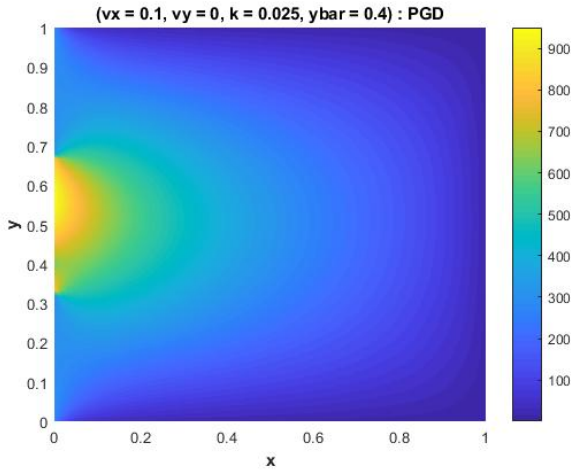


Figure A.5: PGD solution for $u_1 = 0.10\ ms$, $u_2 = 0\ ms$ and $\kappa = 0.025\ ms^2$ with a mesh of 61 elements for $x$ and $y$; 21 elements for $u_1$ and $u_2$; and 31 elements for $\kappa$.

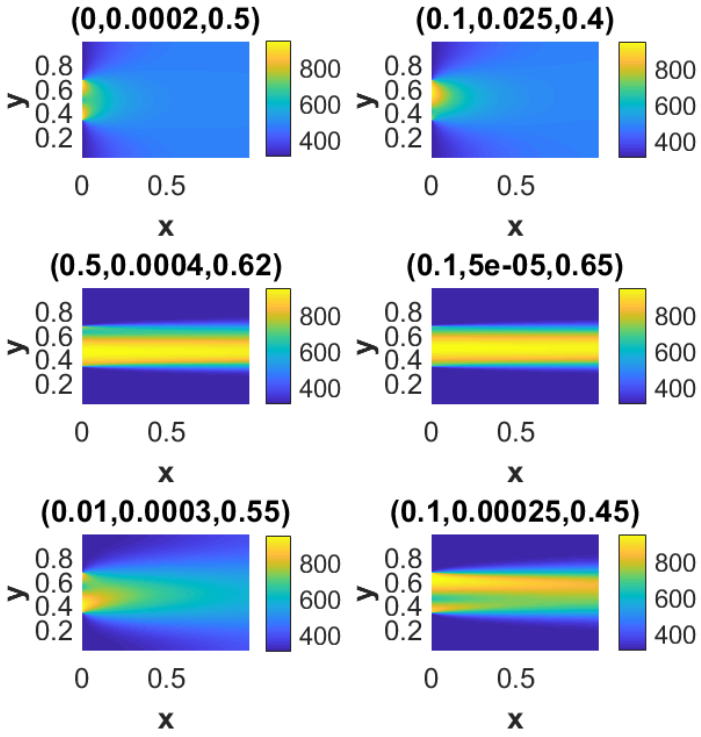


Figure A.6: Several solutions of the Advection-Diffusion problem. In the title: $(u_1, \kappa, \bar{y})$; $u_2 = 0$ for all the test cases.

# Images of the Solution

## POD Modes

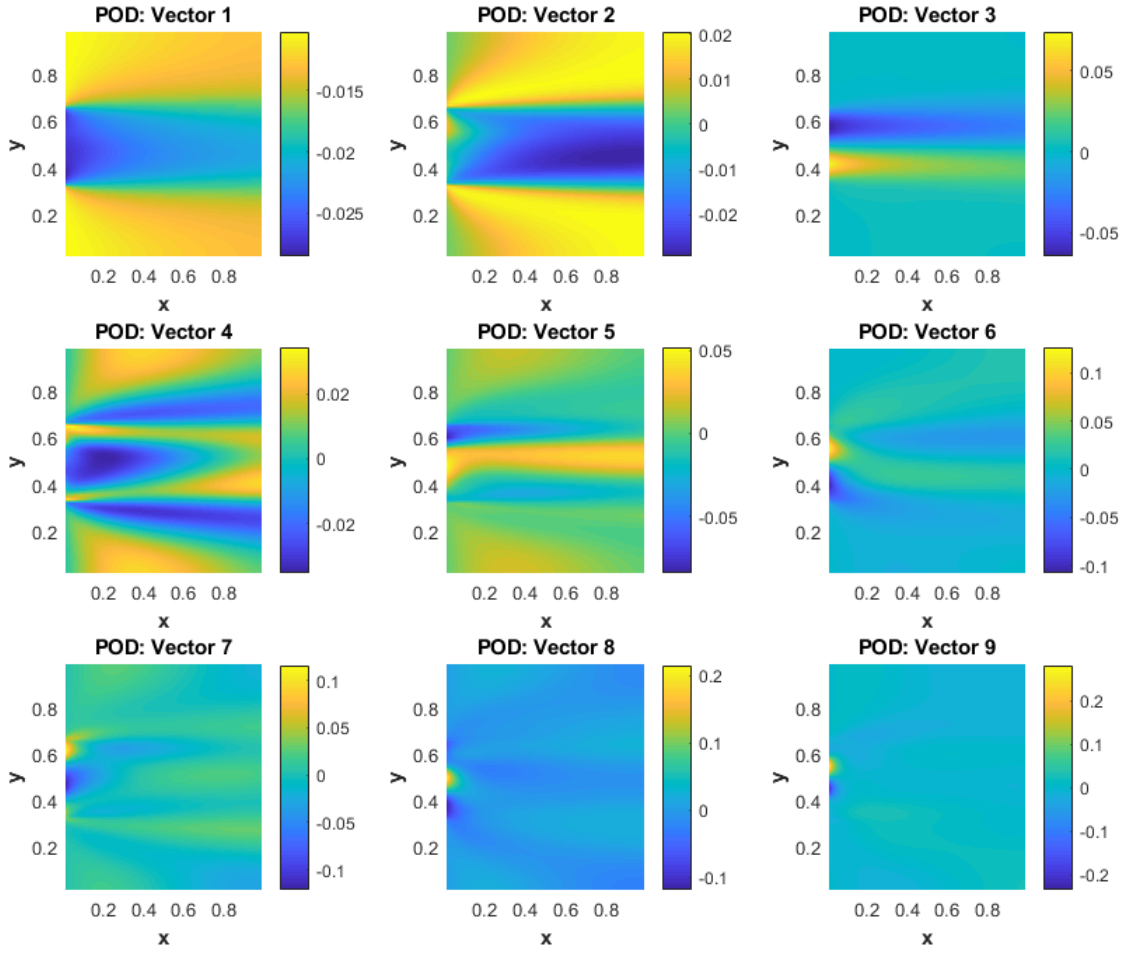


Figure A.7: The nine resulting modes of the Greedy+POD algorithm.

### PGD solution normalized functions

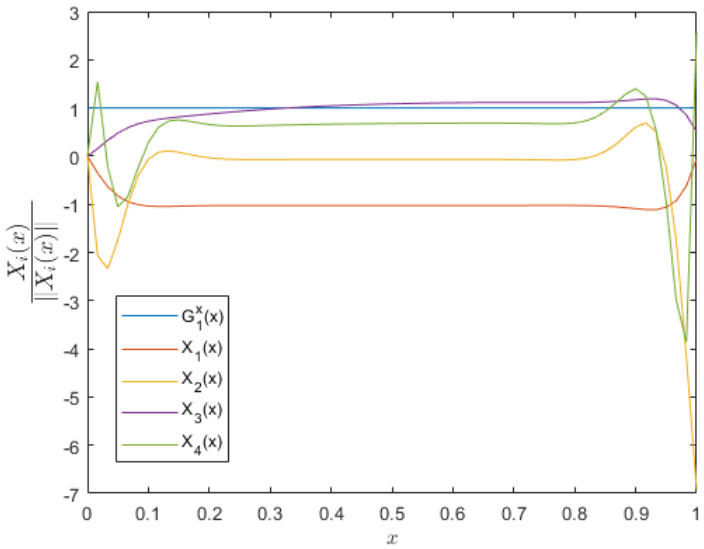


Figure A.8: Normalized functions $X_i(x)$ for $i = 1, \cdots, 5$ produced by the PGD solution of the Advection-Diffusion problem.

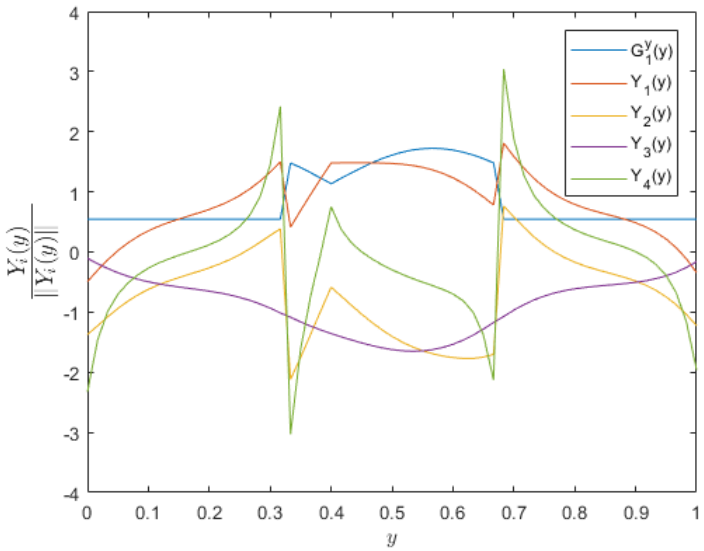


Figure A.9: Normalized functions $Y_i(y)$ for $i = 1, \cdots, 5$ produced by the PGD solution of the Advection-Diffusion problem.

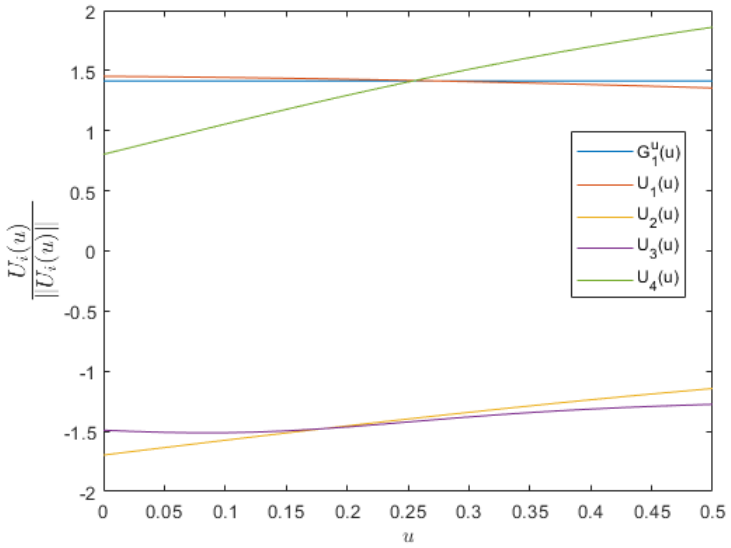


Figure A.10: Normalized functions $U_i(u_1)$ for $i = 1, \cdots, 5$ produced by the PGD solution of the Advection-Diffusion problem.

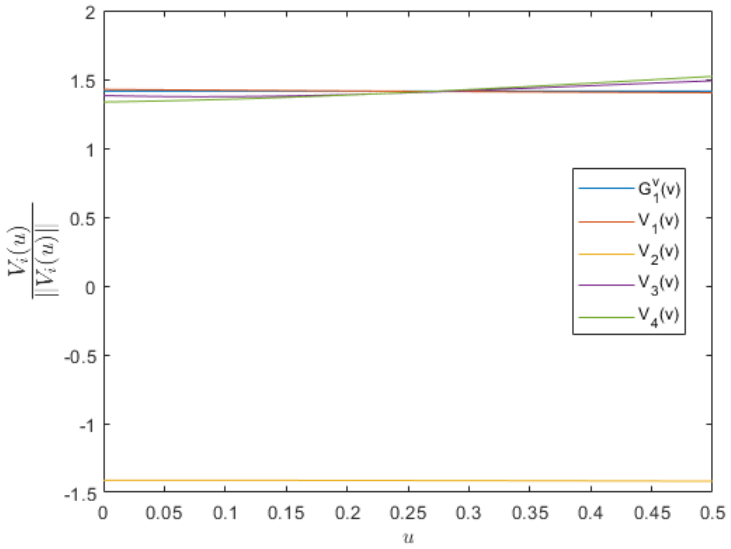


Figure A.11: Normalized functions $V_i(u_2)$ for $i = 1, \cdots, 5$ produced by the PGD solution of the Advection-Diffusion problem.

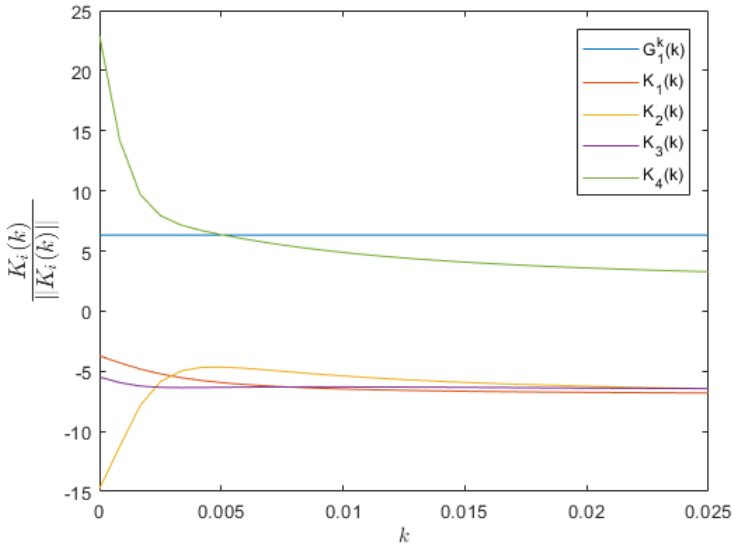


Figure A.12: Normalized functions $K_i(\kappa)$ for $i = 1, \cdots, 5$ produced by the PGD solution of the Advection-Diffusion problem.