Chapter 1

Bivariate decompositions

Contents

In order to give the full picture of data reduction technique, it is crucial to begin with bivariate problems. Indeed almost all multivariate techniques result from these 2D versions. Bivariate decomposition techniques were mainly theoretical at the time they were proposed in the first half of the 20$^{\text{th}}$ century [Pea01, Hot33], manual computations limited the size of the studied problems. But the numerical analyis and properties have been studied in details with emerging spectral theory [EY36, Kos43]. Actual implementations were carried on later in the second half of the 20$^{\text{th}}$ for fluid dynamics systems [Lum67, BHL93, Sir87]. 2D data reduction techniques are well understood and have been applied to the widest variety of problems in the last 20 years either to compress data or build reduced order model [Fah01, NAM+03, AF08].

In order to offer a broader view of the possible uses of bivariate decomposition, Fig. 1.1 proposes a schematic view of bivariate problem Reduced order modeling methods. The decomposition techniques presented in this section form the base material of many ROMs. They are organized as follow. The dashed black line shows the dichotomy between the continuous approach[1] and the discrete one. Then the orange dashed line separates

[1] These approaches are conceptually continuous but their implementations requires discrete description of the continuous space including grids, discrete operators,...

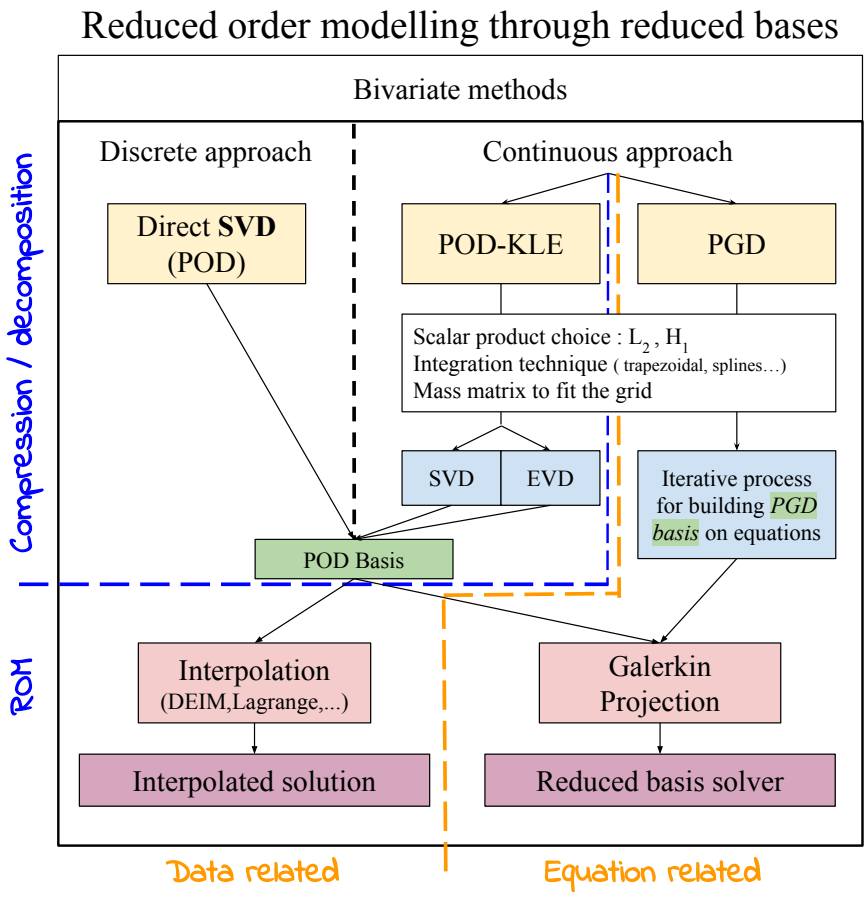


Figure 1.1: Synthetic view of the procedures described in chapter 1 for model order reduction of bivariate PDEs. The vertical arrows describe the work flow of these techniques and the dotted lines highlight the conceptual differences between them.

the techniques that only apply to data –namely SVD and POD– from the PGD which is usually used on the equation itself but can be degraded into a data decomposition method. Finally, the blue dashed line emphases the data compression nature of the POD and SVD while noting the possibility to obtain a ROM through the obtained basis as shown in the lower part of the diagram.

This chapter is organized as follow. First, section 1.1 describes the singular value decomposition (SVD) as it is both the simplest and the most used methods in this manuscript. Then, in section 1.2, proper orthogonal decomposition (POD) is built and analyzed, its discrete version is presented as well. Section 1.3 provides a 2D construction of the proper generalized decomposition (PGD), and how degrading it turns the method into a decomposition that is in fact an iterative algorithm to compute POD. Finally, section 1.4 exhibits a series of numerical tests and application of these methods as well as insight on their relative advantages. All along these sections, a particular focus is given on the numerics associated with the studied methods.

1.1 Singular Value Decomposition

The Singular Value Decomposition (referred as **SVD**) is a generalization of the eigenvalues decomposition for rectangular matrices. Among its many applications it can be seen as a discrete version of the POD.

Theorem 1.1.1 (Singular Value Decomposition [PS14]). *For any matrix $A \in \mathbb{R}^{m\times n}$, there are orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ so that*

$$A = U\Sigma V^\intercal$$

where Σ is a diagonal matrix of size $n \times m$ with diagonal elements $\sigma_{ii} \geq 0$.

Hereafter, it is assumed that the singular values are ordered decreasingly i.e. if $i < j$ then $\sigma_{ii} \geq \sigma_{jj}$. The SVD is not unique since the signs of U and V may vary.

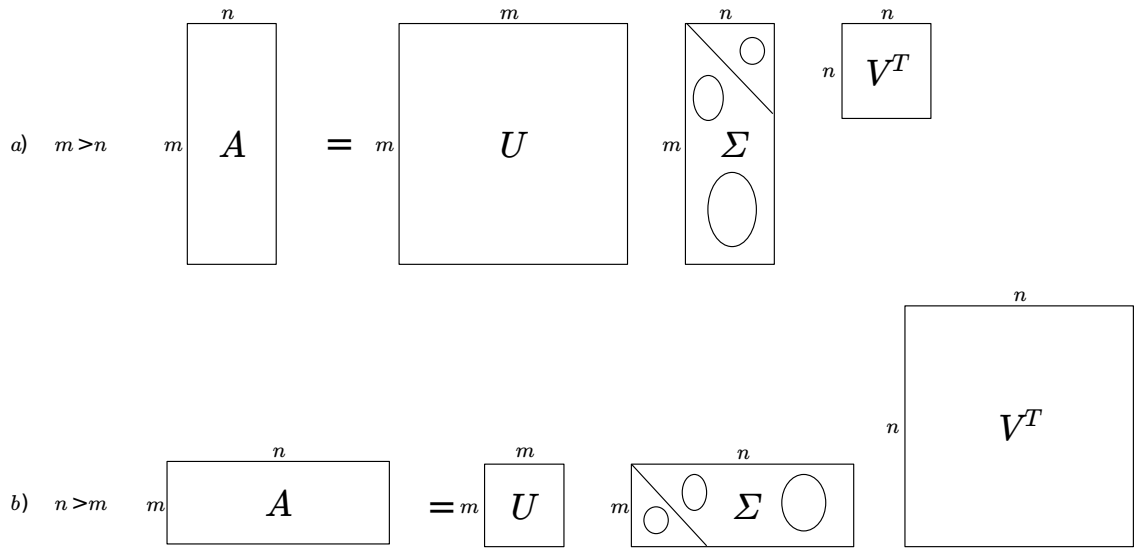


Figure 1.2: Singular Value Decomposition two configurations

One should note from figure 1.2 that a part of U in case a) and V in case b) only serves a dimension match without entering calculation of A, then the SVD reads for case a)

$$A = [U_1, U_2][\Sigma_1, 0]^\intercal V^\intercal = U_1\Sigma_1 V^\intercal$$

Let $\text{rank}(A) = r$ then for $k > r$, $\sigma_k = 0$. The SVD of A can be written as sum

$$A = \sum_{i=1}^{r} \sigma_i U_i V_i^\intercal$$

where σ_i are the diagonal entries of Σ and U_i and V_i refer to the columns of U and V respectively. Then $\|A\|_2 = \sqrt{\sum_{i=1}^{r} \sigma_i^2}$ leads to the optimality theorem proven by Eckart and Young in 1936 [EY36].

Theorem 1.1.2 (Eckart-Young). *Let $k < r$ and $A_k = \sum_{i=1}^{k} \sigma_i U_i V_i^\intercal$ where the singular values are ordered decreasingly then*

$$\min_{\text{rank}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1} \tag{1.1.1}$$

Remark (Link with the eigenvalue decomposition). Singular and eigenvalues are closely linked. Let $A \in \mathbb{R}^{m\times n}$ with $m > n$. $A^\intercal A = V\Sigma^\intercal\Sigma V^\intercal \longrightarrow AA^\intercal = V\Sigma_1^2 V^\intercal$. Then the eigenvalue problem of $A^\intercal A$ is equivalent to the right singular value problem of A with $\lambda_i = \sigma_i^2$ and the eigenvectors of $A^\intercal A$ are collinear to A's right singular vectors $\boldsymbol{v}_i$. The same applies to $\boldsymbol{u}_i$ and the eigenvectors of $A^\intercal A$.

Remark (Solving least square minimization problem with the SVD). The classical least square minimization problem i.e. find x^n of minimum Euclidean norm that reaches the minimum of $\|b - Ax\|_2$ for $A \in \mathbb{R}^{m\times n}$, is solved by the SVD and the Monroe-Penrose pseudo inverse of A (see [PS14]).

The main information contained in the Eckart-Young theorem is that the truncated SVD (see Fig. 1.3) i.e. only keeping the k dominant modes gives an optimal approximation of rank-k of the matrix A which rank is $r \geq k$. It means that the k first singular vectors form the optimal projection basis of size k that reads as follow,

$$A \approx A_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i \tag{1.1.2}$$

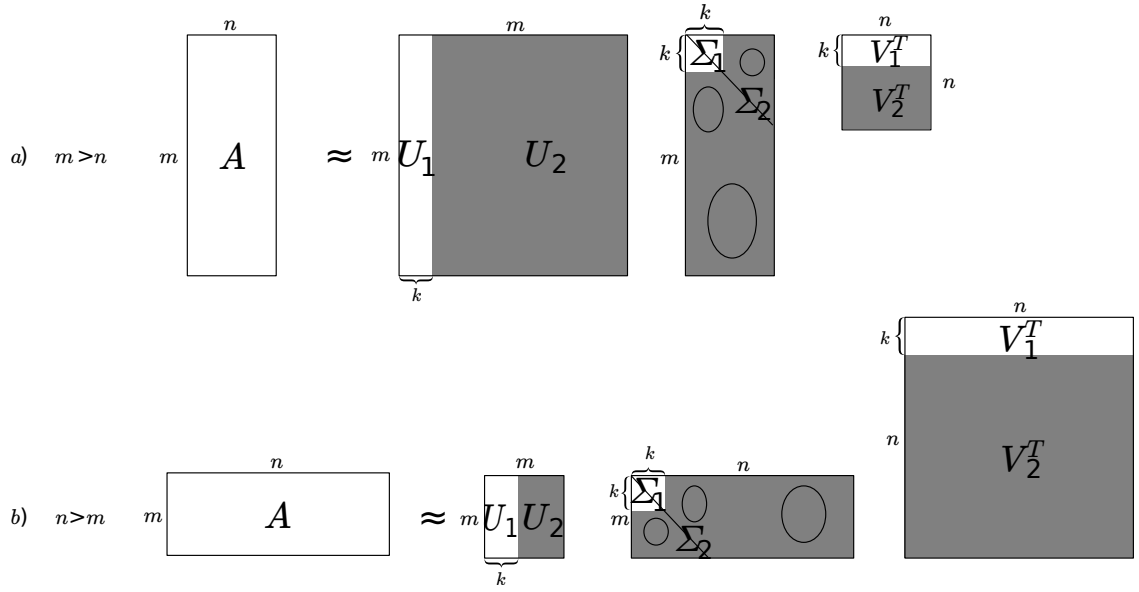


Figure 1.3: Rank k truncated-SVD for both configurations, the shadowed part is dropped upon truncation. $k \leq n,\ k \leq m$.

Numerics As for the eigenvalue decomposition, there are many algorithm to compute the SVD, among them, the QR algorithm is particularly well suited to slim matrices. In subsequent numerical experiment the LAPACK library is used either as direct SVD solver `dgesdd` or through eigenvalue decomposition `dsyev` if the matrix is slim (this strategy is also well suited for discrete POD as discussed in the next section). `dgesdd` relies on a divide and conquer approach which is one of the most efficient way to handle matrices of large size.

Other algorithm provide direct truncated SVD mainly based on iterative algorithm such as *Arnoldi procedure* based library `ARPACK`. However it is mostly suited for sparse matrices and the PGD fixed point procedure (presented in section 1.3) provides us with a way to obtain of truncated basis. It should be noted that iterative algorithm are very efficient at finding eigen/singular values at both end of the spectrum but face accuracy issues in other regions, especially for ill-conditioned matrices. This results in non orthonormal bases which may impair decomposition or ROM accuracy.

1.2 Proper Orthogonal Decomposition

The POD was discovered many times in many different fields, however it is often attributed to Kosambi [Kos43] who introduced it in 1943. Also, the POD comes under many names depending on the field in which it is used or devised. For instance, it is rigorously equivalent to the Karhunen-Loève expansion [Loè77] or Principal Component Analysis (PCA) usually attributed to [Pea01]. It is an elegant way to approximate a high dimensional[2] system into a low dimensional one. To do so, a linear procedure is devised to compute a basis of orthogonal proper modes that represent the energy repartition of the system. They are obtained by solving Fredholm's equation for data (usually) obtained through numerical simulations. Additionally the POD offers an optimal representation of the energy in term of L^2 norm.

It has been applied to extract dominant patterns and properties in wide variety of fields such as signal, data compression, neural activity, mechanics or fluid dynamics to name only a few. An enlightening description of the use of POD is given by Bergmann [Ber04]": *The POD defines uniquely and without ambiguity coherent structures*[3]*, as the realization of largest projection on the mean realization contained in the database*".

Problem formulation (scalar case). Find the best approximation, in the sense of a given inner product $(\cdot,\cdot)$ and average operator $\langle\cdot,\cdot\rangle$, of $f : \mathcal{D} = \Omega_x \times \Omega_t \longrightarrow \mathbb{R}$ as a finite sum in the form

$$\tilde{f}_r(\boldsymbol{x}, t) = \sum_{k=1}^{r} a_k(t)\phi_k(\boldsymbol{x}) \tag{1.2.1}$$

where $(\phi_k)_k$ are orthogonal for the chosen inner product. a_k is given by $a_k(t) = (f(\cdot, t), \phi_k(\cdot))$ then a_k only depends on ϕ_k.

Discrete POD problem is often found in the literature as follow. Let $\{f_1, ..., f_{n_t}\}$ the snapshots of f i.e. the representation of f at discrete time $\{t_j\}_{j=1}^{n_t}$. It is assumed that $\mathcal{F} = \text{span}\{f_1, f_2, ..., f_{n_t}\}$.

POD generates an orthonormal basis of dimension $r \leq n_t$, which minimizes the error from approximating the snapshots space $\mathcal{F}$. The POD basis verifies the optimum of the following:

$$\min_{\{\phi\}_{k=1}^{r}} \sum_{j=1}^{n_t} \|f_j - \tilde{f}_{r,j}\|^2, \text{ s.t. } (\phi_k, \phi_j) = \delta_{kj} \tag{1.2.2}$$

where $\tilde{f}_{r,j} = \sum_{k=1}^{r} (f_j, \phi_k)\, \phi_k$ and δ_{kj} is the Kronecker symbol. One may observe that $\sum_{k=1}^{r} \cdot$ is the first order approximation of the time mean operator $\langle\cdot\rangle$. This problem can be solved with discrete Eigen Value Decomposition (EVD). Although it is the most common formulation of discrete POD in mechanics literature, it can be misleading regarding the construction and properties of the POD. This is why a much more detailed presentation is given in this thesis.

1.2.1 Building the POD

This subsection aims at providing a rigorous, however mechanics oriented presentation of the POD. The present approach is based on Bergmann thesis manuscript [Ber04] and

[2]Here, high dimensionality is to be understood as rich phenomenon that require many degrees of freedom to be described properly as opposed to simpler system which are described by few degrees of freedom e.g. simple pendulum.

[3]The notion of coherent structures, introduced by Lumley (1967) [Lum67,Lum81] is central in the use of POD for mechanics.

lecture notes at Von Karman Institute together with Cordier [CB03a, CB03b] as well as some of the vast corpus available including [Ale15, Cha00, Fah01]. Since POD is the cornerstone of several multivariate data reduction techniques, it is crucial to provide the mathematics underlying this method. Without loss of generality, the usual framework for POD where the two variables are space (possibly a position vector) and times. It makes mental representation easier for the reader and most of the POD jargon was introduced with time-space POD.

Let $\boldsymbol{X} = (\boldsymbol{x}, t) \in \mathcal{D} = \Omega_x \times \Omega_t$ and $\boldsymbol{u} : \mathcal{D} \longrightarrow \mathbb{R}^d$ a vector valued function. Additionally we assume[4] that a scalar product $(\cdot, \cdot)$ is defined on $\mathcal{D}$ and $||\cdot||$ its associated norm while an average operator $\langle\cdot\rangle$ is defined on $\mathcal{D}$[5]. We also need the following u to be of finite norm. The dominant modes of a set of realization $\{\boldsymbol{u}(\boldsymbol{X})\}$ are sought, i.e. the function $\boldsymbol{\phi}$ whith the largest projection on realizations $\{\boldsymbol{u}(\boldsymbol{X})\}$ in the least square sense. In other words, we seek $\boldsymbol{\phi}$ that maximizes $|(\boldsymbol{u}, \boldsymbol{\phi})|$ where $\boldsymbol{\phi}$ is normalized. Then the maximum of this expression is sought

$$\frac{\langle|(\boldsymbol{u}, \boldsymbol{\phi})|^2\rangle}{\|\boldsymbol{\phi}\|^2} \tag{1.2.3}$$

This leads to the following constrained maximization problem

$$\max_{\boldsymbol{\psi} \in L^2(\mathcal{D})} \frac{\langle|(\boldsymbol{u}, \boldsymbol{\psi})|^2\rangle}{\|\boldsymbol{\psi}\|^2} = \frac{\langle|(\boldsymbol{u}, \boldsymbol{\phi})|^2\rangle}{\|\boldsymbol{\phi}\|^2} \tag{1.2.4}$$

with

$$(\boldsymbol{\phi}, \boldsymbol{\phi}) = 1$$

In order to rewrite problem (1.2.4), a linear operator $\mathcal{R} : L^2(\mathcal{D}) \longrightarrow L^2(\mathcal{D})$ is introduced, it is defined as

$$\mathcal{R}\boldsymbol{\phi}(\boldsymbol{X}) = \int_{\mathcal{D}} R(\boldsymbol{X}, \boldsymbol{X'})\boldsymbol{\phi}(\boldsymbol{X'})d\boldsymbol{X'} \tag{1.2.5}$$

where $R(\boldsymbol{X}, \boldsymbol{X'}) = \langle \boldsymbol{u}(\boldsymbol{X}) \otimes \boldsymbol{u}(\boldsymbol{X'})\rangle$ is the tensor of spatio-temporal correlations. Now suppose that $\langle\cdot\rangle$ and $\int$ can be permuted then the following holds

$$\begin{aligned} (\mathcal{R}\boldsymbol{\phi}, \boldsymbol{\phi}) &= \langle|(\boldsymbol{u}, \boldsymbol{\phi})|^2\rangle \quad \geq 0 \\ (\mathcal{R}\boldsymbol{\phi}, \boldsymbol{\psi}) &= (\boldsymbol{\phi}, \mathcal{R}\boldsymbol{\psi}) \qquad \forall(\boldsymbol{\phi}, \boldsymbol{\psi}) \in [L^2(\mathcal{D})]^2 \end{aligned}$$

Since $\mathcal{R}$ is a positive self-adjoint operator, the spectral theory applies and the solution of problem (1.2.4) is given by the largest eigen value of this new problem

$$\mathcal{R}\boldsymbol{\phi} = \lambda\boldsymbol{\phi} \tag{1.2.6}$$

It can be written as a Fredholm integral equation:

$$\sum_{j=1}^{d} \int_{\mathcal{D}} R_{ij}(\boldsymbol{X}, \boldsymbol{X'})\phi^j(\boldsymbol{X'})d\boldsymbol{X'} = \lambda\phi^i(\boldsymbol{X}) \quad \forall i \tag{1.2.7}$$

[4] We will see in 1.2.2 that x and t play symmetric roles as long as the operators are well defined.

[5] The natural choice for fluid dynamics applications $L^2(\Omega_x)$ scalar product and a time average. The choice of the average operator $\langle\cdot\rangle$ kind (temporal, spatial,...) determines which kind of POD is used.

Some fundamental properties of the POD.

1. For $\mathcal{D}$ bounded, Hilbert-Schmidt theory applies and ensures the existence of countably infinitely many solutions to equation (1.2.7)

$$\sum_{j=1}^{d} \int_{\mathcal{D}} R_{ij}(\boldsymbol{X}, \boldsymbol{X'}) \phi_r^j(\boldsymbol{X'}) d\boldsymbol{X'} = \lambda_r \phi_r^i(\boldsymbol{X}) \tag{1.2.8}$$

where $\lambda_r, \boldsymbol{\phi}_r$ are respectively the POD eigenvalues and eigen functions of order $r = 1, 2, ..., +\infty$. Each new eigen function is defined as the solution of problem (1.2.6) adding a new constraint: orthogonality with the already known eigen functions.

$$\sum_{i=1}^{d} \int_{\mathcal{D}} \phi_r^i(\boldsymbol{X}) \phi_p^i(\boldsymbol{X}) d\boldsymbol{X} = \delta_{rp} \tag{1.2.9}$$

2. $\mathcal{R}$ is positive self-adjoint then $\lambda_i \geq 0$. Additionally, they are taken decreasing and they form a converging series i.e.

$$\sum_{r=1}^{\infty} \lambda_i \leq +\infty$$

.

3. The POD eigen functions form a complete basis, any realization $u(\boldsymbol{X})$ can be represented in that basis.

$$u^i(\boldsymbol{X}) = \sum_{r=1}^{\infty} a_r \phi_r^i(\boldsymbol{X}) \tag{1.2.10}$$

4. a_r is obtained by projecting $\boldsymbol{u}$ on $\boldsymbol{\phi}_r$

$$a_r = (\boldsymbol{u}, \boldsymbol{\phi}_r) = \sum_{i=1}^{d} \int_{\mathcal{D}} u_i(\boldsymbol{X}) \phi_r^i(\boldsymbol{X}) d\boldsymbol{X} \tag{1.2.11}$$

5. **Mercer's Theorem.** The spatio-temporal correlation matrix at two points R_{ij} is kernel based on $\mathcal{R}$ then Mercer's theorem provides a series representation,

$$R_{ij}(\boldsymbol{X}, \boldsymbol{X'}) = \sum_{r=1}^{\infty} \lambda_r \phi_r^i(\boldsymbol{X}) \phi_r^j(\boldsymbol{X'}) \tag{1.2.12}$$

6. Thanks to the previous property, it can be shown [CB03a] that the coefficients a_r are uncorrelated and their quadratic average is equal to the POD eigenvalues

$$\langle a_r, a_p \rangle = \delta_{rp} \lambda_r \tag{1.2.13}$$

7. Using Mercer's theorem and the orthogonality of POD eigen functions, the following expression emerges

$$\sum_{i=1}^{d} \int_{\mathcal{D}} R_{ij}(\boldsymbol{X}, \boldsymbol{X}) d\boldsymbol{X} = \sum_{r=1}^{\infty} \lambda_r = E \tag{1.2.14}$$

Where E coincides with kinetics energy if $\boldsymbol{u}$ is the velocity field of a fluid for example. Then λ_r indicates the weight of each modes in terms of energy.

Remark. These properties ensure the uniqueness of the proper orthogonal decomposition (given that $||\boldsymbol{\Phi}|| = 1$).

Optimality of the POD basis. Let $\boldsymbol{u} : \mathcal{D} \longrightarrow \mathcal{E} \subset \mathbb{R}^d$ with $\boldsymbol{u} \in L^2(\mathcal{D})$ and $\bar{\boldsymbol{u}}$ an approximation of $\boldsymbol{u}$. On a any basis $(\boldsymbol{\psi}_r(\boldsymbol{X}))_{r=1}^{\infty}$ one can write

$$\bar{u}_i(\boldsymbol{X}) = \sum_{r=1}^{\infty} b_r \psi_r^i(\boldsymbol{X}) \tag{1.2.15}$$

Let $\{\boldsymbol{\phi}(\boldsymbol{X})\}_{r=1}^{\infty}$ a set of orthogonal POD eigen functions and $\{\lambda_r\}_{r=1}^{\infty}$ their associated eigenvalues. Then, $\boldsymbol{u}^{POD}$ the POD approximation of u is considered

$$u_i^{POD}(\boldsymbol{X}) = \sum_{r=1}^{\infty} a_r \phi_r^i(\boldsymbol{X}) \tag{1.2.16}$$

Properties 6 and 7 state that if $(\boldsymbol{\psi}_r(\boldsymbol{X}))_{r=1}^{\infty}$ are non dimensional, $\langle b_r, b_r \rangle$ represents the energy of mode n. Cordier and Bergmann [CB03b] proved the optimality of the POD basis through the following lemma.

Lemma 1.2.1. *Optimality of POD basis For any rank $R \in \mathbb{N}^*$ the following inequality holds*

$$\sum_{r=1}^{R} \langle a_r, a_r \rangle = \sum_{r=1}^{R} \lambda_r \geq \sum_{r=1}^{R} \langle b_r, b_r \rangle \tag{1.2.17}$$

In other words, among all linear decomposition, POD is the most efficient, i.e. for a given number of POD modes R, the projection on the subset produced by the first R POD eigen-functions is the one that contains on average the most (kinetic) energy possible.

Operation Count In order to evaluate the number of operations required to compute the POD decomposition of simulation data we consider the simple case where both variables have the same number of samples N, i.e. $\Omega = \Omega_x \times \Omega_t$ is discretized in an $N \times N$ matrix. For $f : \Omega \to \mathbb{R}$, first, the correlation matrix is computed

$$\boldsymbol{R}(x, x') = \int_{\Omega_t} f(x, t) f(x', t) dt \tag{1.2.18}$$

Obviously, the cost depends on the integration technique. For this evaluation we choose a second order method: trapezoidal integration rule which cost is in $\mathcal{O}(N)$. Then this operation has to be performed for each discrete combination of x and x' which results in N^2 evaluations. The global cost to evaluate $\boldsymbol{R}(x, x')$ on the discrete grid is $\mathcal{O}(N^3)$ double precision operations. Then the first $R \ll N$ eigen values of problem (1.2.5) are sought. This problem can be solved using a Lanczos algorithm which requires very few iteration to compute the first eigenvalues, it requires $\mathcal{O}(MN^2)$. Then an estimate of the operation count to compute the M mode POD of f with a Lanczos algorithm is

$$\mathcal{O}(N^3 + MN^2) = \mathcal{O}(N^3) \tag{1.2.19}$$

As shown in the next sections choosing to apply the POD to the dimension with the lowest degrees of freedom (DoF) number will lead to much lower number of operation. Especially if one dimension DoF number is much lower than the other.

A POD algorithm. One of the many possible implementations of the POD is proposed in this section. Although it might not be the most computationally efficient version, it preserves all the functional approach framework. Indeed the user is free to implement any integration method so that the projector also apply to L^2, not to any matrix space. This

statement is also true the linear operator $\boldsymbol{\mathcal{R}}\phi^m$ which eigenvalue problem is solved by a iterative orthonormal power method.

Algorithm 1: POD (*Standard, Deflation Power Method*)

input : f, target error ϵ
output: $\tilde{f}_r = \sum_{k=1}^{r} \sigma_k X_k Y_k$

m=0
$\boldsymbol{R}(x,x') = \int_{\Omega_t} f(x,t)f(x',t)dt$;
while $\frac{\sigma_m}{\|f\|_{L^2}} \geq \epsilon$ **do**
  $k = k+1$
  $(\lambda_k, \phi_k) =$ `Orthonormal_Power_method` $[(\boldsymbol{\mathcal{R}} - \tilde{\boldsymbol{f}}_{\boldsymbol{k-1}})\phi_k = \lambda_k\phi_k]$
  $\sigma_k = \sqrt{\lambda_k}$
  $a_k = \int_{\Omega_x} f(x,t)\phi_k(x)dx/\ \sigma_k$
  $\tilde{f}^k = \tilde{f}_{k-1} + \sigma_k\phi_k a_k$
return f_k

1.2.2 Discussion on the POD variations

Choosing the right realizations Since all information used in POD comes from the chosen realizations[6] $\boldsymbol{X}$, one might wonder on which criteria to choose them. It is a complex question and in some cases it has been shown that major flow properties are not preserved (e.g. Noack [NAM+03]).

Choosing the inner product One very interesting property of POD is that the inner product can be chosen depending on the studied problem. For instance, if a problem requires to preserve certain properties such as incompressibility, one can choose a inner product that is more suitable for this task. In fluid dynamics applications, mainly the following two possibilities emerge.

Inner product on L^2. $L^2(\Omega)$ is the Hilbert space of square integrable functions is usually well suited for fluid dynamics applications. The inner product on $L^2(\Omega)$ for two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined by

$$(\boldsymbol{u}, \boldsymbol{v}) = \int_{\Omega} \left(\sum_i u_i v_i \right) d\boldsymbol{x} \tag{1.2.20}$$

where $\|\boldsymbol{u}\|^2 = (\boldsymbol{u}, \boldsymbol{u})$ is the associated norm.

A fluid kinetic energy is proportional to $\|\boldsymbol{u}\|^2$ in fluid dynamics applications. Then, it seems reasonable to use this inner product for general fluid dynamics problems.

Inner product on H^1. Iollo et al. [ILD00] were among the first to advocate the use of Sobolev spaces for improved quality in POD based reduced models. Indeed, L^2 norm was found to be unstable. H^1 norm has been continuously used since then, for example for parabolized Navier-Stokes equation [DNS+12]. $H^1(\Omega)$ is the Sobolev space containing $L^2(\Omega)$ functions which first derivative are also part of $L^2(\Omega)$. The inner product on H^1 for two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$(\boldsymbol{u}, \boldsymbol{v})_\epsilon = \int_{\Omega} \boldsymbol{u} \cdot \boldsymbol{v}\ d\boldsymbol{x} + \epsilon \int_{\Omega} (\boldsymbol{\nabla u} \cdot \boldsymbol{\nabla v})\ d\boldsymbol{x} \tag{1.2.21}$$

where ϵ is a numerical parameter accounting for different measures.

[6] Also known as *snapshots* in the fluid dynamics community.

The POD was described in a general framework in the previous section. However in practical applications, the choice of the actual first and second variable has a great influence on the numerical computing of POD bases. Choosing time or space as the first variable will affect both speed and accuracy of POD algorithms. The standard POD was introduced by Lumley [Lum81] while Sirovich [Sir87] proposed the snapshots version.

1.2.2.1 Standard POD

Lumley's approach [Lum81] for POD relies on choosing $\langle\cdot\rangle$ as a temporal average of the realizations i.e.

$$\langle \cdot \rangle = \frac{1}{T}\int_{\mathcal{T}} \cdot \, dt \tag{1.2.22}$$

where $\mathcal{T} = [0, T]$. It is assumed that $\mathcal{T}$ is a period in which all realizations are known and is long enough to represent the flow. The space variable, $\boldsymbol{x}$, lives in $\Omega \in \mathbb{R}^d$. For a 3D fluid dynamics simulation, $d = 3$, we can focus on the velocity fields $\boldsymbol{u} : \Omega \times \mathcal{T} \longrightarrow \mathbb{R}^3$ with the usual $L^2(\Omega)$ scalar product:

$$(u, v)_{L^2(\Omega)} = \sum_{i=1}^{d}\sum_{j=1}^{d}\int_{\Omega} u_i v_j d\boldsymbol{x}, \quad \forall u, v \in L^2(\Omega) \tag{1.2.23}$$

Then Fredholm's equation (1.2.7) now reads

$$\sum_{j=1}^{d}\int_{\Omega} R_{ij}(\boldsymbol{x}, \boldsymbol{x'})\phi^j(\boldsymbol{x'})d\boldsymbol{x'} = \lambda\phi^i(\boldsymbol{x}), \quad \forall 1 \leq j \leq d \tag{1.2.24}$$

Where R_{ij} is the spatial correlation tensor now reads

$$R_{ij}(\boldsymbol{x}, \boldsymbol{x'}) = \frac{1}{T}\int_{\mathcal{T}} u_i(\boldsymbol{x}, t)u_j(\boldsymbol{x'}, t)\, dt \tag{1.2.25}$$

It should be noted that in this case, $\boldsymbol{\phi}$ is purely spatial.

Size of the eigenvalue problem, discrete case. Let n_x be the number of spatial grid points. Usually, $n_x \in [10^5, 10^9]$ for DNS simulations. d is the number of elements of $\boldsymbol{u}$, e.g. $d = 3$ for 3D cases. Then this approach becomes intractable as soon as 3D problems are studied, even using adapted algorithm/softwares.

It should be used only when data is spatially sparse, for example within particle tracking problems. An efficient way to overcome this difficulty for DNS is to use the snapshots method.

1.2.2.2 Snapshots POD

The snapshots method was originally introduced by Sirovich [Sir87] in 1987. It is the counterpart of the standard POD where the role of $\boldsymbol{x}$ and t are inverted. It is well suited for data where there is a large number of spatial grid points while relatively low number of time frames e.g. DNS output data. The average operator is a spatial average that reads

$$\langle \cdot \rangle = \int_{\Omega} \cdot \, d\boldsymbol{x} \tag{1.2.26}$$

i.e. for two fields, the $L^2()$ scalar product defined in 1.2.23. Then, the POD scalar product is the time integral on $\mathcal{T}$. The eigen problem now reads

$$\int_{\mathcal{T}} C(t, t')a(t')\, dt' = \lambda a(t) \tag{1.2.27}$$

where C is the temporal correlation matrix that does not account for cross correlation. In order to preserve consistency with the previous definition a $1/T$ is imposed before the integral in C definition.

$$C(t,t') = \int_\Omega \sum_{i,j=1}^{d} u_i(\boldsymbol{x},t)u_j(\boldsymbol{x},t')\, d\boldsymbol{x} \tag{1.2.28}$$

The eigen functions are functions of time only. Then the eigen problem of this snapshot POD is of reasonable size, $r = n_t$ the number of time frames/snapshots. This is particularly well suited if $n_t \ll n_x$ which is the usual case for DNS output data. One can recover the spatial POD modes $\boldsymbol{\phi}_n(\boldsymbol{x})$ by projecting the snapshots on the function with or without normalization as per the user preference i.e.

$$\boldsymbol{\phi}_k(\boldsymbol{x}) = \int_{\Omega_t} \boldsymbol{u}(\boldsymbol{x},t)a_k(t)dt, \quad \forall 1 \leq k \leq r \tag{1.2.29}$$

Both these methods share a set of properties.

- Any realization $u_i(\boldsymbol{x},t)$ can be represented exactly on the full POD basis which is orthonormal.
$$u_i(\boldsymbol{x},t) = \sum_{n=1}^{N_{POD}} \sigma_n a_n(t)\phi_n^i(\boldsymbol{x})$$
where $N_{POD} \leq \infty$. Potentially, an infinite number of modes may be required.

- Temporal modes (a_n) form an orthogonal family (that can be normalized) while spatial modes $(\boldsymbol{\Phi}_n)$ form an orthonormal family.

- Any property that can be written as a linear combination of the realizations is directly passed to the spatial eigen-functions. Incompressibility or Dirichlet boundary conditions are two examples of properties that are passed to the basis $\boldsymbol{\phi}_n$, $\forall n < N_{POD}$ if u has these properties.

1.2.3 SVD $\simeq$ POD

From the previous sections, it clearly appears that POD and SVD share many of their properties. One can adopt two different angles to explore the link between POD and SVD.

a. Use the the optimality of the SVD to solve the discrete POD minimization problem. This is a straightforward application of the fact that eigenvectors can be computed either from eigenvalue decomposition or SVD. This approach has been described in detail by Bergmann and Fahl's work [Ber04,Fah01]. Compared to the SVD technique described in section 1.1, it adds a mass matrix but does not require any linear algebra analysis to be linked with POD. Details are given in section 1.2.3.1.

b. The other way of looking at this link, was proposed among many others by Chaterjee [Cha00]. It is a simpler presentation of the POD, only valid in the discrete framework. It relies on the fact that the SVD solves optimally a matrix problem that may be seen as the discrete equivalent of the infinite dimensional problem (1.2.2) using the Euclidian norm for vectors. As will be shown in the general framework of tensor spaces in chapter 2, this approach is justified by the applicability of the same algorithm to tensor spaces of different nature, either continuous or discrete.

It shall be noted that these two interpretations leads to different algorithms which may not display the same properties of accuracy or efficiency especially when the basis is used for reduced order modeling as its orthogonality is a very important feature. The very illustration is the possibility to chose a problem adapted inner product in the POD algorithm while SVD is blind to data and will be performed in the same fashion for any problem, sometimes without preserving physical properties.

1.2.3.1 Numerical implementation of snapshots L^2 POD

In order to implement efficiently the snapshot L^2 POD one needs to carefully compute all the scalar products according to a given integration scheme and accuracy. From that point there are two possibilities either one chose to program weighted sum and solve Fredholm's equation (1.2.7) using a suitable algorithm e.g. power iteration/deflation method. Or they chose to use linear algebra tools (SVD, EVD solver) and the integration becomes a matrix vector product. Additionally a change of basis on the snapshot data is required for the discrete operator to preserve L^2 properties. The second option has been selected in this work after observing that it is much more computationally efficient in the fortran implementation. Moreover, it allows one to use optimally programmed solver such as `LAPACK` or `ARPACK` libraries as well as `BLAS` operations. A brief overview of this method is given next.

A reasonably general setup is chosen for this typical implementation of discrete POD. Let $\mathcal{D} = \Omega \times T \subset \mathbb{R}^d \times \mathbb{R}$ a spatio-temporal domain that is discretized in $n_x \times n_t$ elements and scalar function $f \in L^2(\mathcal{D})$. Let $\boldsymbol{F} \in \mathbb{R}^{n_x \times n_t}$ the matrix representation of f which columns are $\boldsymbol{f}_i = (f(x_1, t_i), f(x_2, t_i), \cdots, f(x_{n_x}, t_i))^\intercal$. The goal is to find the discrete representation $\{\Phi, A\}$ of the POD basis $\{\phi_i, a_i\}$.

- **Inner product and associated norm:**
  Let u and v two scalar functions of $L^2(\mathcal{X})$, $\mathcal{X}$ is either Ω or T which discretization are $\boldsymbol{u}$ and $\boldsymbol{v} \in \mathbb{R}^n$. Then $(u, v)_{L^2(\mathcal{X})}$ the inner product of these functions in $L^2(\mathcal{X})$ is discretized as $(\boldsymbol{u}, \boldsymbol{v})_{\boldsymbol{M}}$ where $\boldsymbol{M}$ is the interpolation matrix. For instance, $\boldsymbol{M}$ is diagonal for a trapezoidal rule with the weights associated to this integration scheme. The discrete integration operator reads

$$(u, v)_{L^2(\mathcal{X})} \xrightarrow{disc} (\boldsymbol{u}, \boldsymbol{v})_{\boldsymbol{M}} = \boldsymbol{u}^\intercal \boldsymbol{M} \boldsymbol{v}$$

  The norm associated to these inner product behave identically $||u||_{L^2} \xrightarrow{disc} \|\boldsymbol{u}\|_{\boldsymbol{M}}$. This procedure is applied on both time and space variables. $M^{1/2}$ is the Choleski decomposition left matrix of $\boldsymbol{M}$ i.e. $\boldsymbol{M} = \boldsymbol{M}^{1/2}\boldsymbol{M}^{1/2\intercal}$. As stated earlier for usual integration techniques (trapozoidal, Simpson's, etc.), the matrix is diagonal and $m_{ij}^{1/2} = \sqrt{m_{ij}}$. Let $\boldsymbol{M}_t$ be the time integration matrix and $\boldsymbol{M}_x$ the space integration matrix. For instance a trapezoidal rule on a uniform grid yields the following time integration matrix,

$$(M_t)_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ \delta t & \text{if } 1 < i < N_t \\ \delta t/2 & else. \end{cases}$$

  We assume that the space integration is also diagonal with $(M_x)_{ii} = w_i$ where w_i is the weight corresponding to the integration formula e.g. P0, P1, etc. for finite elements, trapezoidal for finite differences, etc.

- **Discrete autocorrelation function for snapshot POD:**
  The autocorrelation function

$$C(t,t') = \int_\Omega f(\boldsymbol{x},t) f(\boldsymbol{x},t')\, d\boldsymbol{x}$$

  is discretized into a matrix $\boldsymbol{C} \in \mathbb{R}^{n_t \times n_t}$ whose elements are defined as

$$C(t_i,t_j) \approx C_{ij} = \sqrt{(M_t)_{ii}(M_t)_{jj}}\, \boldsymbol{f}_i^\intercal \boldsymbol{M}_x \boldsymbol{f}_j \tag{1.2.30}$$

  One can write $\boldsymbol{C}$ as a matrix product that reads $\boldsymbol{C} = \boldsymbol{M}^{1/2}\boldsymbol{M}^{1/2\intercal}\boldsymbol{F}^\intercal \boldsymbol{M}_x \boldsymbol{F}$, this operation can be seen as a change of basis for the autocorrelation matrix. Then one can apply the SVD (thin SVD or any EVD algorithm) on $\boldsymbol{C}$,

$$\boldsymbol{C} = \tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\,\tilde{\boldsymbol{V}}^\intercal \tag{1.2.31}$$

  Here, it is chosen to keep the full discrete basis. In practice, a truncated SVD will be used as discussed in sections 1.1 and 1.2.

  In order to recover the actual $\{\phi_i, a_i\}$ POD basis one need to apply the change of basis back to the discrete representation of the basis function and project f on the temporal basis,

$$\{a_i\}_i \approx \boldsymbol{A} = \boldsymbol{M}_t^{-1/2\intercal}\tilde{\boldsymbol{U}} \tag{1.2.32}$$

$$\{\phi_i\}_i \approx \boldsymbol{\Phi} = (\boldsymbol{F}^\intercal, \boldsymbol{A})_{\boldsymbol{M}_t} = \boldsymbol{F}\boldsymbol{M}_t\boldsymbol{A} \tag{1.2.33}$$

where $\boldsymbol{\Phi}_i$ is to be normalized for actual basis.

Proof. Proving that these matrix-vector products are indeed equivalent to the discrete L^2 POD is fairly straightforward discretization of operators. One can wonder why the POD is often presented as in eq. (1.2.2), this is because, in standard POD, only the spatial basis is kept to build a ROM, thus this mixed $L^2\Omega/l^1(T)$ is sufficient. □

Remark (Versatility of such implementation). Implementing this discrete POD enables very easy change in the integration method. The special case of $M = I$ is equivalent to the usual algebraic solver (with a slight overhead). This means that numerically, any method relying on SVD is interchangeable with the equivalent discrete POD. Thus the user can very easily switch from l^2-norm to L^2-norm and virtually use any integration scheme for POD. In order to prevent misunderstanding, l^2 POD algorithm will be referred as `SVD_by_EVD`.

1.3 Proper Generalized Decomposition

In order to provide a general overview of the bivariate methods, we now focus on a method that is more recent than POD and SVD. The Proper Generalized Decomposition (PGD) has been developed by a relatively small cluster of researchers during the 2000's including Chinesta, Cueto, Ladevèze, Ammar among others. The method is a variation of the popular LATIN method [CL93] developed in the 1980's by Ladevèze et al. It was first developed in the context of mechanics [CLA+09, CALK11, ACDH10] and later extended to general systems of PDEs. Nouy and Falco have extended it to a more general framework [Nou10, FN12] and provide additional numerical analysis.

This presentation of the PGD is very restrictive as compared to the full capabilities of this method. On the one hand, only the bivariate case is shown here, a multivariate

version is given in section 3.1. On the other hand, the PGD is a general algorithm that can be applied to partial differential equation (PDE), the separated approximation problem can be written as PDE $u = f$ where u is sought as a separate sum, this problem can be refered as *a posteriori* PGD. This section is restricted to describing how the PGD is build in the case of bivariate data post-processing.

1.3.1 Constructing a bivariate *a posteriori* PGD

Let $f : \Omega = \Omega_x \times \Omega_y \to \mathbb{R}$ a square integrable function, i.e. $f \in L^2(\Omega)$. As in the POD, the goal of the PGD is to provide a separated approximation of f that reads

$$f(x, y) \approx u_r(x, y) = \sum_{i=1}^{r} X_i(x) Y_i(y) \tag{1.3.1}$$

where $X_i \in L^2(\Omega_x)$, $Y_i \in L^2(\Omega_y)$, $\forall i \leq r$ that form an orthogonal basis of rank r of $L^2(\Omega_x)$ and $L^2(\Omega_y)$. The sequence (u_r) converges toward f, i.e. $u_r \xrightarrow[r\to\infty]{} u = f$

1.3.1.1 An Enrichment Process

In order to determine each element of the sequence an enrichment process has been proposed by Chinesta et al. [CKL13]. This algorithm enriches the decomposition basis recursively, each time adding a new pair of basis vectors $\{X_i, Y_i\}$.

X_r and Y_r are computed by a fixed point algorithm alternating directions. The weak formulation of the *a posteriori* PGD problem reads

$$\forall u^\star \in H^1(\Omega), \quad \int_\Omega u^\star (u - f) = 0 \tag{1.3.2}$$

At the begining of each step it is assumed that $u_{r-1}(x, y) = \sum_{i=1}^{r-1} X_i(x) Y_i(y)$ is known thus u_r is sought under the form

$$u_r(x, y) = u_{r-1}(x, y) + X_r(x) Y_r(y) \tag{1.3.3}$$

The process of adding terms to the sum, i.e. computing the sequence $(u_p)_{p=1}^r$ is called the *enrichment process*. This process ends when a stopping criterion is fulfilled. Since in the general case, one does not know the exact solution, it is chosen to stop the process when the weight of the last term compared to the rest of the series becomes negligible. This reads

$$\mathcal{E}(r) = \frac{||X_r Y_r||_{L^2(\Omega)}}{||X_1 Y_1||_{L^2(\Omega)}} = \frac{||Y_r||_{L^2(\Omega)}}{||Y_1||_{L^2(\Omega)}} \leq \varepsilon_{enrichment} \tag{1.3.4}$$

Indeed the terms are of decreasing norm, then there is no need to compare the whole series, the first term is sufficient. In addition to that, we define $\{X_i\}$ such that $\forall i < N, \quad ||X_i||_{L^2(\Omega)} = 1$ all the information about the norm is transfered to $\{Y_i\}$.

Remark (Choice of u_0). It is not trivial to chose u_0 as it influences the convergence speed of the fixed point algorithm and may even prevent it from converging. However in most cases it does not have much influence as the first mode contains a lot of energy that leads to quick convergence of the fixed point algorithm.

1.3.1.2 Fixed point algorithm

In this section, an iterative algorithm called Fixed Point Algorithm (FPA) is described, it enables the computing of a new term (X_r, Y_r) to the basis. This is the version described by Chinesta [CKL13]. In practice, either it converges in a few iterations or it does not converge at all. The key feature of this algorithm is its alternated direction nature, i.e. each direction is computed one at a time.

It is assumed that $\tilde{X}^k$ and $\tilde{Y}^k$ are known after step k of the FPA . Thus $\tilde{u}(x,y) = u_{r-1}(x,y) + \tilde{X}^k(x)\tilde{Y}^k(y)$. Moreover, the computing of $\tilde{X}^{k+1}$ relies on $\tilde{Y}^k$ while $\tilde{X}^{k+1}$is used to compute $\tilde{Y}^{k+1}$. $u^\star$ is set to

$$u^\star(x,y) = \begin{cases} X^\star(x)\tilde{Y}^k(y) & \text{searching } X^{k+1} \\ \tilde{X}^{k+1}(x)Y^\star(y) & \text{searching } Y^{k+1} \end{cases} \tag{1.3.5}$$

For simplicity reasons, the subsequent development will only address the computations needed to evaluate $\tilde{X}^{k+1}$ since the same process is at work for $\tilde{Y}^{k+1}$. Given the previous equations, the following weak formulation holds

$$\int_\Omega \left[X^\star(x)\tilde{Y}^k(y) \left(u_{r-1}(x,y) + \tilde{X}^{k+1}(x)\tilde{Y}^k(y) - f(x,y) \right) \right] dxdy = 0 \tag{1.3.6}$$

This equation can be written as follow

$$\alpha^x \int_{\Omega_x} X^\star(x)\tilde{X}^{k+1}(x)dx = -\int_{\Omega_x} X^\star(x) \sum_{j=1}^{r-1} \left(\beta_j^x X_j(x) \right) dx + \int_{\Omega_x} X^\star(x)\gamma^x(x)dx \tag{1.3.7}$$

where

$$\alpha^x = \int_{\Omega_y} (\tilde{Y}_i^k)^2 \tag{1.3.8}$$

$$\beta_j^x = \int_{\Omega_y} \tilde{Y}^k Y_j \quad \forall j < p \tag{1.3.9}$$

$$\gamma^x(x) = \int_{\Omega_y} \tilde{Y}^k f \tag{1.3.10}$$

Finally the strong formulation stands

$$\begin{cases} \tilde{X}^{k+1}(x) = \frac{-\sum_{j=1}^{r-1}\left(\beta_j^x X_j(x)\right) + \gamma^x(x)}{\alpha} & ,\forall x \in \Omega_x \\ \tilde{Y}^{k+1}(y) = \frac{-\sum_{j=1}^{r-1}\left(\beta_j^y Y_j(y)\right) + \gamma^y(y)}{\alpha} & ,\forall y \in \Omega_y \end{cases} \tag{1.3.11}$$

$\tilde{X}^{k+1}$ is normalized i.e. $||\tilde{X}^{k+1}||_{L^2(\Omega_x)} = 1$, so that all the information relative to the norm is transfered to $\tilde{Y}^{k+1}$. This algorithm is performed alternatively along x and y direction, every time $\tilde{Y}^{k+1}$ is computed the subsequent stopping criterion is checked.

$$\mathcal{E}_{fixed\,point}(k) = \frac{||\tilde{Y}^{k+1} - \tilde{Y}^k||_{L^2(\Omega_y)}}{||\tilde{Y}^k||_{L^2(\Omega_y)}} < \varepsilon_{fixed\,point} \tag{1.3.12}$$

Algorithm. A synthetic view of the algorithm is given to ease the implementation of the PGD.

Algorithm 2: PGD (*a posteriori*)

input : f
output: $u_r = \sum_{i=1}^{r} X_i Y_i$

$u_0 = 0$, $n = 0$, $(X_i, Y_i)_{i=1}^{r} = \{\}$;
while $\mathcal{E}(r) \geq \mathcal{E}_{enrich}$ **do**
  $r = r + 1$
  $(X_r, Y_r) =$ `fixed_point` $((X_i, Y_i)_{i=1}^{r-1}, f, n)$
  $u_r = u_r + X_r Y_r$
  $\mathcal{E}(r) = \frac{||Y_r||_{L^2(\Omega_y)}}{||Y_1||_{L^2(\Omega_y)}}$
return $u_r = \sum_{i=1}^{r} X_i Y_i$

Algorithm 3: `Fixed_point`

input : $(X_i, Y_i)_{i=1}^{r-1}$, f, n
output: $X_r Y_r$

$u_{r-1} = \sum_{i=1}^{r-1} X_i Y_i$, $k = 0$, $(\tilde{X}^k, \tilde{Y}^k) = (0, 0)$;
while $\varepsilon \geq \mathcal{E}_{fixed\_point}$ **do**
  $k = k + 1$
  compute α^x, $\beta^x(j)\ \forall j < n$, γ^x
  $\tilde{X}^{k+1} = \frac{-\sum_{j=1}^{r-1}\left(\beta_j^x X^j(x)\right) + \gamma^x(x)}{\alpha}$, $\forall x \in \Omega_x$
  $\tilde{X}^{k+1} = \tilde{X}^{k+1} / \|\tilde{X}^{k+1}\|_{L^2(\Omega_x)}$
  compute α^y, $\beta^y(j)\ \forall j < n$, γ^y
  $\tilde{Y}^{k+1}(Y) = \frac{-\sum_{j=1}^{r-1}\left(\beta_j^y Y^j(y)\right) + \gamma^y(y)}{\alpha}$, $\forall y \in \Omega_y$
  $\varepsilon = ||\tilde{Y}^{k+1}||_{L^2(\Omega_y)} / ||\tilde{Y}^{k}||_{L^2(\Omega_y)}$
return $(X_r = \tilde{X}^k, Y_r = \tilde{Y}^k)$

1.3.2 Equivalence of PGD with POD/SVD for bivariate decomposition

In section 1.2, algorithm 1 was given to compute the POD of a function, it relies on a deflated power iteration method. In order to show the connection between PGD and POD/SVD, this method is detailed here.

POD reminder and notations. Assume that $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^s$ are two bounded domains, d and s are integers ≥ 1. Let f be a given function in the Lebesgue space $L^2(X \times Y)$. For practical reasons, the integral operator $\mathcal{B}$ with kernel f is introduced

$$\varphi \mapsto \mathcal{B}\,\varphi, \qquad (\mathcal{B}\,\varphi)(y) = \int_X f(x,y)\varphi(x)\,dx. \tag{1.3.13}$$

The operator $\mathcal{B}$ maps $L^2(X)$ into $L^2(Y)$, is bounded and has an adjoint operator $\mathcal{B}^*$ defined from $L^2(Y)$ into $L^2(X)$ as

$$v \mapsto \mathcal{B}^*\,v, \qquad (\mathcal{B}^*\,v)(x) = \int_Y f(x,y)v(y)dy. \tag{1.3.14}$$

The self-adjoint operator $\mathcal{R} = \mathcal{B}^*\mathcal{B}$ is also an integral operator whose kernel $R \in L^2(X \times X)$ is the autocorrelation function defined in section 1.2.1,

$$R(x,x') = \int_Y f(x,y)\,f(x',y)\,dy. \tag{1.3.15}$$

Thus we recover Fredholm's equation ((1.2.7)) with the eigenvalues λ_n, such as

$$\mathcal{R}\,\varphi_n = \lambda_n\,\varphi_n, \qquad \forall n \geq 0. \tag{1.3.16}$$

A straightforward effect of the diagonalization of the operator $\mathcal{R}$ is the following singular value decomposition of the operator $\mathcal{B}$.

Lemma 1.3.1. *There exists a system $(\varphi_n, v_n, \sigma_n)_{n\geq 0}$ such that $(\varphi_n)_{n\geq 0}$ is an orthonormal basis in $L^2(X)$, $(v_n)_{n\geq 0}$ an orthonormal system in $L^2(Y)$ and $(\sigma_n)_{n\geq 0}$ a sequence of nonnegative real numbers such that*

$$\mathcal{B}\,\varphi_n = \sigma_n\, v_n, \qquad \mathcal{B}^*\, v_n = \sigma_n\, \varphi_n. \tag{1.3.17}$$

The sequence $(\sigma_n)_{n\geq 1}$ is ordered decreasingly and decays toward zero.

As discussed previously, we have $\sigma_n = \sqrt{\lambda_n}$, $\forall n \geq 1$, additionally the singular vectors $(\varphi_n)_{n\geq 1}$ are the same as the eigenvectors of $\mathcal{R}$.

1.3.2.1 Power iteration method

The power iteration ranges among the simplest computational eigenvalue methods. We apply this iterative method to approximate the dominant eigenvalues of $\mathcal{R}$ or equivalently the dominant singular values of $\mathcal{B}$. The convergence results of the power iteration will addressed of briefly. A lot of work has been done in this issue. For instance, one can refer to [GL96] and references there in.

The basic form of the iterate power method applied to $\mathcal{R} = \mathcal{B}^*\mathcal{B}$ aims to construct the largest eigenvalue $\lambda_1 = (\sigma_1)^2$ and the related eigenvector φ_1. Thus, σ_1 is the largest singular value of B. The scaled version of it is recommended to avoid underflow/overflow. It can be presented as follows:

Algorithm 4: Power iteration (eigen value problem)

input : $\mathcal{R}$
output: Largest eigenvalue and vector, $\{\lambda_1, \varphi_1\}$
Choose $\varphi^{(0)} \in L^2(X)$ with $\|\varphi^{(0)}\|_{L^2(X)} = 1$
repeat
  $\chi^{(k)} = \mathcal{R}\varphi^{(k-1)}$
  $\varphi^{(k)} = \frac{\chi^{(k)}}{\|\chi^{(k)}\|_{L^2(X)}}$
until *convergence*;
return $\{\lambda_1, \varphi_1\} = \{\|\chi^{(k)}\|_{L^2(X)}, \varphi^{(k)}\}$

The limit of the sequence $(\varphi^{(k)})_{k\geq 0}$ is the eigen-function φ_1 and the sequence $(\|\chi^{(k)}\|_{L^2(X)})_{k\geq 0}$ converges toward the dominant eigenvalue λ_0. Also, for practical reasons, we set $\lambda^{(k)} = \|\chi^{(k)}\|_{L^2(X)}$.
Rewritten in terms of the singular value approximation the algorithm reads

Algorithm 5: Power iteration (singular value ploblem)

input : $\mathcal{B}, \mathcal{B}*$
output: Largest singular value and vectors, $\{\sigma_1, \varphi_1, v_1\}$
Choose $\varphi^{(0)} \in L^2(X)$ with $\|\varphi^{(0)}\|_{L^2(X)} = 1$
repeat
  $w^{(k)} = \mathcal{B}\varphi^{(k-1)}$
  $\chi^{(k)} = \mathcal{B}^* w^{(k)}$
  $\varphi^{(k)} = \frac{\chi^{(k)}}{\|\chi^{(k)}\|_{L^2(X)}}$
until *convergence*;
return $\{\sigma_1, \varphi_1, v_1\} = \{\|w^{(k)}\|_{L^2(X)}, \varphi^{(k)}, w^{(k)}/\|w^{(k)}\|_{L^2(Y)}\}$

If the convergence is ensured then the sequence $(w^{(k)}))_{n\geq 0}$ tends toward w_0. We introduce also the notation $\sigma^{(k)} = \|w^{(k)}\|_{L^2(Y)}$. Passing to the limit, it is easily checked out that $\lim_{k\to\infty} \sigma^{(k)} = \sigma_0$.

1.3.2.2 Connection between the methods

Let us try first to express in a variational form the collection of problems involved in the second version of the power iteration algorithm (5). Line 3 variational form reads

$$\int_Y w^{(k)}(y)w^*(y)\,dy = \int_Y (f(\cdot,y),\varphi^{(k-1)})_{L^2(X)}\,w^*(y)\,dy, \qquad \forall w^* \in L^2(Y).$$

On account of the normalization $\|\varphi^{(n-1)}\|_{L^2(X)} = 1$ we obtain that

$$\int_{X\times Y} \left(f - \varphi^{(k-1)} \otimes w^{(k)}\right)(\varphi^{(k-1)} \otimes w^*)\,dxdy = 0, \qquad \forall w^* \in L^2(Y).$$

We turn now to the evaluation of $\varphi^{(k)}$. A variational form of line 4 $\chi^{(k)} = \mathcal{B}^* w^{(k)}$ reads

$$\int_X \chi^{(k)}(x)\varphi^*(x)\,dx = \int_X (f(x,\cdot),w^{(k)})_{L^2(Y)}\,\varphi^*(x)\,dx, \qquad \forall \varphi^* \in L^2(X).$$

Since $\|w^{(k)}\|_{L^2(Y)} = \sigma^{(k)}$ we derive

$$\int_{X\times Y} \left(f - \frac{1}{(\sigma^{(k)})^2}\chi^{(r)} \otimes w^{(k)}\right)(\varphi^* \otimes w^{(k)})\,dxdy = 0, \qquad \forall \varphi^* \in L^2(X).$$

Remark. In the last equation, the function $\frac{\chi^{(k)}}{(\sigma^{(k)})^2}$ converges toward φ_0. As a result, the following limit holds

$$\lim_{k\to\infty} \frac{\lambda^{(k)}}{(\sigma^{(k)})^2} = 1.$$

Once the dominant singular value σ_1 with its corresponding modes (φ_1, w_1) are approximated, one has to evaluate the following ones. The *deflation-based power iteration process* succeeds in doing so. Assume the first modes $(\sigma_n, \varphi_n, w_n)$ with $n < N$ are known, then compute the next mode $(\sigma_N, \varphi_N, w_N)$. The deflation mechanism is described first for the computation of (σ_N, φ_N) as the eigen(vector, value) of the operator $\mathcal{R}$. Let

$$\tilde{\mathcal{R}} = \mathcal{R} - \mathcal{R}_{N-1} = \mathcal{R} - \sum_{1\le n<N} \lambda_n\varphi_n \otimes \varphi_n. \tag{1.3.18}$$

Then, the deflated iterate power method is given in algorithm 6

Algorithm 6: Deflated Power Iteration method (eigen value problem)

input : $\mathcal{R}$, required number of modes N
output: Eigenvalues and vectors, $\{\lambda_n, \varphi_n\}_{n>0}$

```
1 n=0
2 while n < N do
3     Choose φ^(0) ∈ L^2(X) with ||φ^(0)||_{L^2(X)} = 1
4     repeat
5         χ^(k) = R̃ φ^(k-1)
6         φ^(k) = χ^(k) / ||χ^(k)||_{L^2(X)}
7     until convergence;
8     {λ_n, φ_n} = {||χ^(k)||_{L^2(X)}, φ^(k)}
9 return {λ_n, φ_n}_{n≤N}
```

In order to operate the deflated algorithm on the operator $\mathcal{B}$, the following result on the kernels is necessary

Proposition 1.3.2. *This equality holds*

$$(R - R_{N-1})(x, x') = \int_Y (f - f_{N-1})(x, y)(f - f_{N-1})(x', y)\, dy, \qquad \forall (x, \xi) \in X \times X.$$

Proof. Computations starts as follows

$$\begin{aligned}\int_Y (f - f_{N-1})(x, y)(f - f_{N-1})(x', y)\, dy = &\int_Y f(x, y) f(x', y)\, dy \\ &- \sum_{0 \le n < N} \varphi_n(x') \int_Y f(x, y) w_n(y)\, dy - \sum_{0 \le n < N} \varphi_n(x) \int_Y w_n(y) f(x', y)\, dy \\ &+ \sum_{0 \le n < N} \sum_{0 \le k < N} \varphi_n(x) \varphi_k(x') \int_Y w_n(y) w_k(y)\, dy\end{aligned}$$

Various orthogonalities yield that

$$\begin{aligned}\int_Y (f - f_{N-1})(x, y)(f - f_{N-1})(x', y)\, dy = &\int_Y f(x, y) f(x', y)\, dy \\ &+ \sum_{0 \le n < N} \lambda_n \varphi_n(x) \varphi_n(x') - 2 \sum_{0 \le n < N} \lambda_n \varphi_n(x) \varphi_n(x') \\ = &\int_Y f(x, y) f(x', y)\, dy - \sum_{0 \le n < N} \lambda_n \varphi_n(x) \varphi_n(x') = (\mathcal{R} - \mathcal{R}_{N-1})(x, x').\end{aligned}$$

The proof is complete. □

Now, we introduce the deflated operators $\tilde{\mathcal{B}}$ defined by

$$\tilde{\mathcal{B}} = \tilde{\mathcal{B}} - \tilde{\mathcal{B}}_{M-1} = B - \sum_{1 \le k < M} \lambda_k \varphi_k \otimes w_k.$$

Corollary 1.3.2.1. $\tilde{\mathcal{R}} = \tilde{\mathcal{B}}^* \tilde{\mathcal{B}}$ *holds.*

Proof. After observing that that kernels of $\tilde{\mathcal{R}}$ and $\tilde{\mathcal{B}}$ are $(\mathcal{R} - \mathcal{R}_{N-1})(x, x')$ and $(f - f_{N-1})$ respectively, it is a direct consequence Proposition 1.3.2. □

The power iterations on $\mathcal{R}$ can be written as in algo. 4 for the approximation of λ_N. Then, each iteration can be split into two steps as in algo. 5 where the operator $\tilde{\mathcal{B}}$ plays the central role. Both versions are necessarily equivalent. Detailing in the same way as for the dominant singular value, one obtains

$$\int_Y w^{(r)}(y) w^*(y)\, dy = \int_Y \left((f - f_{N-1})(\cdot, y), \varphi^{(k-1)}\right)_{L^2(X)} w^*(y)\, dy, \qquad \forall w^* \in L^2(Y),$$

that also reads

$$\int_{X \times Y} \left(f - f_{N-1} - \varphi^{(k-1)} \otimes w^{(r)}\right) \left(\varphi^{(k-1)} \otimes w^*\right) dx dy = 0, \qquad \forall w^* \in L^2(Y). \tag{1.3.19}$$

One can recognize (1.3.6). The next line of the algorithm is the calculation of $\varphi^{(k)}$, it is conducted as follows

$$\int_X \chi^{(k)}(x) \varphi^*(x)\, dx = \int_X \left((f - f_{N-1})(x, \cdot), w^{(k)}\right)_{L^2(Y)} \varphi^*(x)\, dx, \qquad \forall \varphi^* \in L^2(X).$$

or equivalently

$$\int_{X\times Y}\left(f - f_{N-1} - \frac{1}{(\sigma^{(k)})^2}\chi^{(k)}\otimes w^{(k)}\right)(\varphi^*\otimes w^{(k)})\,dxdy = 0, \qquad \forall\varphi^*\in L^2(X). \quad (1.3.20)$$

The $\varphi^{(k)}$ is then defined by the normalization of $\chi^{(k)}$.

Finally equations 1.3.19 and 1.3.20 are identical to the weak *a posteriori* PGD algorithm presented in section 1.3.1. This construction of the deflated power iteration (DPI) is based on a POD and equivalent SVD formulation. Thus one can conclude that in the bivariate framework the *a posteriori* PGD is equivalent to a POD solved through DPI algorithm. It should also be noted that this section has also offered and equivalence between the usual POD and a singular alternative that one may identify to a continuous SVD (algorihtm 4 versus 5). Thus three algorithm families emerge to separate bivariate data : SVD , POD/EV and PGD/DPI solvers.

Additionally the interpretation of the PGD as a DPI algorithm open the field of improving PGD algorithms through the vast knowledge on deflation algorithm, first in the *a posteriori* framework, then in the more complex *a posteriori* framework. One among the attractive features common to numerous iterative solvers (Richardson, Gradient, Arnoldi, GMRES, etc.) of linear (and even nonlinear!) systems resides in their capacity to come up with the solution without accessing the related full matrix at once. Being able to operate that matrix on vectors is sufficient to start and end up those iterative solvers.

Such improvements of the PGD have been investigated in the literature, some interesting papers [TLN14, ACL15, CKL13] among many others.

1.4 Numerical experiments

In this section a few numerical tests are conducted on all three methods. Although it has been shown that they are mathematically equivalent, the difference between these algorithm will inevitably produce different behavior, especially for ill-conditioned problems/matrices. This first numerical section provides a suggested technique over the others depending on the problems studied. First some synthetic data is used i.e. analytical functions, then an image is compressed with various levels of accuracy. Finally, data from numerical simulations is separated.

Here we briefly recall the problem and the measure of success to solve it, i.e., the approximation error.

$$\boxed{\text{Find the best approximation of } f(x,y) \text{ such as } f_r(x,y) = \sum_{i=1}^{r} X_i(x)Y_i(y)}$$

with the error measured as $||f - f_r||_{L^2}$ or $||f - f_r||_{\mathrm{F}}$ depending on the nature of the method[7].

[7] Actually the choice of the norm has little influence on the numerical results. This is especially true for trapezoidal rule on a Cartesian grid. The main purpose of this distinction is consistency and to some extent application to ROM in part II.

1.4.1 Synthetic data

Let $\Omega = [0,1]\times[0,1]$ be the studied domain and four square integrable functions $f_1, f_2, f_3, f_4 : \Omega \to \mathbb{R}$ defined by

$$f_1(x,y) = xy \tag{1.4.1}$$
$$f_2(x,y) = \frac{1}{1+xy} \tag{1.4.2}$$
$$f_3(x,y) = \sin(\sqrt{x^2+y^2}) \tag{1.4.3}$$
$$f_4(x,y) = \sqrt{1-xy} \tag{1.4.4}$$
$$f_5(x,y) = \frac{1}{(1+xe^y)} \tag{1.4.5}$$

These functions range from already separated (f_1) to weakly separable, also known as *singular* functions in the literature. Thus these two expressions will be used indifferently in this manuscript. They are chosen to be easily extended to multiple variables.

The four methods PGD, POD ($L^2(\Omega)$) SVD and `SVD_by_EVD` are applied on these functions for a 32×32 regular Cartesian grid on a single processor *Fortran* or *python* implementation. Integrals are computed with trapezoidal rule.

Modes shape. It is not a trivial task to interpret the modes computed by these methods especially for (x, y) as shown in Fig. 1.4. First we focus on Fig. 1.4a and 1.4b which shows the first modes in for both variables yielded by all three methods. As expected, since l^2 scalar product is used, all methods yield the same normalized modes (Fig. 1.4a and 1.4b) for x and y since f_3 is symmetric. In Fig. 1.4c, one can see that the amplitudes of the PGD modes is plummetting with i, this is simply caused by the definition of the PGD sequence (see eq. (1.3.1)) which transfers the "relative weight" of a couple of mode to the last coordinate i.e. Y_i for the bivariate problem. The decay here is very fast since this function is easily separable. Finally, Fig. 1.4d displays the same modes obtained through POD, thus of norm 1. The apparent lack of smoothness is due to the coarseness of the mesh but it affects very weakly the accuracy[8]

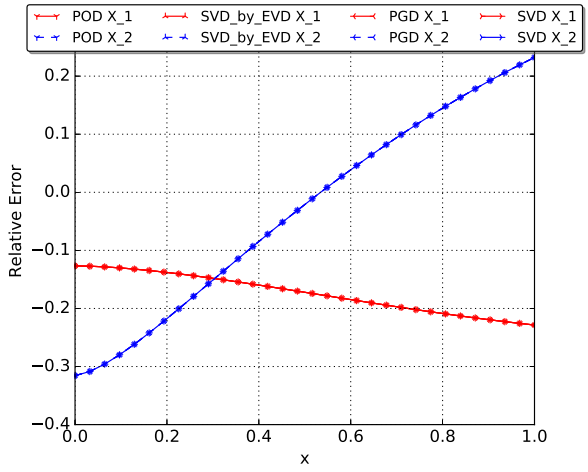
Typical Decrease in approximation error and singular values. The bivariate functions can be sorted in two groups with respect to these decomposition techniques.

Definition 1.4.1 (Exponentially Separable function)**.** *A function is called* exponentially separable *if the decrease in the singular values, thus in the approximation error, is exponential. In other words, a* semi-log *plot of the error is a straight line, regardless of its slope.*

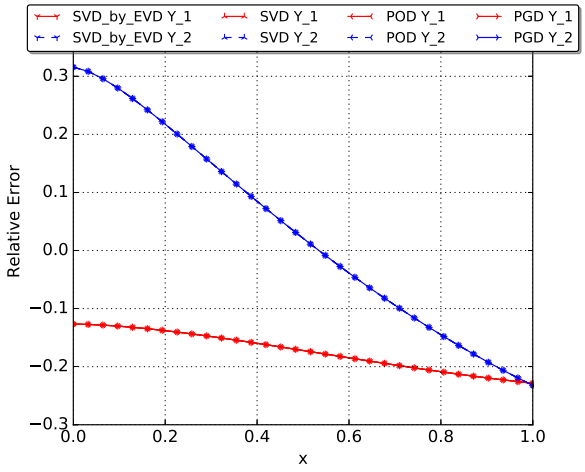
Definition 1.4.2 (Linearly separable function)**.** *A function is called* linearly separable *or weakly separable if the decrease in the singular values, thus in the approximation error, is linear. In other words, a* log-log *plot of the error is a straight line, regardless of its slope.*

These definition will be extended directly to multiparameter functions. Typically, weakly separable function are produced by highly non-linear processes or functions that

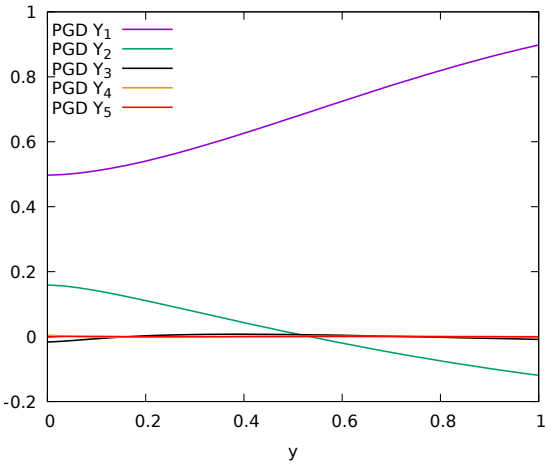
[8] For instance using several grids from 16×16 to 1024×1024 has shown no improvement in the approximation error for 5 modes and a slight decrease in the accuracy with finer grid when using 10 modes. Thus as long as the sampling is fine enough to capture the features of the field $f(x,t)$ refining the grid does not improve the accuracy of the representation at the sampling points.



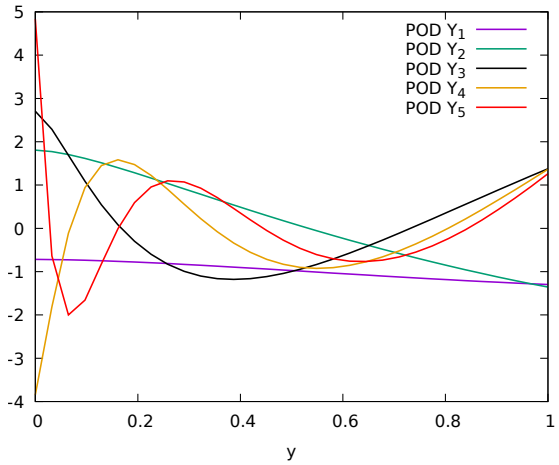
(a) Normalized modes X_1, X_2 with each method.



(b) Normalized modes Y_1, Y_2 with each method.



(c) Modes $(Y_i)_{i=1}^5$ obtained with PGD.



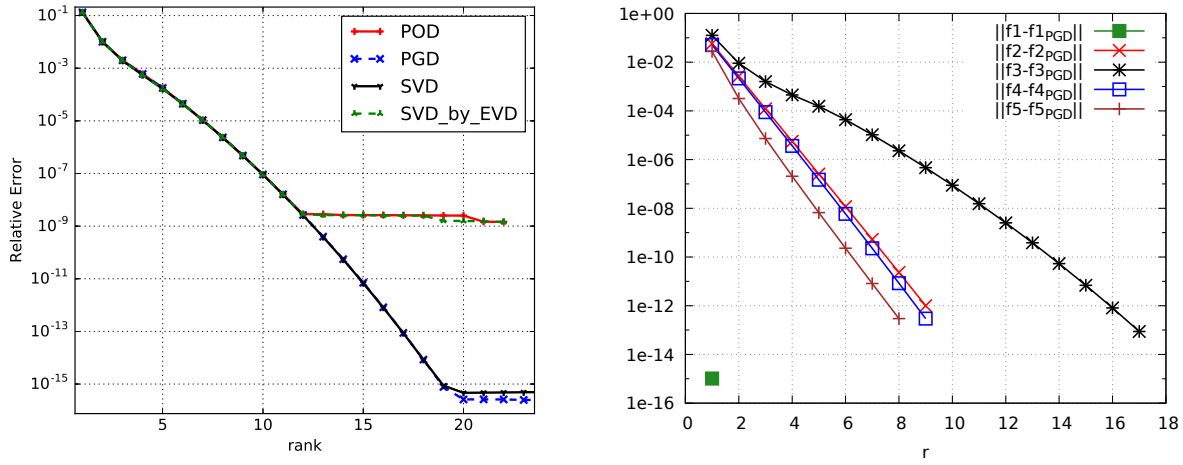
(d) Modes $(Y_i)_{i=1}^5$ obtained with POD.

Figure 1.4: Decomposition modes of f_3.

display a sharp singularity. Thus singular function is often used to replace weakly separable in the literature as well as in this manuscript. Additionally, various levels of separability may be observed depending on the nature of the function. A moderate slope will often be referred to as less separable and an almost linear decay declared weakly separable. Finally, some peculiar function may show two different regions (relative to r) with distinct behavior such as first a sharp exponential decay followed by a milder linear one. This generally fits the properties of the function such as length scale or turbulent behavior in fluid dynamics.

As mentioned in the theoretical paragraphs, a very efficient way to measure the separability of a field is to observe the decay of the singular values. It is also a reliable way to estimate the error decay. Fig. 1.5a presents a comparative view of the decay of the approximation error for f_3 which is a very common function for testing this property. The singular values are not displayed as their behavior is very similar to the error. All four methods are equivalent up to $r \simeq 12$ which is in agreement with the mathematical equivalence shown in the theoretical presentation. However for $r \geq 12$ it seems that the error is stuck in the 10^{-8} regions. This is explained by the ill-conditioning[9] of matrix C

[9] As the conditionning is defined by the ratio of the largest and smallest singular values, it is obvious that the conditionning is very poor since the singular values range from $\mathcal{O}(1)$ and $\mathcal{O}(10^{-16})$.



(a) 2D decompositions error of f_3.

(b) PGD applied on a pool of functions.

Figure 1.5: Approximation error (L^2 or Frobenius norm) for bivariate methods

in eq. (1.2.30) that causes a loss or the orthonormality property of the POD/SVD basis. Most importantly, this is due to the limited computer precision for solving the intermediate eigen value problem for `SVD_by_EVD` and POD. Since $\lambda_{\min} \approx 10^{-16}$ and $\sigma_i = \sqrt{\lambda_i}$, the smallest singular value is $\sigma_{\min} \approx 10^{-8}$ which is also approximately the approximation error.

Table 1.1: Numerical orthonormality of the snapshot POD basis for f3

i	$\|\|X_i\|\|_{L^2}$	$\|\|Y_i\|\|_{L^2}$	(X_i, X_{i-1})	(Y_i, Y_{i-1})
1-8	1.0000	1.000	$\mathcal{O}(10^{-16})$	$\leq 10^{-8}$
9	0.999999	0.9999	4.02E-16	-2.93E-7
10	0.999999	1.000	-1.52E-16	5.16E-6
11	0.999999	1.000	1.68E-16	**-6.77E-6**
12	0.999999	1.000	1.66E-16	**-3.19E-3**
13	1.00000	0.9999	6.24E-17	**0.558**
14	0.999999	0.9999	-8.76E-17	0.934
15-32	0.999999	0.9999	$\mathcal{O}(10^{-16})$	$\mathcal{O}(1)$

Table 1.1 presents a test of the orthogonality of the basis obtained through POD. One can see that the `dsyev` routine preserves the orthonormality of the $\{X_i\}$ basis (the one it is directly computing) but that $\{Y_i\}$ gradually looses orthonormality as i grows. The transition from suitable orthogonality to none takes place on a very limited number of modes, here from 11 to 13 this property is lost with the consequence that the accuracy of the representation reaches a threshold. An efficient solution to overcome this limit is to use a method that ensures this property. One can choose a reorthogonalization technique such as Gram-Schmidt orthogonalization process to the POD basis or alternatively as proposed here rely of recursive algorithm to compute both bases, namely the PGD.

As one can see in Fig. 1.4d, in this case as long as the process converges[10] the approximation error decreases (exponentially here) as the number of enrichment grows. Then we

[10] Convergence of the PGD fixed point alternating direction method is not ensured (especially for weakly separable functions) and may be improved with gradient research for instance. However stopping it at a reasonable number of iteration e.g. 10 or 20, has proven efficient in the many numerical experiments

Table 1.2: Comparison of POD, PGD and SVD for a target error of $\epsilon = 10^{-6}$.

	SVD			PGD			POD		
	n	σ^{n+1}	error	n	σ^{n+1}	error	n	σ^{n+1}	error
f_1	1	1.27E-15	6.68E-16	1	≈ 0	1.02E-15	1	9.22E-17	3.45E-16
f_2	5	7.91E-6	2.96E-7	5	2.12E-7	2.55E-7	5	2.12E-7	2.55E-7
f_3	9	1.04E-5	4.73E-7	9	3.20E-7	4.67E-7	9	4.67E-7	3.20E-7
f_4	4	1.03E-5	2.45E-7	4	2.74E-7	2.07E-7	4	2.74E-7	2.07E-7
f_5	5	3.32E-6	1.67E-7	5	9.20E-8	1.49E-7	5	9.20E-8	1.49E-7

can conclude that this function is separable. Up to $r = 12$ one may choose any of the three presented method as the result are extremely similar. However for the next experiment, shown in Fig. 1.4c, functions f_1 to f_5 have been separated using only the PGD. One can see that all these functions are separable although two functions stand off. $f_1 = xy$ is already separated and the PGD only requires 1 mode to represent it to the machine error. f_3 seems to be less separable than the others as its slope is lower. Nonetheless it clearly displays an exponential decay as the section from $r = 12$ to 17 is straight.

Comparison of the methods. Last the mathematical equivalence of the four methods is tested on the least separable of our synthetic function. Fig. 1.4 and 1.5 let us think that they are also equivalent numerically, at least as long as the POD/SVD is properly solved. Table 1.2 further confirms that statement. Indeed, one can see that the number of modes to reach the target error of 10^{-6} is always the same for all three methods and the observed error is also very close. Meanwhile the σ^{n+1} depend on the scalar product used in these methods which is why POD and PGD versions are close while SVD singular values are 1 or 2 orders of magnitude bigger.

Consequently, one may choose whichever of these three method to separate a bivariate function. However, this must be done knowing the relative limitations of these functions. Having both available in code is highly advisable as their advantages are situation related.

1.4.2 Image compression by decomposition

As stated in the SVD section, these techniques can be used on any kind of data. An interesting example while presenting the data compression aspect of these methods is to apply it to images. Indeed it is efficient to compress large images. Indeed numerical images are stored in many formats but it always boil down to an array of integers representing colors. Let us consider the simpler case of grayscale images, usually stored in 1 byte per pixel. That is to say, the original 4000×3000 pixels grayscale image `"singapore.tiff"` used in Fig. 1.6 is a matrix of the same size which coefficients are integers in $[\![0, 255]\!]$ which means 12×10^6bytes $\approx$ 12Mb without compression. Table 1.3 gives the compression rate for different number of SVD modes retained as displayed in Fig. 1.6. One can see easily that preserving very few modes yields high levels of compression but the image features are not preserved. Indeed, on can see in Fig. 1.6 top two lines[11] that keeping

that I have ran during this research. The "remaining part" of the basis is "caught" by the next enrichment step in an adjustment pattern.

[11]The reader is advised to follow this description in the PDF version as it allows zooming of the row of small pictures.

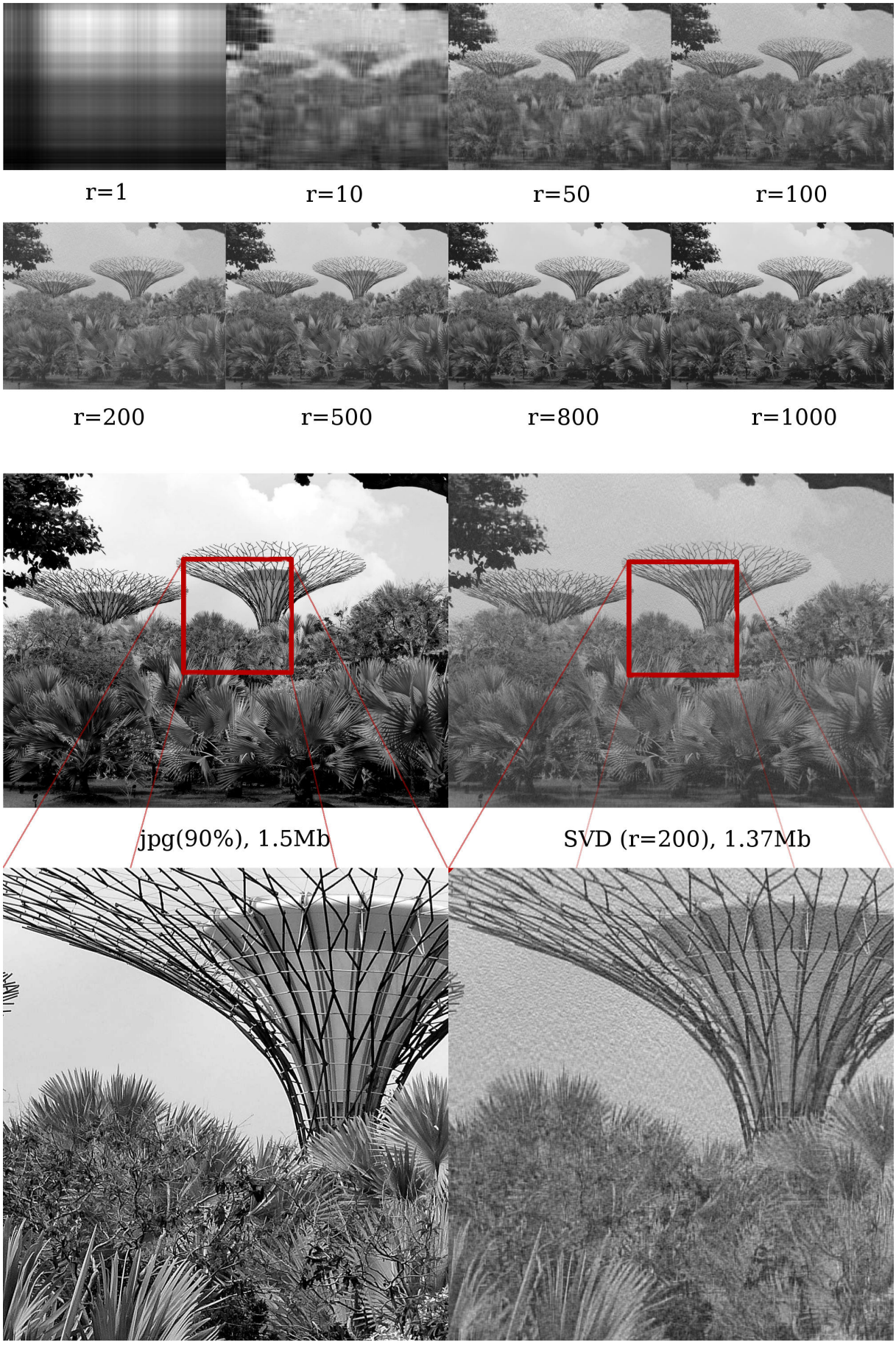


Figure 1.6: A 4000×3000 pixels picture of Singapore Gardens by the Bay compressed through SVD as compared with JPEG compression.

only one mode gives a unrecognizable image. Increasing number of retained modes r leads to gradually better representation, 10 modes is sufficient to perceive the big structures of the image. The big leaves and sharp metallic structures are captured with 50 modes while 100 modes is enough to distinguish palm leaves. This behavior continues up to a few hundreds where all human-eye relevant structures are captured by the SVD compressed image. However at $r = 200$, the image is grainy (especially visible in the sky part) which is striking in the larger SVD image and close-up in the lower part of Fig. 1.6. Adding more and more modes reduces the noise of the image, at $r = 1000$ it is hard to tell that the image has been compressed without any reference point, while the size of the image is still halved as compared with the original uncompressed file. The only difference lies in the contrast level as one can see that the very dark and very bright regions of the image are not as deep as in the original image.

r	SVD size (Mb)	CR (%)	Err. (%)
1	0,01	99.9	41.5
10	0,07	99	31.2
50	0,33	97	25.7
100	0,67	94	22.2
200	1,34	89	17.2
500	3,34	72	9.4
800	5,34	55	5.2
1000	6,68	44	3.2

Table 1.3: Compression rate using SVD on 4000×3000 pixels grayscale image. Where CR is the compression rate and the error is computed with Frobenius norm.

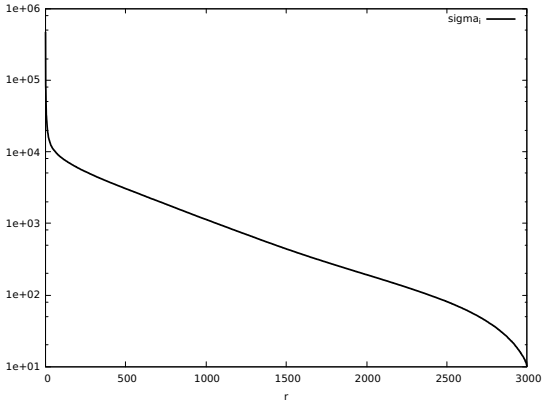


Figure 1.7: Singular values of `"singapore.tiff"`

A very interesting feature of this data lies in the very slow decay of the singular values, shown in Fig. 1.7. Indeed is was chosen on purpose so that no clear directional pattern appeared in the image and all length scales were present. Consequently, the first 50 singular values plummet then the slope become a lot milder with a decay of one order of magnitude per thousand modes. One can assert that the first exponential decay, associated with the large structures of the image, is followed by a linear one due to the profusion of small scales. This is the first example of this behavior shown in this thesis. It will appear again in complex flows and physics problem, either in 2D or 3D. As usual, if all modes are kept the image is exactly recovered. However, there is overhead in the storage space as U is of the same size as the original data and one still needs to store V and Σ.

To conclude on the image compression abilities of SVD, it is fairly efficient for large images as the ratio r/n_{pix} is very small but the method is not well suited for human-eye use. The Frobenius error presented in table 1.3 does not fit with the human experience of the image produced by SVD comparison. Indeed, SVD compares poorly with well established formats such as JPEG which was specifically designed to retain eye sensitivity such as contrast, color depth, etc.

1.4.3 Physics problem data decomposition

We have shown in the previous examples a series of properties of the bivariate decomposition. Now, we focus on data from physics, in particular data obtained by numerically solving partial differential equations (PDEs). A single example is presented as most data from fluid dynamics problem share similar decomposition pattern.

1.4.3.1 Data decomposition of a singular lid driven cavity flow

In this subsection, a brief analysis of the separation properties of the POD on the classical instability problem of a singular lid driven cavity (LDC) at high Reynolds number is given. Further detailed on this very complex flow are given in chapter 5.

The 2D LDC problem is defined on a square domain $\Omega = [0,1] \times [0,1]$ on a time domain $\mathcal{T} = [0,T]$. The upper side of the domain is moving rightward at constant speed U while all other walls are immobile as shown in Fig. 1.8. We chose a "high" Reynolds number, here $Re = 9000$, which means above the first Hopf bifurcation (see 5 for more details) i.e. the flow is unstable. The fluid is at rest at t=0. The data is obtained through a Cartesian grid high accuracy CFD code developed by T.K Sengupta's team (more details available in section 5.1 and [LBA+18]. The vorticity formulation of the Navier-Stokes equation is used, i.e. $\omega = \nabla \times u$, ultimately the non-dimensionalized problem reads

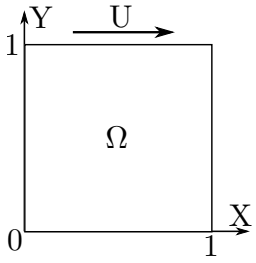


Figure 1.8: Schematic view of the LDC

$$\begin{cases} \nabla^2 \psi = -\omega \\ \frac{\partial \omega}{\partial t} + (\vec{V} \cdot \nabla)\omega = \frac{1}{Re} \nabla^2 \omega \end{cases} \tag{1.4.6}$$

The nature of this flow is very complex and is detailed later in section 5.1. Nevertheless a sample of the vorticity field at time $t = 1900.2$ for $Re = 9800$ is given in Fig. 1.9. The reader can see that the flow is mainly composed of three zones.

Drive This is the dark blue and red region at the top of the cavity. It is characterized by high amplitude vorticity and high shear in the flow, especially near the top right angle.

Core This is the green region of the flow displays very little variation as the exponential contour line levels highlight. Indeed the green color is limited to ± 0.3 around the value of the center of the cavity. This part of the flow has been shown by T.K. Sengupta et al. [SLV09] to display triangular vortices using high accuracy NCCD scheme. These triangular vortices have also been observed for real fluids [CK94, BvH98]. This is the region that presents the most interest for POD analysis as it is complex and very sensitive to numerical error.

Edges/Corners This regions is composed of the three remaining edges and "corner" zones. As for the drive region, they display high levels of vorticity but shear is usually lower since the fluctuations in vorticity are less dramatic. The usual streamlines plot for the LDC would show (nested) recirculation at both lower corners.

In order to have the best accuracy in the POD decomposition, a centering of the vorticity is performed, then all results presented subsequently are produced from the fluctuation of the vorticity field $\omega' = \omega - \bar{\omega}$ where $\bar{\omega}$ the time average of ω is given by

$$\bar{\omega} = \frac{1}{T} \int_{\mathcal{T}} \omega$$

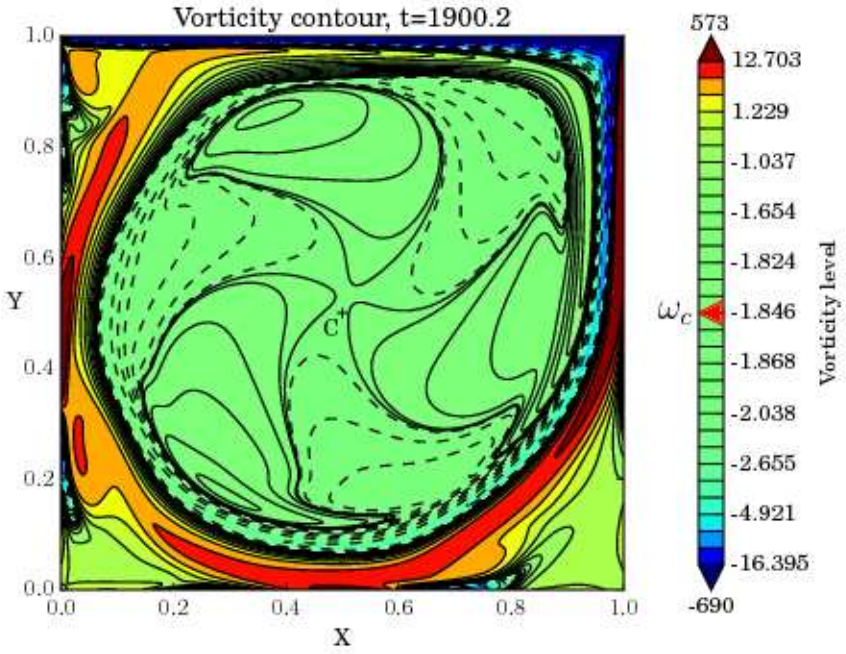


Figure 1.9: Vorticity contour of LDC DNS at $Re = 9800$, $t = 1900.2$. The vorticity contour lines are "centered" to $\omega_c = \omega(0.5, 0.5) \approx -1.84$ to emphasize the orbiting triangular structures in the middle,

Here the POD is applied on the bivariate scalar field $\omega'(\boldsymbol{x}, t)$ in a snapshot fashion, the scalar product used is a measure of enstrophy $||\omega'||^2_L(\Omega)$ instead of the historic approach of kinetic energy [HLB96, NAM+03, Sir87]. In vortex dominated inhomogeneous flows, rotational energy is a better descriptor of POD over translational kinetic energy, as highlighted by Sengupta et al. work [SDS03, Sen12]. Then the correlation matrix is

$$C(t, t') = \int_\Omega \omega'(\boldsymbol{x}, t)\omega'(\boldsymbol{x}, t')\, d\boldsymbol{x} \tag{1.4.7}$$

which is numerically treated by a 2D trapezoidal rule.

Finally, to ease the presentation the POD approximation of the vorticity of rank r reads

$$\omega_r^{\text{POD}}(\boldsymbol{x}, t) = \sum_{i=1}^{r} \phi_i(x) a_i(t)$$

where ϕ_i is normalized ($||\phi_r||_{L^2} = 1$) and the weight of the mode is carried by a_i. In this section the time interval for the POD of the vorticity field is taken in the stable limit cycle range (see 5.1) $T = [1900, 1940]$ with 200 equally spaced snapshots.

Fig. 1.10 shows the decay of the POD approximation error with the number of modes, up to $r \approx 14$ (region A), the decay is exponential while it is linear right to the dashed line (region B). Looking closely, one can see that the decay is actually occurring by pairs of modes in the exponential part. This is due to the nature of flow as the representation of the different physical frequencies yields two POD modes. Indeed one can see that the time modes of Fig. 1.11 are almost identical functions, only separated by a quarter period phase shift. Their amplitude i.e. the measure of their relative contribution to enstrophy is also very close within pairs but changes of magnitude between pairs. Finally the frequency is doubled for every added pair. This behavior is also followed by the spatial modes shown in Fig. 1.12. The number of structures in the core is doubled for every pair and one can

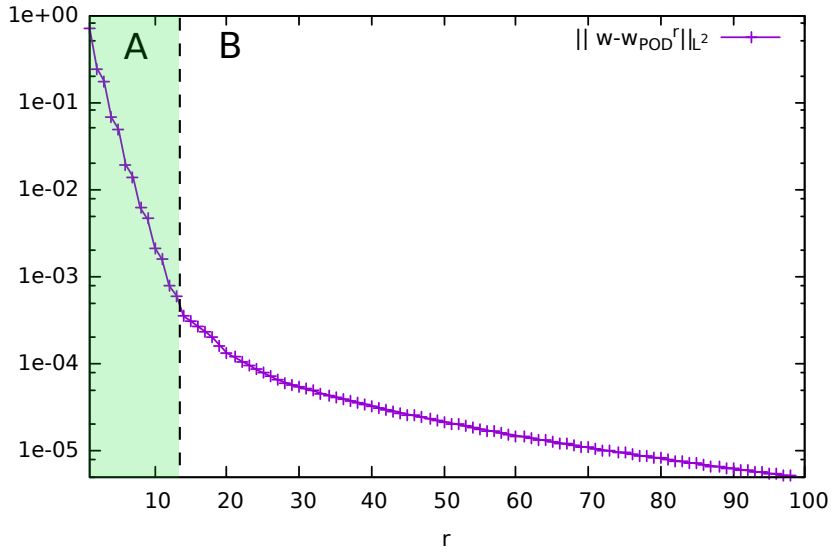


Figure 1.10: POD approximation error decay with the number of modes.

see that the structures are slightly shifted between two members of a pair. The linear decay is associated to numerical noise of the simulation, one can infer that variations below the threshold $r = 14$ or vorticity variation smaller than 10^3 are not representative of the physics of this flow.

This unstable flow example will be studied thoroughly using POD in section 5.

1.4.4 Numerical issues and proposed improvements

The bivariate techniques presented in this section are the foundation of all multivariate methods described in this manuscript. Then it is crucial to target the numerical shortcomings that have been observed. Mainly, one can sort them as accuracy issues (the main focus of this work) and computing efficiency issues. Some solution solve both these problems but may interfere with programming ease. The main difficulty lies in solving the eigenvalue problem (1.2.7) in its various formulations.

Accuracy issues For applications such as building ROMs from separated fields, the accuracy of the basis must be ensured since properties such as orthogonality of the basis are fundamental in many cases (Galerkin projection, etc.). To a lesser extend, loss of orthonormality may prevent the decomposition to converge. Moreover, many of the mathematical properties of these bases rely on the orhtonormality of the bases.

It was observed that the orthogonality is not always preserved for low energy modes. This is generally explained by the poor conditioning of the correlation matrix, typical values of the condition number $\mathcal{K}(A) = ||A^{-1}|| \cdot ||A|| = \frac{\sigma_{\max}}{\sigma_{\min}}$ have been evaluated to 10^{16}. This result is actually expected since the decomposition yields singular values ranging from $\mathcal{O}(1)$ or more to $\mathcal{O}(10^{-15})$. Usual preconditioning techniques such multiplying by the diagonal prove inefficient to reduce the condition number as well as the orthogonality of the basis.

As preconditioning is hopeless due to the nature of the studied matrices, one should carefully choose the eigenproblem solver. For instance, `LAPACK`'s `DGEEV` does not preserve the orthogonality of the eigen vectors beyond the sixth or seventh mode, on the other hand, `LAPACK`'s `DSYEV`, that uses symmetric matrices properties, ensure orthogonality to machine precision. However, the secondary basis (the one obtained by projection of the data on

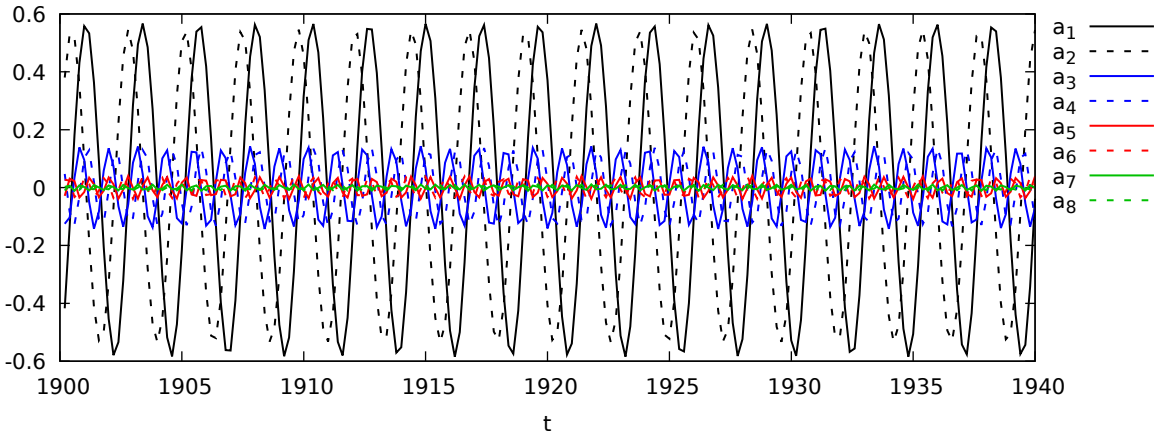


Figure 1.11: The first 8 time modes obtained through POD of the vorticity disturbance field ω' of the LDC for $t \in [1900 : 1940]$. The norm of the couples $\{ \phi_i, a_i\}$ is stored in a_i which is why they tend toward zero.

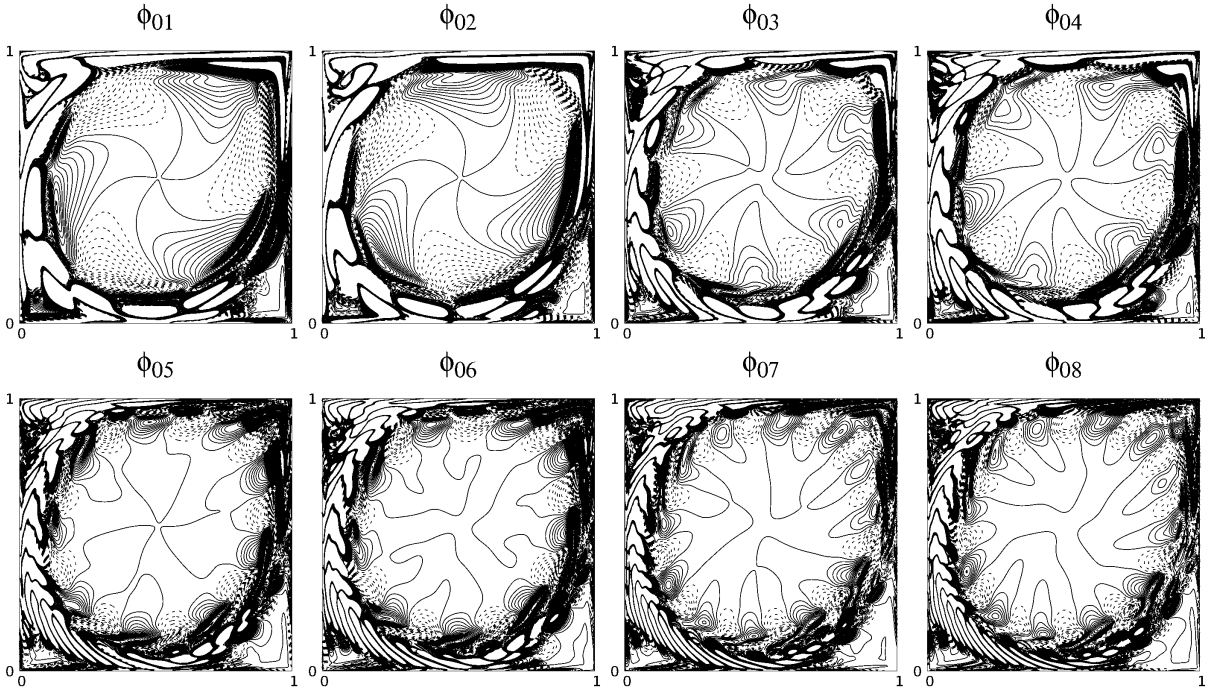


Figure 1.12: The first 8 spatial modes contour obtained through POD of the vorticity disturbance field ω' of the LDC for $t \in [1900 : 1940]$.

the primary basis) does not ensures orthogonality as the modes number grows as shown in Tab. 1.1. Synthetic data exacerbate this phenomenon as numerical simulation output do not provide data with sufficient accuracy to preserve lower modes. Other algorithm were tested performs variably well, `DSYEV` being the upper bound for accuracy. Consequently one must always check the accuracy of the orthogonality if this property is used for further development. For compression purpose this assumption is secondary and the only measure of efficiency is the compression rate.

Efficiency issues. It clearly appears that solving the full eigen problem is generally inefficient regarding the operation count. Indeed, the nature of these decomposition strategies is to truncate the separated representation to the smallest number of modes possible. Then, computing all modes to throw away the vast majority of them, as one would do us-

ing `LAPACK` routines, is inappropriate. However, to the best of my knowledge, most partial eigenproblem solvers rely on iterative processes that are very efficient for sparse matrices or when the user requires very few ($\mathcal{O}(10)$) eigenvectors. In our framework, it is usually interesting (sometimes compulsory) to compute tens of modes to represent accurately complex datasets. So far using full direct `LAPACK` solvers on correlation matrices have proven quick. Regarding our PGD based iterative solver, it turns out that it's efficiency for numerical simulation output of reasonable size ($\mathcal{O}(100MB)$) is poor. Computing time for 10 modes is orders of magnitudes longer than its `LAPACK` counterpart to obtain the full basis. Scaling to bigger problem is simply out of range with the proposed implementation as the number of numerical integration is too big. One way to circumvent these issues is to make sure one solves the eigen problem on the smallest of the dimension, usually using a snapshot method for CFD outputs.

Memory overload. The main limitation to the current version of the library, is actually memory use as some datasets don't fit in `RAM` typically $\mathcal{O}(100GB)$ is overloading memory even on large memory computing nodes.

Proposed Improvement. The following points are possible directions for solving these issues and improve the library.

- Implement a version of the library that does not require loading the full dataset as to compute the decomposition, rather loading periodically chunks of the data to compute parts of the eigen-problem. Iterative methods are the most promising candidates. One could use a block power iterate (that would also allow *parallel computing*) or some improvement of it such as Arnoldi although the implementation for this kind of application might be more difficult. However, the implementation of such techniques must be thought carefully as the alternate direction scheme (PGD) has proved inefficient for large problems.

- In case of overload of `dsyev` processing capacity, one can rearrange data in a multivariate fashion and use tensor decomposition technique. For example instead of separating time and space, one could also separate x and y thus making a trivariate decomposition.

- Implement a parallel version of these algorithms to overcome memory limitation as well as reducing computing time.

Conclusion

In this section, three bivariate data decomposition approaches were presented: SVD, POD and PGD. It was shown that they are different algorithm that produce mathematically equivalent results in the sense that the modes obtained are the same, with the same decay of singular values. This is confirmed by numerical experiments (Fig. 1.4). But they require different computing time, PGD is thus discarded as soon as one requires several modes for large problems. The main difference between these methods lies in the choice of the inner product determines the set in which the orthonormality of the decomposition is ensured. On the one hand, one can choose SVD which is well suited for simple data compression as it does not require any knowledge on the data properties or any grid/integration scheme. Thus it allows to compress any kind of data from images Fig. 1.6 to CFD data. On the other hand, one can prefer POD if interested in inner products that are suited to physical

properties of the data. A typical example is the analysis of CFD data with an energetic norm ($L^2(\Omega)$) or a norm adapted to the properties of the underlying equation (e.g. $H^1(\Omega)$ for NSE) for further use in a ROM by Galerkin projection on the reduced basis.

The presented method have been implemented and thoroughly tested. Consequently they form the basic unit of the multivariate decomposition methods that presented in the next sections.