SAPIENZA
UNIVERSITÀ DI ROMA

A Lipschitzian Global Optimization Algorithm and Machine Learning for Fluid Dynamics

Department of Computer, Control and Management Engineering

PhD degree in Operations Research - XXXIII Cycle

Candidate

Danny D'Agostino

ID number 1247958

Thesis Advisor

Prof. Stefano Lucidi

Co-Advisor

Dr. Matteo Diez

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Operations Research

January 2021

Thesis defended on 19 May 2021
in front of a Board of Examiners composed by:

Prof. Fabio Tardella (chairman)

Prof. Giuseppe Baselli

Prof. Stefano Panzieri

A Lipschitzian Global Optimization Algorithm and Machine Learning for Fluid Dynamics

Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: danny.dagostino@uniroma1.it

To Hideo.

Abstract

The research conducted and resumed in this thesis covers two different topics.

In chapter 1, I focused my research on the development of a new Global Optimization algorithm informed with an estimate of the Lipschitz constant of the objective function. Estimation of the Lipschitz constant is obtained using the tools from the Extreme Value Theory. To extract the information of the local behavior of the objective function, I proposed a clustering strategy to enlighten the algorithm of the local Lipschitz constants.

In chapters 2, 3, and 4, I show my research by developing and applying Machine Learning methodologies to three Fluid Dynamics phenomena of different nature.

Specifically, in chapter 2, I propose a new framework for design space dimensionality reduction for shape optimization based on Probabilistic Linear Latent Variable models. The new framework performs the classical reduction of the number of the design variables, which is crucial to speed up the convergence of the optimization process. Furthermore, It provides the uncertainty of the new geometrical parametrization by introducing a constraint in the optimization problem based on the Mahalanobis distance.

In chapter 3, my research is concentrated on the extraction and the interpretation of highly nonlinear turbulent phenomena measured with the Particle Image Velocimetry technique. Data-driven analysis is carried out for two high Reynolds number vortices flows namely for a uniform and buoyant jets and 4- and 7-bladed propeller wakes.

In chapter 4, I focused on the prediction of the ship motion at a high sea state level. For this application, Deep Learning methods for sequential data such as Recurrent-type Neural Networks have very desirable properties due to the high non linearities present inside the system. Besides the model's predictive performance, the uncertainty information is retrieved from a Bayesian perspective through Variational Inference.

Acknowledgments

Firstly, I would like to thank my supervisors, Prof. Stefano Lucidi and Dr. Matteo Diez, whose support and knowledge were precious in formulating the research questions and methodology.

I would also like to thank all my colleagues of the Multidisciplinary Analysis and Optimization group at the Institute of Marine Engineering of the Italian National Research Council, for their helpful contributions and insightful suggestions during my journey.

I am also grateful to all my Professors of the Department of Computer, Control and Management Engineering at the Sapienza University of Rome for allowing me with their expertise to grow as a research scientist.

Finally, I could not have completed this dissertation without the unconditional love of my family, friends and my beautiful Nicole...I kindly hold you in my heart.

Contents

Chapter 1

Lipschitzian Optimization with a Statistical Estimate of the Lipschitz Constant

In this first chapter, we propose a new Lipschitz optimization algorithm for Global Optimization using a statistical estimate of the Lipschitz constant. The Lipschitz constant estimation procedure is based on the well developed Extreme Value Theory of Statistics. The evaluation of the Lipschitz constant is carried out using the maximum likelihood estimation method for the three parameters reversed Weibull distribution, where the location parameter represents our estimate of the Lipschitz constant. Our contribution is to explore and evaluate the possibility to integrate a statistical estimation of the Lipschitz constant inside a global optimization procedure. This procedure is performed inside a popular deterministic partition-based algorithm for Global Optimization, namely the DIRECT algorithm. Furthermore, to model the local behavior of the objective function, we propose a clustering strategy. In this way, we also provide the optimizer the information of the local Lipschitz constants, exploiting the local trend of the objective function. A new definition for potentially optimal hyperrectangles is provided. The new proposed algorithm is compared against DIRECT with respect to a set of 14 bi-dimensional functions and the high dimensional CEC2014 Benchmark.

1.1 Introduction

In every field of the science, researchers and engineers encounter every day decision processes with the objective to take the action that will maximize/minimize the output of some process. Sometimes, some decisions are not available because there is an underlying reason for which they are not admissible. In mathematics this can be translated in following way

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{1.1}$$

$$\text{subject to: } \mathbf{x} \in \mathcal{F} \tag{1.2}$$

The function f represents the objective or the quantity of interest that we want to optimize. Depending on the nature of the problem this could be a maximization or a minimization problem. The set $\mathcal{F}$ is where all the admissible decision variables $\mathbf{x}$ 'live'. Unfortunately, there are situations where there is not a unique best optimal solution. This because of the particular form of f (e.g convexity) or because we cannot obtain any information from f (e.g. derivatives) to understand if the problem has a unique global solution or not. This happens when a closed form of the objective function f is not available. For this reasons problems of this kind are called *black box*.

Global Optimization (GO) is the field of Operations Research where methodologies and algorithms are developed to find the global solution. During the last 40 years of intense academic research many approaches have been proposed. Basically, GO methodologies divide in two kind of procedures: stochastic and deterministic. Stochastic algorithms are characterized by the introduction of source of randomness, mainly to allow diversity and exploration of the entire feasible set of solutions during the iterations. Among them we recall the Controlled Random Search [99], Genetic Algorithms [26] and Evolutionary Algorithms like Simulated Annealing [68], Particle Swarm Optimization [72] and Covariance Matrix Adaptation Evolutionary Strategy [58] are known to have been effective in many engineering applications. As part of the deterministic approaches, Lipschitz Optimization is based based on a strong mathematical foundation and convergence properties for find the global minimum of a function. Diagonal based strategies [123] and partition based [125, 87, 82] have been proposed. Among them, the DIRECT (DIviding RECTangles) algorithm [69] is one of the most attractive and developed during the recent years due to its effectiveness and simplicity. Many different versions of the DIRECT algorithm have been proposed as [46, 81, 88, 92, 89, 80, 79, 124]. The DIRECT algorithm performs the GO process without the knowledge of the Lipschitz constant. This approach is attractive because the estimate of the Lipschitz constant L is usually very hard to perform and still nowadays an open problem in Mathematics. The knowledge of L, permits inside a GO scheme to compute valid lower bounds on f and consequently drive the exploration based on this information. In this work we try to tackle this challenging problem integrating a statistical procedure proposed in [146] for the estimation of the Lipschitz constant with a partition based strategy for GO. More in detail we integrate the procedure inside the DIRECT algorithm. Our approach use the current information obtained by the DIRECT algorithm to compute the absolute slopes. For definition, the slopes represent a lower bound on the Lipschitz constant of f. From the well developed Extreme Value Theory of Statistics, in case of independent and identically distributed random variables the distribution of the maximum slopes converges asymptotically to the Reverse Weibull distribution. The location parameter or the maximum of the support of the reversed Weibull distribution is our estimate of the Lipschitz constant. The parameters of the reversed Weibull distribution are found using the Maximum Likelihood Estimation procedure. This is demonstrated for univariate function in [146]. The objective is to obtain a good estimate of the Lipschitz constant during the optimization to drive the DIRECT algorithm to a more effective exploration and exploitation of the feasible set of candidate solutions.

Unfortunately, even with the knowledge of the global Lipschitz constant could lead to a slow convergence optimization procedure. For example, the global Lipschitz

constant could represent a bad approximation of the maximum norm of the gradient in regions where f has gentler behavior. In this work we will try to solve this problem through a clustering phase. Our procedure will provide multiple Lipschitz constants estimates. At the end we propose a new definition for the potentially optimal hyperrectangles defining a new sampling criteria in the algorithm.

Finally to assess the numerical capability, we tested our algorithm respect to DIRECT using 14 bi-dimensional functions and a set of 30 challenging high dimensional problems taken from the CEC2014 [78] competition.

1.2 A Brief Introduction to Lipschitzian Optimization

In this work we consider the following GO problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{1.3}$$

$$\text{subject to: } \mathbf{x} \in \mathcal{F} \tag{1.4}$$

where $f : \mathcal{R}^N \to \mathcal{R}$, $\mathbf{x} \in \mathcal{R}^N$ and $\mathcal{F}$ is the feasible region of $\mathcal{R}^N$. The objective function f is supposed to be multiextremal on $\mathcal{F}$ and treated as black box.
A first important result is given by the Weierstrass Theorem that gives a sufficient condition for the existence of the global optimum requiring only the continuity of f and that $\mathcal{F}$ is closed and bounded.

Theorem 1.2.1 (Weierstrass Theorem)**.** *If $\mathcal{F}$ is a non empty closed and bounded set and f is continuous, then a global minimum of f in $\mathcal{F}$ exist.*

Even if the Weierstrass theorem gives to us a first existence condition on the global optimality, this result is not practical from an algorithmic point of view. In order to start to characterize the mathematical properties of the global optima we require the feasible set $\mathcal{F}$ to be *robust*.

Definition 1. *Given $\mathcal{F} \subset \mathcal{R}^N$ nonempty and bounded, then if $\mathcal{F} = Cl(Int(\mathcal{F}))$ then the set $\mathcal{F}$ is robust.*

It follows that $\mathcal{F}$ represents the closure of its nonempty interior. The first characterization of the global minima can be obtained introducing the notion of *level set*, $\mathcal{L} = \{\mathbf{x} \in \mathcal{D} : f(\mathbf{x}) < c\}$ for $c \in \mathcal{R}$.

Theorem 1.2.2. *Given $\mathcal{F} \subset \mathcal{R}^N$ robust and f continuous on $\mathcal{F}$. The vector $\mathbf{x}^\star$ is the global minimizer of f if and only if*

$$meas(\mathcal{L}^\star) = 0 \tag{1.5}$$

with $\mathcal{L}^\star = \{\mathbf{x} \in \mathcal{F} : f(\mathbf{x}) < f(\mathbf{x}^\star)\}$)

Where $meas(\mathcal{L})$ can represents a volume or more in general a Lebesgue measure. From now on we consider the feasible region to be an hyperrectangle or a box $\mathcal{D} = \{\mathbf{x} \in \mathcal{R}^N : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$. The N-dimensional vectors $\mathbf{l}$ and $\mathbf{u}$ represent the lower and upper bound on $\mathbf{x}$.

To describe efficient procedures for find the solution of a global optimization problem, we require a further condition respect to the function f. More in particular we require f to be *Lipschitz continuous* on $\mathcal{D}$.

Definition 2. *Let $\mathcal{D} \subset \mathcal{R}^N$ and $f : \mathcal{F} \to \mathcal{R}$. The function is Lipschitz-continuous on $\mathcal{D}$ if*

$$|f(\mathbf{x}) - f(\bar{\mathbf{x}})| \leq L||\mathbf{x} - \bar{\mathbf{x}}|| \qquad \forall \mathbf{x}, \bar{\mathbf{x}} \in \mathcal{D} \tag{1.6}$$

with Lipschitz constant L of f.

where $||\cdot||$ is the Euclidean norm but it could be any norm of $\mathcal{R}^N$.

The Lipschitz continuity of f is an important assumption because we can obtain a valid lower-bound for the global optimum $f(\mathbf{x}^\star)$. In fact knowing the function value at the point $f(\bar{\mathbf{x}})$ then for any $\mathbf{x} \in \mathcal{D}$ the following relation is valid

$$f(\bar{\mathbf{x}}) - Lh \leq f(\bar{\mathbf{x}}) - L||\mathbf{x} - \bar{\mathbf{x}}|| \leq f(\mathbf{x}) \qquad h \geq \max_{\forall \mathbf{l}, \mathbf{u} \in \mathcal{D}} ||\mathbf{u} - \mathbf{l}|| \tag{1.7}$$

which means that if f is Lipschitz-continuous with Lipschitz constant L, then is also Lipschitz continuous with an overestimate of the Lipschitz constant $\bar{L} > L$.

Now we show that the requirement for f to be Lipschitz-continuous is not particularly restrictive, indeed this class of functions is very general.

Theorem 1.2.3. *If f is continuously differentiable on a open convex set $\mathcal{D}_0$ and $\mathcal{D}$ be a bounded, closed set such that $\mathcal{D} \subseteq \mathcal{D}_0$. Then f is Lipschitz continuous on $\mathcal{D}$.*

Proof. Writing a first order Taylor expansion with $\theta \in [0, 1]$

$$f(\bar{\mathbf{x}}) = f(\mathbf{x}) + \nabla f(\mathbf{x} + \theta(\bar{\mathbf{x}} - \mathbf{x}))^\top (\bar{\mathbf{x}} - \mathbf{x}) \tag{1.8}$$

from which we have

$$|f(\mathbf{x}) - f(\bar{\mathbf{x}})| \leq L||\mathbf{x} - \bar{\mathbf{x}}|| \tag{1.9}$$

where L is given by

$$\max_{\mathbf{x} \in \mathcal{D}} ||\nabla f(\mathbf{x})|| \tag{1.10}$$

□

The maximum norm of the gradient of f can be used as valid Lipschitz constant. The constant L is in general unknown. Computing L maximizing the norm of the gradient of f will be as complex as find the global solution of f and consequently not a viable option.

With the mathematical tools exposed until now is possible to characterize procedures for find global optima of a function. In the next chapter we introduce briefly a general class of global optimization algorithms that sequentially produce partitions of the feasible set $\mathcal{D}$.

1.2.1 Partition Based Strategies for Global Optimization

In this chapter we describe the main properties of a particular kind of deterministic GO algorithms where in each iteration the feasible set $\mathcal{F}$ is divided into partitions. Assuming that the feasible set is given by the box $\mathcal{D}$ we introduce the following definition.

Definition 3. *Given $\mathcal{D} \subset \mathcal{R}^N$ nonempty and robust and I a finite set of indices. A collection of sets $\{\mathcal{D}^i : i \in I\}$ is called a robust partition of $\mathcal{D}$, if $\mathcal{D}^i$ is a robust set $\forall i \in I$. Furthermore*

$$\mathcal{D} = \bigcup_{i \in I} \mathcal{D}^i \tag{1.11}$$

$$\mathcal{D}^i \bigcap \mathcal{D}^j = \delta\mathcal{D}^i \bigcap \delta\mathcal{D}^j \qquad \forall i, j \in I, \quad i \neq j \tag{1.12}$$

where $\delta\mathcal{D}^i$ represents the boundary of $\mathcal{D}^i$.

From Def. 3 we can conclude that the feasible set $\mathcal{D}$ is divided in non-overlapping subsets. Below we describe the main steps of a partition-type scheme for GO.

General Partition Algorithm

Step 1 Set $k = 0$ and the set of indices $I = \{\emptyset\}$ and the initial partition $\mathcal{D}_0 = \mathcal{D}$

Step 2 Select a finite number of sample points $\mathbf{x}^{i_k} \in \mathcal{D}^{i_k}$ with $i_k \in I_k$ and obtain objective function value f for every $i_k \in I_k$

Step 3 Select a subset of indices $I^\star \subset I_k$

Step 4 For every $i_k^\star \in I_\star$ the relative $\mathcal{D}^{i_k^\star}$ is divided such that

$$\mathcal{D}^{i_k^\star} = \bigcup_{j \in I^{i_k^\star}} \mathcal{D}_j^{i_k^\star} \qquad \mathcal{D}_j^{i_k^\star} \subset \mathcal{D}^{i_k^\star} \tag{1.13}$$

For every $i_k^\star \in I^\star$ the new set of indices is given by $I_{k+1} = I_k \cup \{j : j \in I_\star\} \setminus i_k^\star$.

Step 5 Set $k = k + 1$ and return to Step 1

In the first step we initialize the set of indices to an empty set whereas the initial partition to the whole feasible set. Then, we perform for each partition a finite number of function evaluations (not yet defined in which location). In the third step (with a non defined criteria) we select a subset of indices where the relative subset is considered of 'interest'. In the last steps those subsets are further divided and the set of indices updated. This kind of scheme for GO has been object of intense studies as in [122, 82]. To highlight the convergence properties of this kind of procedures is important to highlight the characteristic of the partitions generated during the iterations. A particularly relevant property is the following.

Definition 4. *For each nested sequence of partitions subsets $\{\mathcal{D}^{i_k}\}$, then is called strictly nested if*

$$\mathcal{D}^{i_{k+1}} \subset \mathcal{D}^{i_k} \qquad k \to +\infty \tag{1.14}$$

Another important definition is given by the *everywhere dense* property of a subset

Definition 5. *Let $\mathcal{D} \subset \mathcal{R}^N$. Then a subset $\mathcal{D}_0 \subset \mathcal{D}$ is everywhere dense in $\mathcal{D}$ if for every $\epsilon > 0$ and for every point $\mathbf{x} \in \mathcal{D}$ a point $\tilde{\mathbf{x}} \in \mathcal{D}_0$ exist and such that $\tilde{\mathbf{x}} \in \mathcal{B}(\tilde{\mathbf{x}}, \epsilon)$.*

This implies the following definition.

Definition 6. *Given a nested sequence $\mathcal{D}^{i_k}$ produced by the partition algorithm, then it is strictly nested if and only if*

$$\bigcap_{k=0}^{\infty} \mathcal{D}^{i_k} = \{\tilde{\mathbf{x}}\} \tag{1.15}$$

An advisable property is that a partition type algorithm should produce dense sets only around the global minimizer of f.

In the next section we will introduce a partition based algorithm that has the everywhere dense property. For its simplicity and also its effectiveness in many applications, became during the last years one of the most popular methods of this kind.

1.3 The DIRECT Algorithm

In the previous section we described a general partition scheme for GO. On purpose we did not described where we should sample the points $\mathbf{x} \in \mathcal{D}^i$ and neither what kind of criteria we can use in order to judge a particular partition 'interesting'. The DIRECT algorithm proposed by [69] is a popular approach in order to find the global minimizers $\mathbf{x}^\star$ of f. First of all the DIRECT algorithm divides the search space $\mathcal{D} = \{\mathbf{x} \in \mathcal{R}^N : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$ in hyperrectangles and hypercubes and sample a new point exactly at the center of them. More important is how DIRECT judge a particular interval 'interesting'. This is described in the definition of *potentially optimal* hyperrectangles.

1.3.1 Potentially Optimal Hyperrectangles

The selected partitions for further exploration are called potentially optimal and their definition is given below.

Definition 7. *Given the partition $\{\mathcal{D}^i : i \in I\}$ of the feasible set $\mathcal{D}^i$, where $\mathcal{D}^i = \{\mathbf{x} \in \mathcal{R}^N : \mathbf{l}^i \leq \mathbf{x}^i \leq \mathbf{u}^i\}$ with*

$$\mathbf{x}^i = \frac{\mathbf{u}^i - \mathbf{l}^i}{2} \tag{1.16}$$

and given

$$f_{\min} = \min_{i \in I} f(\mathbf{x}^i) \tag{1.17}$$

then a subset $\mathcal{D}^h$, with $h \in \mathcal{D}^h$ is called potentially optimal if given $\epsilon > 0$ exist a constant $\bar{L}$ such that

$$f(\mathbf{x}^h) - \bar{L}d^h \leq f(\mathbf{x}^i) - \bar{L}d^i \qquad \forall i \in I_k \tag{1.18}$$

$$f(\mathbf{x}^h) - \bar{L}d^h \leq f_{\min} - \epsilon|f_{\min}| \tag{1.19}$$

where d^i represents the distance from the center $\mathbf{x}^i$ to the vertices of $\mathcal{D}^i$.

From Def. 7 is possible to interpret how the DIRECT algorithm 'filters' the potentially optimal rectangles from the other subsets. In fact DIRECT can be seen as a Lipschizian optimization algorithm even if it does not ever try during the iterations to obtain an estimate of the Lipschitz constant. The selection is based on the existence of a Lipschitz constant value that makes a particular interval interesting or not. Following Def. 7 candidates to be potentially optimal hyperrectangles are obtained taking for each d^i, $\forall i \in I$ the hyperrectangle with the lowest function value. Potentially optimal hyperrectangles can be simply obtained by computing the vertices of the lower convex hull defined in $\mathcal{R}^2$ where for each centers $\mathbf{x}$, in the x-axis there is the distance from its vertices and on the y-axis its objective function value. Once we selected the set of potentially optimal hyperrectangles we need to discuss how DIRECT perform the sampling of new points inside them.

1.3.2 Dividing Strategy

We will explain now how is performed the division in case $\mathcal{D}^i$ is an hypercube. The first iteration of DIRECT consist to perform $2N + 1$ function evaluations on the feasible region $\mathcal{D}^0$ which is properly normalized into a unit hypercube. The first point to sample is the center $\mathbf{x}^0$ of the hypercube $\mathcal{D}^0$. Given the N-dimensional euclidean normal basis $\mathbf{e} \in \mathcal{R}^N$ we sample the new points at $x_j^0 \pm \delta e_j$, $\forall j = 1, \ldots, N$. Where δ is $\frac{1}{3}$ the side length of the hypercube. We define

$$r_j = \min\{x_j^0 - \delta e_j, x_j^0 + \delta e_j\}, \qquad \forall j = 1, \ldots, N \tag{1.20}$$

we start to divide $\mathcal{D}^0$ in the order given by r_j and perpendicular to direction e_j. If $r_{j^\star} = \min_{j=1,\ldots,N}\{x_j^0 - \delta e_j, x_j^0 + \delta e_j\}$ we perform the first division perpendicular to the direction of $e_i^\star$. Consequently $x_{j^\star}^0 = \arg\min_{j=1,\ldots,N}\{x_j^0 - \delta e_j, x_j^0 + \delta e_j\}$ will be the center of an hyperrectangle of one side δ and the other remaining $N-1$ sides of size 3δ. This means that the sample with the lowest objective function value will be always the center of an hyperrectangle. The procedure ends producing hyperrectangles and hypercubes in the feasible set $\mathcal{D}^0$. In case $\mathcal{D}^i$ is an hyperrectangle this is divided only respect to its longest side j and sampling at $x_j^i \pm \delta e_j$.

1.3.3 Convergences Properties of DIRECT

The properties of DIRECT, given the sequences $\mathcal{D}^{i_k} = \{\mathbf{x} \in \mathcal{R}^N : i_k \in I_k\}$, can be summarized in the following proposition.

Proposition 1. *The DIRECT algorithm has the following properties*

1. *The partitions $\{\mathcal{D}^{i_k}\}$ produced are strictly nested*
2. *For every $\tilde{\mathbf{x}} \in \mathcal{D}$, the partitions $\{\mathcal{D}^{i_k}\}$ are everywhere dense for $k \to \infty$*

$$\bigcap_{k=0}^{\infty} \mathcal{D}^{i_k} = \{\tilde{\mathbf{x}}\} \tag{1.21}$$

Those asymptotic properties are satisfied by the DIRECT algorithm because at each iteration the subset of maximum distance from its vertex is always potentially

optimal, even if the objective function value is arbitrary high. The DIRECT algorithm in general performs well. The strengths are in the simplicity of the implementation which also allows parallel computation quite naturally since the sampling process can be performed independently for each potentially optimal hyperrectangle. The main weakness here is that DIRECT as described in [70] is that sometimes get stuck in local minima for many iterations before it starts to sample in other part of the search space for find the global minimizers.

1.3.4 Convergences Properties of DIRECT with an Overestimate of the Lipschitz Constant

In case an upper bound of the Lipschitz constant is available, the convergences properties of DIRECT become stronger. Following the results showed in [32] is possible to redefine the concept of potentially optimal hyperrectangles in *strongly potentially optimal* hyperrectangles.

Definition 8. *Given the partition $\{\mathcal{D}^i : i \in I\}$ of the feasible set $\mathcal{D}^i$ and a constant $\bar{L} > 0$, then a subset $\mathcal{D}^h$, with $h \in \mathcal{D}^h$ is called strongly potentially optimal if given $\epsilon > 0$*

1. *exist a constant $\bar{L}^h \in (0, \bar{L})$ such that*

$$f(\mathbf{x}^h) - \bar{L}^h d^h \leq f(\mathbf{x}^i) - \bar{L}^h d^i \qquad \forall i \in I_k \tag{1.22}$$

$$f(\mathbf{x}^h) - \bar{L}^h d^h \leq f_{\min} - \epsilon |f_{\min}| \tag{1.23}$$

2. *The constant $\bar{L}$ satisfies*

$$f(\mathbf{x}^h) - \bar{L} d^h \leq f(\mathbf{x}^i) - \bar{L} d^i \qquad \forall i \in I_k \tag{1.24}$$

where $f_{\min}$ is given in Def. 7.

From the Def. 8, only potentially optimal hyperrectangles can be candidates to be strongly potentially optimal hyperrectangles. In this case the procedure shows a strong convergence properties.

Proposition 2. *Given the constant $\bar{L}$ in Def. 8 and $I^\star$ the indices of potentially optimal hyperrectangles. Then*

1. *The algorithm produces at least a strictly nested sequence of sets $\{\mathcal{D}^{i_k}\}$*
2. *If for a global minimum $\mathbf{x}^\star \in \mathcal{X}^\star$, exist an index $\bar{k}$ such that if the subset $\mathcal{D}^{j_k}$, with $j_k \in I_k$ is the subset where $\mathbf{x}^\star \in \mathcal{D}^{j_k}$, then $\forall k \geq \bar{k}$*

$$f(\mathbf{x}^{j_k}) - L d^{j_k} \leq f(\mathbf{x}^\star) \tag{1.25}$$

then every strictly nested partitions $\{\mathcal{D}^{j_k}\}$ produced satisfies

$$\bigcap_{k=0}^{\infty} \mathcal{D}^{j_k} \subseteq \mathcal{X}^\star \tag{1.26}$$

3. *If for every global minimum* $\mathbf{x}^\star \in \mathcal{X}^\star$*, exist a constant* $\alpha > 0$ *and an index* $\bar{k}$ *such that if the subset* $\mathcal{D}^{j_k}$*, with* $j_k \in I_k$ *is the subset where* $\mathbf{x}^\star \in \mathcal{D}^{j_k}$*, then* $\forall k \geq \bar{k}$

$$f(\mathbf{x}^{j_k}) - Ld^{j_k} < f(\mathbf{x}^\star) - \alpha d^{j_k} \tag{1.27}$$

then for every $\mathbf{x}^\star \in \mathcal{X}^\star$ *a partition* $\{\mathcal{D}^{j_k}\}$ *exists and such that is strictly nested and*

$$\bigcap_{k=0}^{\infty} \mathcal{D}^{j_k} = \{\mathbf{x}^\star\} \tag{1.28}$$

The proofs are given in [32] where also a stopping a criteria for the algorithm is provided. Is crucial to understand how we can obtain an estimate of the Lipschitz constant in the DIRECT algorithm. A simple estimate for L could be, $\forall i_k, j_k \in I_k$

$$\max_{k=1,\dots,\bar{k}} \left\{ \frac{|f(\mathbf{x}^{i_k}) - f(\mathbf{x}^{j_k})|}{||\mathbf{x}^{i_k} - \mathbf{x}^{j_k}||} \right\} = \bar{L} \leq L \quad i_k \neq j_k, \quad \forall k = 1, \dots, \bar{k} \tag{1.29}$$

the estimate of the Lipschitz constant is the maximum value of the absolute slopes computed across all the sampled points obtained from DIRECT until iteration $\bar{k}$. In general this can represent just a lower bound of the true value of the Lipschitz constant L. Unfortunately, if the value of $\bar{L}$ is far from the true Lipschitz constant will affect negatively the convergence properties of the algorithm because the procedure becomes more aggressive towards exploitation with the risk that the optimization algorithm could find only a local minima. On the other hand an upper bound of L exaggeratedly high would mean obtain a procedure similar or identical to the DIRECT algorithm. The problem to find an estimate of the Lipschitz constant is in general very challenging, even more if we have only partial informations about the the objective function f (e.g no derivatives) and that should be carried out inside an optimization routine. In the following section we exploit a different method for the estimate of L based the Extreme Value Theory developed in Statistics.

1.4 A Statistical Estimate of the Lipschitz Constant

In the last part of the previous section we highlighted that a possible estimate for the Lipschitz constant L of a function f could be the maximum of all the absolute slopes. This in general represents a lower bound on the true Lipschitz constant L (i.e the maximum norm of the gradient of f). If we see the slopes as a random variables could be interesting to analyse the statistical properties of their maxima.

1.4.1 A Note to Extreme Value Theory

Extreme Value Theory (EVT) is the theory of modeling and quantifying the uncertainty of extreme events namely those events which occur with very small probability. Probabilistic EVT is important for modeling an high variety of applications from finance to natural phenomena like rainfall and pollution.

The framework is developed considering a sequence of independent random variables $Y_1, \dots, Y_M$ and we are interested in the asymptotic behavior of the $\max\{Y_1, \dots, Y_M\}$ for $M \to \infty$. In this setting three main important probability distributions are defined.

Type 1 Gumbel-Type Distribution

$$G(y) = \exp\{e^{(y-u)^w/v)}\} \tag{1.30}$$

Type 2 Frechet-Type Distribution

$$G(y) = \begin{cases} 0 & y \leq u \\ \exp\{(-(\frac{y-u}{v})^{-w})\} & y > u \end{cases} \tag{1.31}$$

Type 3 Reversed Weibull-Type Distribution

$$G(y) = \begin{cases} \exp\{(-(\frac{u-y}{v})^{w})\} & y < u \\ 1 & y \geq u \end{cases} \tag{1.32}$$

where the $u \in \mathcal{R}$, $v > 0$ and $w > 0$ are the *location*, *scale* and *shape* parameters respectively. The classical Weibull distribution can be obtained from the distribution of Y. Those distributions are called *Extreme Distributions* because can be obtained asymptotically for $M \to \infty$ of the greatest value among M independent random variables identically distributed. More formally [144]

Proposition 3. *Given a sequence of independent random variables $Y_1, \ldots, Y_M$ with cumulative distribution function F and suppose that exist two sequence $v_n > 0$ and $u_n \in \mathbb{R}$ such that the following limit converges*

$$\lim_{M\to\infty} p\left(\frac{\max\{Y_1, \ldots, Y_M\} - u_n}{v_n} \leq y\right) = F^M(v_n y + u_n) = G(y) \tag{1.33}$$

If the limit converges then $G(y)$ is called Generalized Extreme Value Distribution and has the following distributional form

$$G(y) = \exp\left\{-(1+wy)^{-\frac{1}{w}}\right\} \tag{1.34}$$

To obtain a non degenerate limiting distribution is necessary to perform a linear transformation. A further development is given by the work of Gnedenko [52].

Proposition 4. *Given a sequence of independent random variables $Y_1, \ldots, Y_M$ with $y^\star = sup\{y : F(y) < 1\}$ be the population maxima, then the normalized distribution $G(y)$ assumes the following forms*

Type 1 (w=0) Gumbel-Type Distribution if and only if

$$\lim_{\beta\to 0^-} \frac{1 - F(\beta - yh(\beta))}{1 - F(\beta)} = e^{-y} \tag{1.35}$$

with $h(\beta) = \frac{\int_\beta^{y^\star} 1-F(r)dr}{1-F(\beta)}$

Type 2 (w>0) Frechet-Type Distribution if and only if $y^\star = \infty$ and for all $y > 0$

$$\lim_{\beta\to\infty} \frac{1 - F(\beta y)}{1 - F(y)} = y^w \tag{1.36}$$

Type 3 (w<0) Reversed Weibull-Type Distribution if and only if $y^\star$ is finite and for all $y > 0$

$$\lim_{\beta \to 0^+} \frac{1 - F(y^\star - \beta y)}{1 - F(y^\star - y)} = y^w \tag{1.37}$$

Gnedenko also showed that these conditions are necessary, as well as sufficient, and that there are no other distributions satisfying the stability postulate [76]. Given the basic theoretical properties of the EVT we can discuss how to use these tools for the estimation of the Lipschitz constant.

1.4.2 Lipschitz Constant Estimation through the EVT

In this section we explain how to use tools from the EVT theory to obtain an estimate of the Lipschitz constant inside the DIRECT algorithm. We showed that using Eq. 1.29 could be a first estimate of the Lipschitz constant. Following our discussion, we can interpret the absolute slope as a random variable Y and we are interested in the statistical properties of the maximum slope, $\max\{Y_1, \ldots, Y_M\}$. If we suppose that the slopes are independent and identically distributed we have the following relation

$$\begin{aligned} p(\max\{Y_1, \ldots, Y_M\} \leq y) &= p(Y_1 \leq y, Y_2 \leq y, \ldots, Y_M \leq y) \\ &= p(Y_1 \leq y) p(Y_2 \leq y) \ldots p(Y_M \leq y) \\ &= F^M(y) \end{aligned} \tag{1.38}$$

which cannot be used in practice since we don't know the distribution F. The EVT gives to us asymptotic conditions on the limiting distribution of $\max\{Y_1, \ldots, Y_M\}$ for $M \to \infty$. Since the slopes are bounded above (by the maximum norm of the gradient) the distribution of the maximum slopes can be modeled by a reversed Weibull distribution. The reversed Weibull distribution is given below

$$p(y|u, v, w) = \frac{w}{v} \left(\frac{u - y}{v} \right)^{w-1} e^{-\left(\frac{u-y}{v}\right)^w} \tag{1.39}$$

defined for $y \leq u$ and $w, v > 0$. The maximum of its support, which is represented by the location parameter u is our estimate of the Lipschitz constant L. The usage of the EVT for the estimation of the Lipschitz constant for univariate functions in [146] where the points from which the slopes are computed are chosen uniformly at random. Some numerical assessment of the same methodology is also showed for multivariate function in [151]. Here we are interested to use this approach inside an optimization algorithm and then the sampling scheme is necessarily different. A simple way to produce an estimate for the Lipschitz constant is to save at each iteration k the maximum of the slopes computed across the points sampled from DIRECT. Once we have enough data, at some iteration $\bar{k}$ we can fit the reversed Weibull distribution to find the location parameter u. Successively we evaluate the strongly potentially optimal hyperrectangles defined in Def. 8. The location u, scale v and shape w parameters of the reversed Weibull distribution can be found through a maximum likelihood estimation (MLE) procedure. The log-likelihood function is

given by

$$\ln p(\mathbf{y}|u,v,w) = N[\log(w) - w\log(v)] + \sum_{k=1}^{\bar{k}}(w-1)\ln(u-y_i) - \left(\frac{u-y_i}{v}\right)^w \quad (1.40)$$

where $\mathbf{y} = (y_1, \ldots, y_{\bar{k}})$. Then the following nonconvex bound constrained optimization problem should be solved

$$\max_{u,v,w} \quad \ln p(\mathbf{y}|u,v,w) \quad (1.41)$$

$$\text{subject to:} \quad v_{lb} \leq v \leq v_{ub} \quad (1.42)$$

$$w_{lb} \leq w \leq w_{ub} \quad (1.43)$$

$$u_{lb} \leq u \leq u_{ub} \quad (1.44)$$

where the bounds for the three parameters are fixed as follows: the scale parameter v must be greater than zero and the upper bound can be fixed to a value arbitrarily large, the location parameter u should be grater than the maximum observation $\max(\mathbf{y})$ and as the scale parameter the upper bound can be fixed to an high value and the shape parameter w a more carefully treatment of its bounds should be performed. In fact, for $w < 1$ the likelihood function approach $+\infty$ as the location parameter u reach the maximum of the observations $\max(\mathbf{y})$. Since a solution of this kind it does not seems to be reasonable, the lower bound for w is $w_{lb} = 1$. An upper bound $w_{ub} = 10$ gives enough flexibility on the shape of the estimated probability distribution.

1.4.3 Multiple Lipschitz Constants Estimates using Clustering

Until now we focused our discussion on the estimate of the global Lipschitz constant. Now we want consider that in general a function f could have a very different behavior across its domain. The value of the global Lipschitz constant is given mainly by the slopes sampled in regions where the objective function has an high rate of change. Consequently could represents a bad overestimate of the local Lipschitz constant function in regions where f shows a 'flatter' behavior.

To solve this problem we propose a clustering strategy applied to the absolute slopes. To give local information on the rate of change of the objective function, the slope associated to the partition $\mathcal{D}^i$, it is given by the maximum of the slopes computed across the nearest neighbor of $\mathbf{x}^i \in \mathcal{D}^i$. The univariate clustering procedure will drive the assignment of the estimated Lipschitz constant across the different partitions generated by DIRECT.

Let's define $\mathbf{s} = (s_1, \ldots, s_M)$ where

$$s_i = \frac{|f(\mathbf{x}^i) - f(\mathbf{x}^j)|}{||\mathbf{x}^i - \mathbf{x}^j||} \quad \forall i,j \in I \quad (1.45)$$

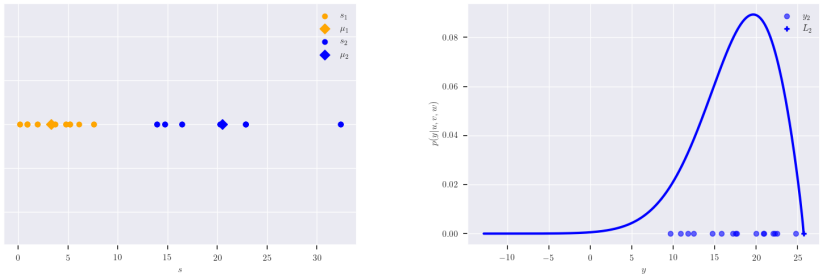
where $\mathbf{x}^j$ is the nearest point (in Euclidean distance terms) of $\mathbf{x}^i$. In case multiple points have the same minimum Euclidean distance from $\mathbf{x}^i$ then we take the maximum of the slopes. Then a clustering algorithm is applied on $\mathbf{s}$ noticing that we are solving a univariate problem. Many clustering approaches are available in literature but

because we are solving a univariate clustering problem we can take advantage of this aspect in order to solve the clustering phase more efficiently. A widely adopted method for divide the data in different partitions is given by solving the k-means problem. In our case we want to find the C centroids μ_c, such that $\sum_m^M \sum_c^C z_{mc}(s_m - \mu_c)^2$ is minimum, where $z_{mc} \in \{0, 1\}$ is the indicator variable. This problem is in general NP-hard [3] and then usually solved using heuristics [83]. In the one-dimensional case the k-means problem can be solved to the optimality in polynomial time through Dynamic Programming [145, 149] in $\mathcal{O}(M^2C)$ which can be further improved in $\mathcal{O}(M \log M + MC)$ as described in [56].

After the clustering procedure is performed we need to extract information about the maximum slopes and the Lipschitz constant representative of each cluster. This is explained in depth the next section.

1.5 A New Proposed Algorithm

Now we will describe step by step our proposed procedure for the estimation of multiple Lipschitz constants L_c, $\forall c = 1 \dots, C$ as summarized Alg. 1. The first iteration ($k = 0$) we produce the first M partitions with defined by the DIRECT algorithm. At $k = 1$ we perform the clustering respect to the slopes obtaining as output the slopes partitioned into C different sets $\mathbf{s}_c^k$, the indices belonging into each cluster $I_{k,c}$ and the centroids μ_c^k. We don't have enough data to perform the MLE procedure and then our first estimate of the Lipschitz constants is given by the maximum of each cluster of slopes as shown in line 6. Successively, given our C estimates L_c^k we evaluate Def. 9. In line 10, our algorithm sampled P points which they have satisfied Def. 9. We compute, in line 11 the slopes respect to this new P points and then assigned to the clusters conditionally to the minimum distance respect to the clusters centroids μ_c, $\forall c = 1, \dots, C$. This procedure divide as before the set $\mathbf{s}^k = (s_1, \dots, s_P)$ in partitions $\mathbf{s}_c^k$, $\forall c = 1, \dots, C$. The maximum of each set $\mathbf{s}_c^k$ is added into the set $\mathbf{y}_c^k$ as defined by the operator '$\frown$' in line 12. The set $\mathbf{y}_c^k$ is composed by all the maximum slopes that are computed along the iterations of our procedure so that its cardinality will increase as k increase. Then a loop respect to



(a) Clustering the slopes

(b) Fitting the reversed Weibull distribution

Figure 1.1. Graphical resume of Alg. 1

the number of clusters is performed in line 13. If we have enough data inside the

set $\mathbf{y}_c^k$ we perform the maximum likelihood estimation fitting a reversed Weibull distribution. Notice that in this particular work we will perform the MLE procedure only for the cluster composed by the highest slopes (namely for $c = C$) to obtain an estimate of the global Lipschitz constant. Otherwise, for the index $c < C$ we fix the estimate for the lower Lipschitz constants respect to the maximum slope (line 18) as before (line 6 and 7). Once the for loop is finished, we update the location of the centroids performing the clustering procedure. Inside the set $\mathbf{y}_c^k$ we always include the maximum of each new updated cluster (line 20). The number of clusters C is chosen respect to the silhouette metric [105] and it is allowed only to increase during the iterations and never to decrease. In the following definition we provide a new selection criteria for the hyperrectangles.

Definition 9. *Given the partition $\{\mathcal{D}^i : i \in I\}$ of the feasible set $\mathcal{D}$, where $\mathcal{D}^i = \{\mathbf{x} \in \mathcal{R}^N : \mathbf{l}^i \leq \mathbf{x}^i \leq \mathbf{u}^i\}$ and given*

$$f_{\min} = \min_{i \in I} f(\mathbf{x}^i) \tag{1.46}$$

given the clustering assignment c, a scalar $\epsilon > 0$, and $\alpha \in (0, 1)$ the partition $\mathcal{D}^j$ is selected if at least one of the following conditions is satisfied

1. *For $j \in I_{m,k}$ where $I_{m,k} = \{i_k \in I_k : d_k^i = d_{\max}^k\}$ then*

$$f(\mathbf{x}_c^j) - L_c^j d^j \leq f(\mathbf{x}_c^p) - L_c^p d^p \qquad \forall p \in I_m \tag{1.47}$$

   where d_{max}^k is the distance from the vertices of the largest hyperrectangle at iteration k.

2. *For $j \in I_k$ then*

$$f(\mathbf{x}_c^j) - L_c^j d^j \leq f(\mathbf{x}_c^p) - L_c^p d^p \qquad \forall p \in I_k \tag{1.48}$$

$$f(\mathbf{x}_c^j) - L_c^j d^j \leq f_{\min} - \epsilon f_{\min} \tag{1.49}$$

3. *For $j \in I_k$ then*

$$f(\mathbf{x}_c^j) \leq f(\mathbf{x}_c^p) \qquad \forall p \in I_k \tag{1.50}$$

$$f(\mathbf{x}_c^j) - L_c^j d^j \leq f_{\min} - \epsilon f_{\min} \tag{1.51}$$

The estimate of the Lipschitz constant for every $i \in I_k$ with $\alpha \in (0, 1)$, is given by

$$L_c^i = \bar{L}_c \alpha^i + s^i (1 - \alpha^i) \tag{1.52}$$

where $\bar{L}_c$ represents the estimate of the Lipschitz constant for all the partitions with clustering assignment c produced from Alg. 1, $\alpha^i = d^i / d_{\max}$ with $d_{\max}$ the overall largest distance from the vertices of the hyperrectangle and s_i as given in Eq. 1.45.

From Def. 9 an hyperrectangle is selected for further exploration if satisfies at least one of three conditions. In the first condition, among the hyperrectangles of maximum size at current iteration k we select the one of minimum lower bound as showed in Eq. 1.47. Consequently, in every iteration an hyperrectangle of maximum

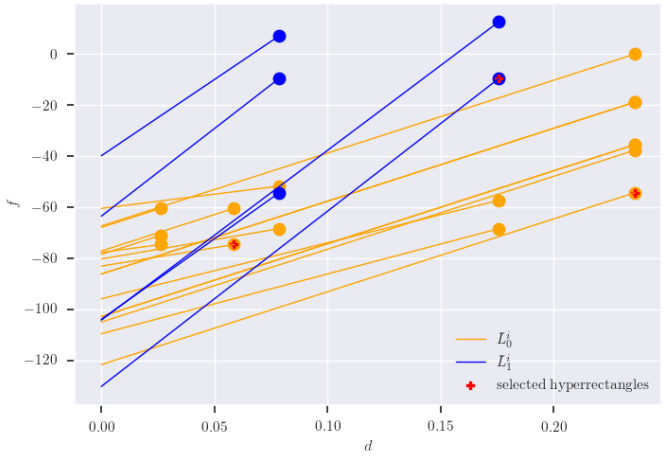


Figure 1.2. Graphical interpretation of Def. 9

size is always selected such that the everywhere dense property of our procedure holds. In the second condition, we select the hyperrectangle that obtains the lowest lower bound respect to all the other partitions Eq. 1.48. The Eq. 1.49 says that the lower bound must allow a minimum improvement. With the last condition we assure that the hyperrectangle with the lowest objective function value for which the Eq. 1.51 holds is selected.

In order to evaluate these conditions we need an estimate the Lipschitz constant. This is given in Eq. 1.52 through a convex combination of the estimate of the Lipschitz constant in its cluster c and slope associated to point $\mathbf{x}^i$ as shown in Eq. 1.45. The coefficient $\alpha^i = d^i/d_{\max}$ where $d_{\max} = d_0$, ensure that when $d^i << d_{\max}$ the slope s^i can provide a good estimate of the behavior of f in the neighbor of $\mathbf{x}^i$. Otherwise, if $d^i \approxeq d_{\max}$, the slope could provide a inexact information and in this case the Lipschitz constant associated with the cluster c will dominate the sum. Consequently, at each iteration at most three hyperrectangles are selected as shown in Fig. 1.2. Observing Fig. 1.2, is interesting that the hyperrectangle with the lowest lower bound is quite far from the potentially optimal hyperrectangles defined by DIRECT. This means that our algorithm can take quite different trajectories in the domain during the optimization respect to the DIRECT algorithm.

Algorithm 1: Lipschitz Constants Estimation Procedure

Input: $M = 0$, $k = 0$, $I_0 = \{\emptyset\}$

if $k = 0$ **then**
  Perform the iteration with DIRECT and $k = k + 1$;
else if $k = 1$ **then**
  For every $i_k \neq j_k$ compute $s_i = \frac{|f(\mathbf{x}^{i_k}) - f(\mathbf{x}^{j_k})|}{||\mathbf{x}^{i_k} - \mathbf{x}^{j_k}||}$ where $\mathbf{x}^{j_k}$ is the nearest neighbor of $\mathbf{x}^{i_k}$;
  Fix the number of clusters C and perform the clustering respect to the slopes $\mathbf{s}^k = (s_1, \ldots, s_M^k)$ obtaining the assignment set $I_{c,k} = \{i \in I_k : s_i \in \mathbf{s}_c^k\}$, the clusters sets $\mathbf{s}_c^k$ and the centroids μ_c^k, $\forall c = 1, \ldots, C$;
  For each cluster $\mathbf{s}_c^k$, take the maximum $y_c^k = \max(\mathbf{s}_c^k)$;
  Fix the Lipschitz constant $\bar{L}_c^k = y_c^k$, $\forall c = 1, \ldots, C$ and evaluate Def. 9;
  Fix $k = k + 1$;
else
  Across the new P points sampled based on Def. 9 compute the slopes respect to their nearest point obtaining P slopes, $\mathbf{s}^k = (s_1, \ldots, s_P)$;
  Assign the slopes respect to the nearest centroid μ_c, $\forall c = 1, \ldots, C$ obtaining new clusters $\mathbf{s}_c^k$;
  For every subset of slopes $\mathbf{s}_c^k$ assigned to the cluster c take its maximum $y_c^k = \max(\mathbf{s}_c)$ and append it to the set $\mathbf{y}_c^k \frown \max(\mathbf{s}_c^k)$, $\forall c = 1, \ldots, C$;
  for $c \leftarrow 1$ **to** C **do**
    if $|\mathbf{y}_c^k| \geq 10$ **then**
      Perform the MLE procedure respect to $p(\mathbf{y}_c^k | u, v, w)$ described in Eq. 1.40 and solve the optimization problem;
      Obtain the estimate $u^\star$ of the Lipschitz constant for the cluster c namely $\bar{L}_c^k$;
    else
      Fix the estimate of the Lipschitz constant for the cluster c with $\bar{L}_c^k = \max(\mathbf{s}_c^k)$;
  Fix the number of clusters C and perform the clustering respect to all $M^k + P$ the slopes $\mathbf{s}^{k+1} = (s_1, \ldots, s_{M^k+P})$ updating the assignment set $I_{c,k} = \{i_k \in I_k : s_i \in \mathbf{s}_c^k\}$, the clusters $\mathbf{s}_c^k$ and the centroids μ_c, $\forall c = 1, \ldots, C$;
  For each cluster $\mathbf{s}_c^k$, append the maximum inside the set of maxima $\mathbf{y}_c^k \frown \max(\mathbf{s}_c^k)$;
  Evaluate Def. 9 ;
  Fix $k = k + 1$ and $M^k = M^{k-1} + P$;

1.6 Numerical Results

In this section we present a first numerical evaluation of our approach that we simply call ALG. The algorithm ALG will confronted respect to the classical DIRECT algorithm with the objective to interpret better its behavior. The set-up for the parameters ALG are the following: the maximum number of cluster allowed is $C = 8$, the optimization process for the MLE of the reversed Weibull distribution is carry out using a multi-start Quasi-Newton method for bound constrained optimization [17] using 125 start points defined using a evenly spaced grid inside the domain. To evaluate the possibility to increase the number of clusters we evaluate the silhouette metric [105].

1.6.1 Numerical Results on Bi-dimensional Test Functions

The first numerical experiment is performed respect to well known bi-dimensional functions. The stopping criteria is fixed when the distance from the global minimum is less than $1e-04$. In Tab. 1.1 we reported the results obtained for ALG and DIRECT. In general the numeric performance of ALG is acceptable respect to

Table 1.1. Resume of the numerical results obtained on simple test functions $N = 2$. The letter F means that the algorithm exceed 1000 function evaluation without reaching the global minimum. Numerical values are the function evaluation needed to reach the global minimum satisfying the predefined accuracy.

Test Functions	ALG	DIRECT
Rosenbrock	F	F
Beale	F	**125**
Tsi. Tang	**97**	165
Branin	**121**	179
Six Hump Camel	**87**	123
Keane	**113**	129
McCormick	**107**	197
Schwefel n.2	**201**	279
Michalewicz	**95**	103
Rastrigin	**77**	103
Eggholder	**960**	F
Matyas	401	**89**
Sphere	**73**	117
Drop-Wave	**583**	F

DIRECT. The algorithm ALG shows a quite faster performance on the identification of the basin of attraction of the global minimum respect to DIRECT. On the Beale, Rosenbrock and the Matyas functions, ALG spends many function evaluations near the global minimum but in a region where the Hessian is ill-conditioned. For this set of functions, this seems the major drawback of ALG. The Eggholder and the Drop-Wave functions are the most challenging to optimize because the first has a large number of local minima and the latter for the highly chaotic behavior. For this two functions DIRECT fails while ALG successfully reach the global minimum. In general, this big difference in the performance between DIRECT and ALG happens the selected hyperrectangle from Def. 9 are not potentially optimal for Def. 7. In ALG, we give high priority to the selection hyperrectangles that have the minimum

lower bound at its vertices among the others even if it is not potentially optimal for DIRECT. In Fig. 1.3 and 1.4 the are sampled points generated by ALG and DIRECT inside the domain of each function considered. For bi-dimensional functions, is possible to visualize the cluster assignments of every point sampled by ALG. A lighter color is associated with low value of slopes while a darker color means that the point belongs in a cluster where the slopes are higher.

1.6.2 Numerical Results on the CEC 2014 Benchmark Functions

In this section, we test our algorithm ALG together with DIRECT using challenging set of functions defined in [78] for the special session and competition on single objective real-parameter numerical optimization CEC 2014. Due to the limited computational resources the maximum number of function evaluations is fixed to $500 * N$ where N is the dimensionality of the problem which (the maximum number of function evaluation allowed in the CEC 2014 competition where it is fixed to $10000 * N$). In Tab. 1.2 the results obtained for $N = 10$ and $N = 30$. The algorithm's performance is assessed respect to the logarithm of the absolute distance from the global minimum, $\log(|f - f^*|)$. To summarize the results, the algorithm

Table 1.2. Resume of the numerical results obtained from the CEC 2014 benchmark functions].

	$N = 10$		$N = 30$	
Test function	DIRECT	ALG	DIRECT	ALG
F1	16.198684	**16.197484**	20.315485	**18.239063**
F2	19.171719	**19.166686**	24.196359	**13.421720**
F3	**8.472994**	9.718675	16.101067	16.101067
F4	2.102766	2.102766	5.543248	**4.457505**
F5	**3.020435**	3.022501	3.034615	3.034615
F6	**1.218654**	1.301020	3.028976	**2.849214**
F7	-0.049974	**−0.119645**	3.138123	**0.794640**
F8	**2.832221**	3.396261	4.794331	**4.455406**
F9	-2.912076	**−3.626542**	-0.144042	**−1.561879**
F10	6.593762	6.595137	8.103852	**8.075270**
F11	7.134420	7.134420	8.453546	**8.298118**
F12	**0.351366**	0.479983	**0.094866**	0.466494
F13	-0.918999	-0.918999	0.034501	**−0.276188**
F14	**−0.984985**	-0.691434	0.279046	**0.078249**
F15	0.546084	**0.343636**	**3.396552**	4.134187
F16	**1.320170**	1.364323	**2.513360**	2.517890
F17	14.540135	14.540135	17.702782	17.702782
F18	19.171787	**17.966430**	21.323664	**21.323656**
F19	16.277759	16.277759	17.819702	17.819702
F20	27.960897	27.960897	**28.914176**	29.614788
F21	16.782134	**11.247926**	**19.082994**	19.212062
F22	27.563391	27.563391	**21.214213**	22.139225
F23	5.555192	5.555192	5.373590	5.373590
F24	5.301019	5.301019	5.301418	5.301418
F25	5.298593	5.298580	5.299440	5.299440
F26	5.298317	**3.892529**	5.298317	5.298317
F27	5.298442	5.298442	5.298481	5.298481
F28	5.299119	5.299081	5.300983	5.300983
F29	10.237105	**8.240411**	20.594835	**17.680589**
F30	16.330736	**15.224995**	20.877168	**19.509255**

ALG performs better than DIRECT on 33% and 46% for $N = 10$ and $N = 30$ respectively. The DIRECT algorithm performs better than ALG for the 23% and 20% for $N = 10$ and $N = 30$ respectively. The remaining percentages no differences at the end of the optimization routine is reported, even if in general by looking at Fig. 1.5, 1.6, 1.7,1.8 ALG shows a better convergence speed.

1.7 Conclusions and Future Work

In this chapter we presented a first tentative to explore the possibility to use the Extreme Value Theory tools from Statistics to estimate global Lipschitz together with a clustering approach to exploit the local variation of the objective function, during the iteration of a GO algorithm. The clustering phase provides, depending on the number of clusters, multiple Lipschitz constants estimates. At the end, we provided a new criteria for the selection of the partitions to sample.

At the end we compared our approach respect to the DIRECT algorithm on simple bi-dimensional functions and on a more challenging and high dimensional test case. The numerical results obtained from our algorithm are encouraging but still some modifications are required and addressed in the future work. For example, as discussed in [151] find conditions on the objective function f for which the Gnedenko condition is satisfied is still an open problem. More in particular, the function defined by the difference between its global Lipschitz constant and the absolute slopes function must to be a regularly varying function [116]. Furthermore, our approach based on k-means provides an hard assignment of the slopes into a particular cluster. Could be interesting to explore the possibility to obtain a soft assignment in terms of probability. This could be achieved using for example a Gaussian Mixture Model (GMM) at cost of a more expensive procedure.

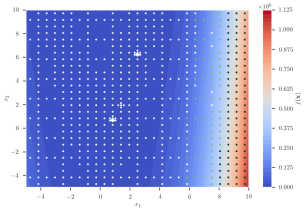
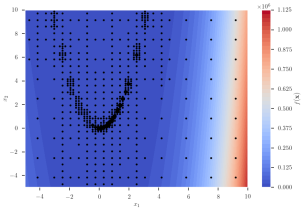
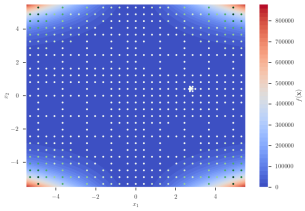
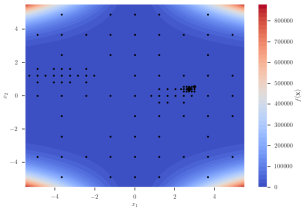
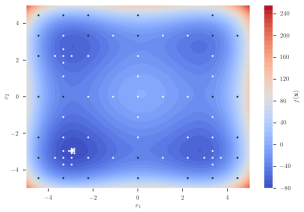
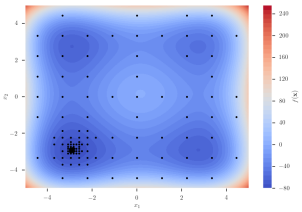
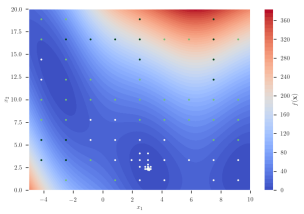
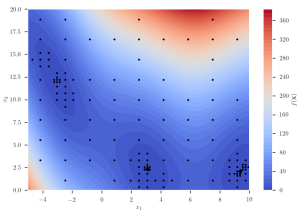
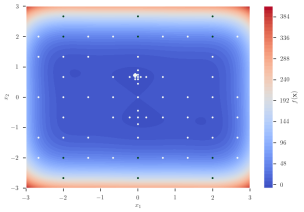
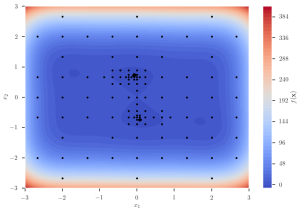
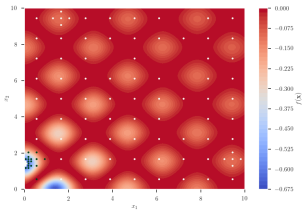
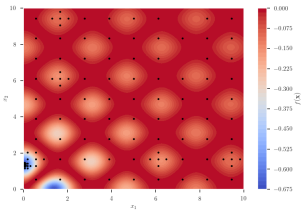
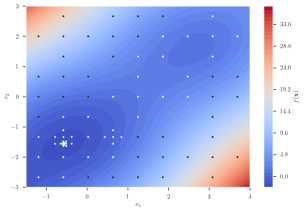
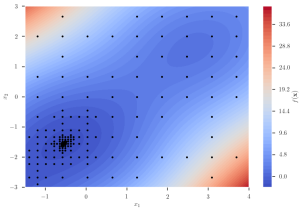
Function Type	ALG	DIRECT
Rosenbrock		
Beale		
Tsi. Tang		
Branin		
Six Hump Camel		
Keane		
McCormick		

Figure 1.3. Behavior of DIRECT and ALG for the simple test functions.

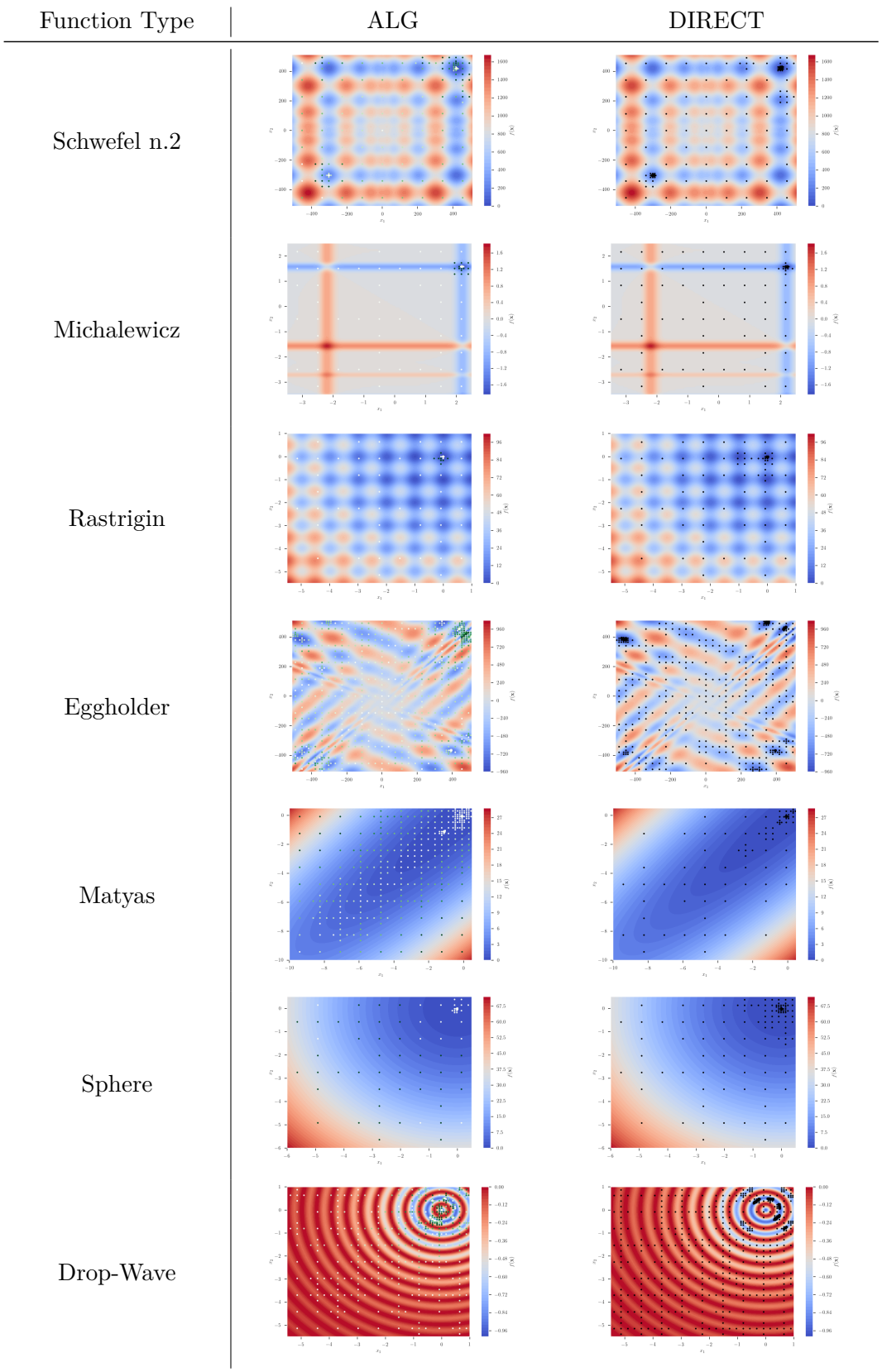
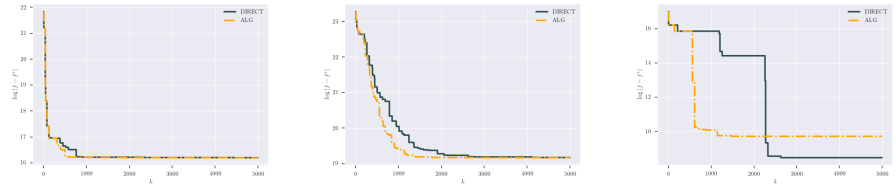
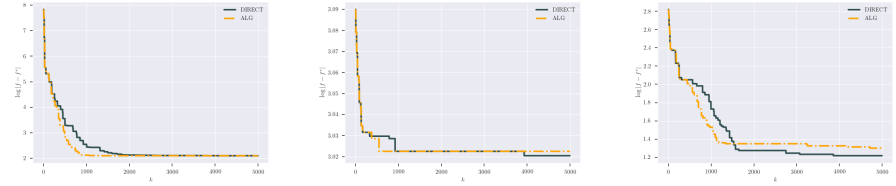


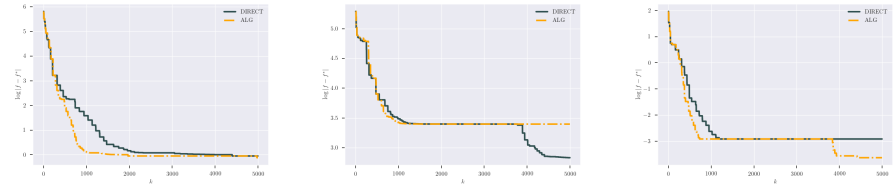
Figure 1.4. Behavior of DIRECT and ALG for the simple test functions.



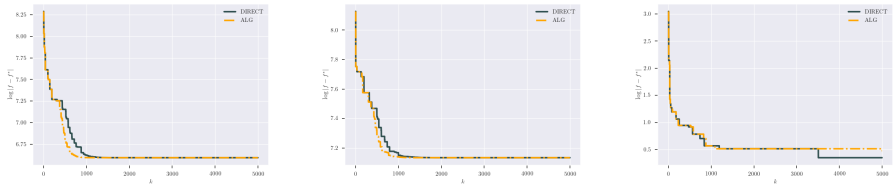
(a) F1 (b) F2 (c) F3



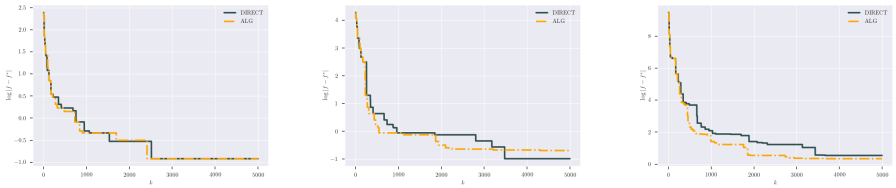
(d) F4 (e) F5 (f) F6



(g) F7 (h) F8 (i) F9

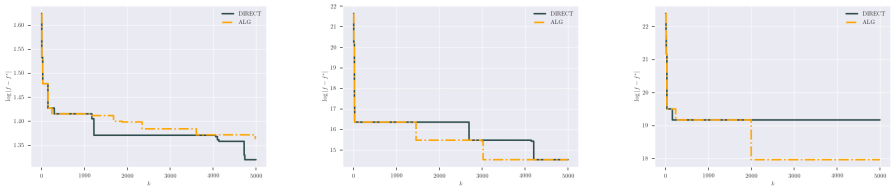


(j) F10 (k) F11 (l) F12

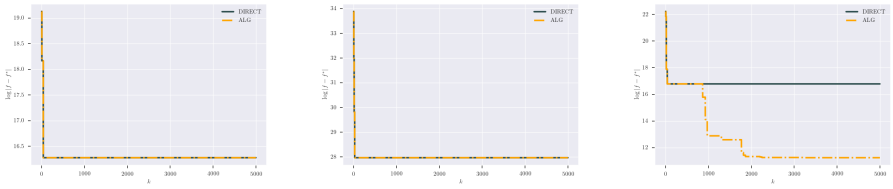


(m) F13 (n) F14 (o) F15

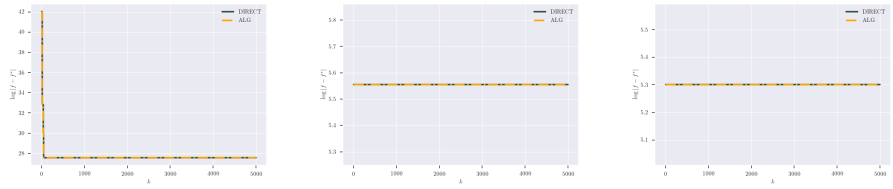
Figure 1.5. Convergence for the first 15 CEC 2014 benchmark functions with $N = 10$



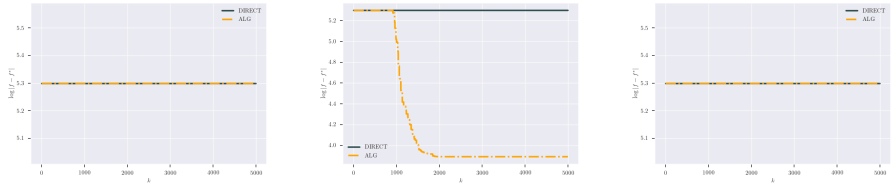
(a) F16 (b) F17 (c) F18



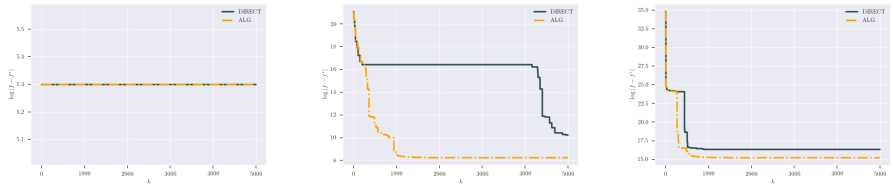
(d) F19 (e) F20 (f) F21



(g) F22 (h) F23 (i) F24

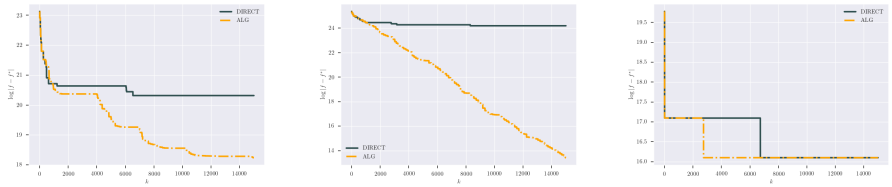


(j) F25 (k) F26 (l) F27

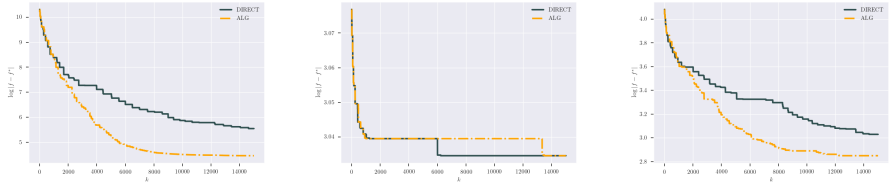


(m) F28 (n) F29 (o) F30

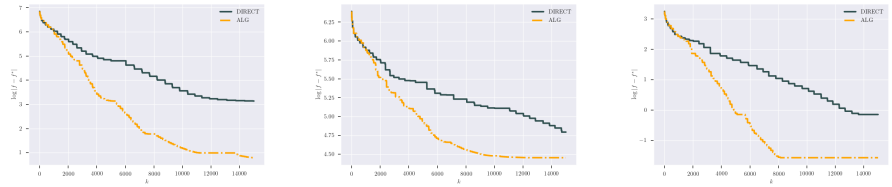
Figure 1.6. Convergence for the last 15 CEC 2014 benchmark functions with $N = 10$



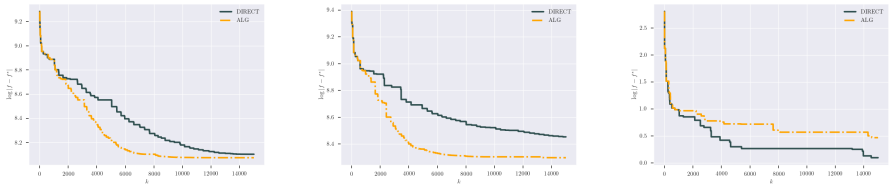
(a) F1 (b) F2 (c) F3



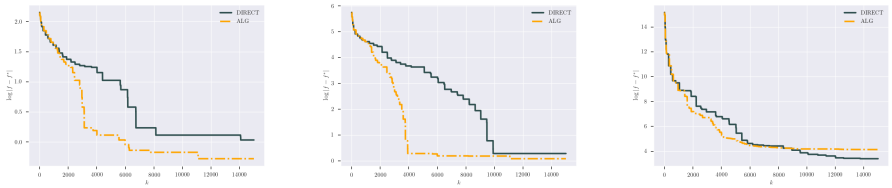
(d) F4 (e) F5 (f) F6



(g) F7 (h) F8 (i) F9

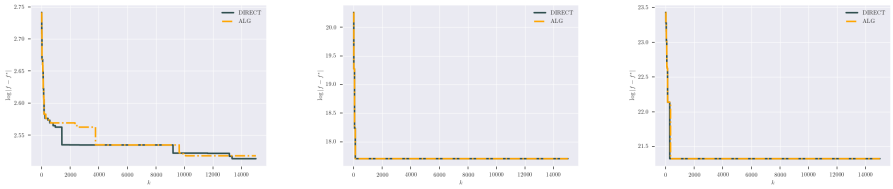


(j) F10 (k) F11 (l) F12

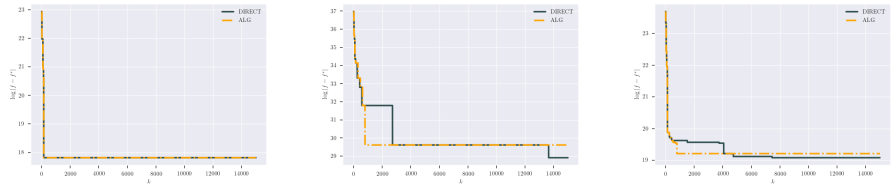


(m) F13 (n) F14 (o) F15

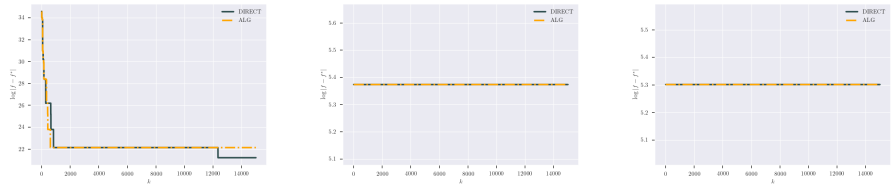
Figure 1.7. Convergence for the first 15 CEC 2014 benchmark functions with $N = 30$



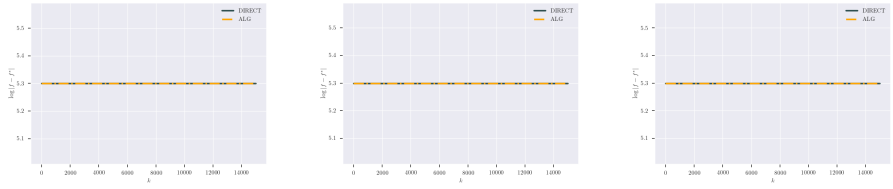
(a) F16 (b) F17 (c) F18



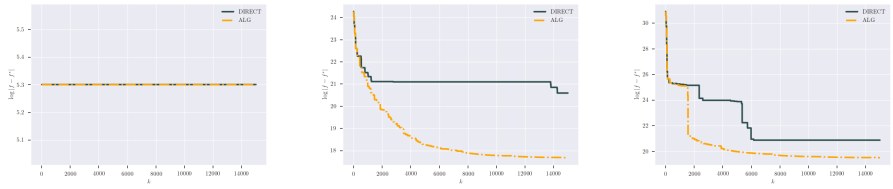
(d) F19 (e) F20 (f) F21



(g) F22 (h) F23 (i) F24



(j) F25 (k) F26 (l) F27



(m) F28 (n) F29 (o) F30

Figure 1.8. Convergence for the last 15 CEC 2014 benchmark functions with $N = 30$

Chapter 2

Probabilistic Linear Latent Variable Models for Shape Optimization

In this chapter, we propose a new framework for design space dimensionality reduction for shape optimization. Our approach produces a new reduced representation of the full original design space assessing the geometrical variance of the shape modification. At the same time, we assure that the reduced parameterization is invariant with respect to the original design space. This is achieved by learning the probability distribution of the data and adding a new constraint inside the optimization model. This constraint is based on the computation of the Mahalanobis distance where the inverse of the covariance is estimated by the probabilistic latent variable models: Factor Analysis and Probabilistic PCA. The procedure is demonstrated for hull shape optimization of the DTMB 5415 model an early and open to a public version of the USS Arleigh Burke destroyer DDG 51, extensively used as an international benchmark for shape optimization problems.

2.1 Introduction

Optimization, thanks to the contribution of decades of academic research became nowadays is one of the most important and precious tool for engineers in industry. In engineering, Multidisciplinary Design Optimization (MDO) focuses on the employment of numerical optimization to perform the design of systems that involve a number of disciplines or subsystems [131]. The crucial aspect in MDO is the full interaction across disciplines in order to find the best *design* which is relevant in many different fields as: aerospace, naval, automotive, mechanical, civil engineering.

One of the first historical example comes around the mid-1970s, where for a structural design problem [113] they combined finite elements and mathematical programming routines for two and three dimensional structural systems as truss, triangular membrane and shear panel elements considering static loading conditions as stress, displacement and member size constraints designing the beginning of new era in engineering design processes.

In the last decades we obtained an incredible progress of the computational power

and mathematical tools, but the complexity of the engineering design processes have also reached an high level of complexity that makes the design optimization process still challenging. This is especially prominent in Simulation Based Design Optimization (SBDO) when time consuming high-fidelity simulators are considered.

Furthermore, one of the most complex challenge is how to deal with high-dimensional large design spaces, when computationally-expensive black-box functions are used for the performance analysis and a global optimum is sought after. Potential design improvements significantly depend on dimension and extension of the design space. Even if efficient Global Optimization algorithms have been proposed [69, 72, 90] and applied with success to SBDO, finding a potentially global optimal solution within reasonable computational time/cost remains a critical issue and a technological challenge. Additionally, Uncertainty Quantification (UQ) of complex applications is computationally very demanding, especially if high-order statistical moments and/or quantiles need to be assessed as in robust and reliability-based design optimization. Both global optimization and UQ are affected by the *curse of dimensionality* as the algorithms' complexity and computational cost rapidly increase with the problem dimension. In this context, shape optimization research has traditionally focused on shape and topology parameterization, as critical factors to achieve the desired level of design variability [108, 13, 110].

The choice of the shape parameterization technique has a large impact on the practical implementation and the success of the optimization process. Shape deformation methods have been an area of continuous and extensive research within the fields of computer graphics and geometry modeling. Consequently, a wide variety of techniques has been proposed during recent years [127]. Several techniques have been developed and applied [110], such as: basis vector methods [94], domain element and discrete approaches [77], partial differential equation [14], CAD-based [150], analytical [59], polynomials [57] and the popular free-form deformation (FFD) [115]. In order for the SBDO to avoid the curse of dimensionality and be successful, the parameterization method must efficiently describe the design variability with as few variables as possible.

A Linear dimensionality reduction model based on the Principal Component Analysis (PCA) (also known as Proper Orthogonal Decomposition (POD) or Karhunen Loeve Expansion (KLE)) have been applied for local reduced-dimensionality representations of feasible design regions [100, 101]. Linear models have been developed with focus on design-space variability and dimensionality reduction for efficient optimization procedures. A method based on the KLE and POD has been formulated in [33], and similarly in [50] for the assessment of the shape modification variability and the definition of a reduced-dimensionality global model of the shape modification vector, for arbitrary modification methods. No objective function evaluation nor gradient is required by the method, as this is entirely based on the concept of geometric variance. KLE/PCA methods have been successfully applied for deterministic [18, 38, 121, 30, 42, 86] and stochastic [34, 35] hull form optimization of mono-hulls and catamarans in calm water and waves, respectively. Similarly, [97] have applied POD to airfoil shape optimization via singular value decomposition (SVD) of an airfoil geometric-data library. Those methods improve the shape optimization efficiency by reparametrization and dimensionality reduction, providing the assessment of the design space and the shape parameterization before

optimization and/or performance analysis are carried out. The assessment is based on the geometric variability associated to the design space, making the method fully off-line and computationally very efficient and attractive, as no simulations are required. The methodology has been extended allowing effective dimensionality reduction in presence of high non linearities for shape optimization in [23, 25, 22] and for a physics informed formulation in [24, 117, 118].

In this work we want to extend the current methodology such that the uncertainty related to the new reduced parameterization is taken into account during the optimization process. This is a crucial aspect because the new reduced representations of the original design variable could sometimes produce shape modifications completely different respect to the original design space parameterization. Also, this could indicate that the reduced design space is indeed 'larger' than the original design space. We propose a new optimization model where the uncertainty of the geometries produced from the new reduced representation (or latent space) is estimated through the computation of the Mahalanobis distance [85]. This is achieved coupling the dimensionality reduction process with a density estimation procedure that learns the probability distribution of the data and consequently the uncertainty associated. In this setting, we propose a new SBDO framework based on Probabilistic linear latent variable models such as Factor Analysis (FA) [7, 132] and Probabilistic PCA [106, 138]. The main assumption here, is that the data is generated by a Gaussian distribution since the two models are part of the Linear Gaussian Models family [107]. The Gaussian assumption is satisfied when the geometrical modification is linear respect to the design variables such in Free Form Deformation [115], NURBS [95, 96] or Radial Basis Function [126]. In this case, the geometries are generated by a linear combinations of uniform distributed design variables so that the design space follows approximately a Gaussian distribution as a direct application of the Central Limit Theorem (CLT) [114].

To assess the performance of this approach, two global optimization algorithm namely DIRECT and Bayesian Optimization are applied for a hull shape optimization of the DTMB 5415 model an early and open to public version of the USS Arleigh Burke destroyer DDG 51, extensively used as an international benchmark for shape optimization problems (e.g., [36, 37]).

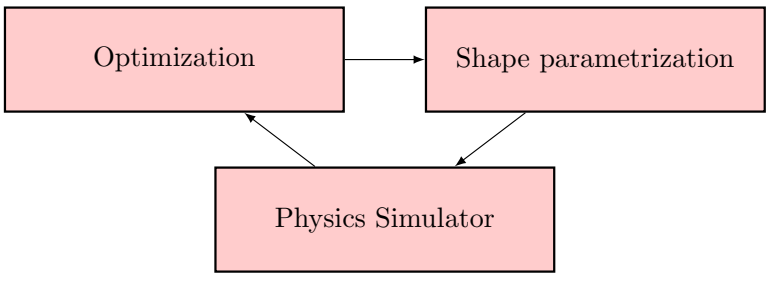


Figure 2.1. Scheme for the SBDO framework.

2.2 The Simulation Based Design Optimization Framework

In the following sections we'll describe the main components of the SBDO framework as shown in Fig. 2.1.

2.2.1 The Optimization Problem

The first block of the SBDO framework is the optimization process. Generally the underlying optimization model that we want to solve, given the design variable $\mathbf{v} \in \mathcal{R}^M$ is the following

$$\min_{\mathbf{v}} \quad f(\mathbf{x}(\mathbf{v})) \tag{2.1}$$

$$g_j(\mathbf{x}(\mathbf{v})) \leq a_j \qquad \forall j = 1, \ldots, J \tag{2.2}$$

$$v_m^{\text{lb}} \leq v_m \leq v_m^{\text{ub}} \qquad \forall m = 1, \ldots, M \tag{2.3}$$

here the objective function Eq. 2.1 represents the quantity that we want to minimize. With the vector $\mathbf{x}(\mathbf{v})$ we represent a geometry $\mathbf{x} \in \mathcal{R}^D$ where its modification depends on the design variable vector $\mathbf{v}$. The optimization is performed respect to the design variable $\mathbf{v}$ since it's responsible of the geometrical modification. The function evaluation is usually performed by a physical simulator because a closed form of the objective function is not available. Usually, black box optimization problems are solved with global optimization algorithms. Also, the time required to perform a single function evaluation could be in the order of hours and consequently the computation of the gradient could be exaggeratedly computationally demanding. In this case global derivative-free optimization methods are usually more attractive. We'll talk in more details about the simulator in the coming sections.

The Eq. 2.18 and Eq. 2.3 represents the constraints. The function $g(\mathbf{x}(\mathbf{v}))$ represents a geometrical constraint that force the optimizer to produce admissible shape modifications. The geometrical constraints are evaluated before the function evaluation is performed. Constraints of this kind are called *hidden constraints*, because they are not specified to the simulator [39].

Finally, we have to define the upper bound and the lower bound for the design variable $\mathbf{v}$. The bounds are very important for the overall SBDO process because an eventually large value of the lower and upper bounds allows the optimizer to search on a larger space, making the optimization more challenging with the possibility to produce undesired geometries (i.e that don't satisfy the geometrical constraints). On the other side, a larger design space we could obtain more different configurations of the design variables with the chance to produce a larger improvement in the objective function.

2.2.2 Shape Parametrization: The Free Form Deformation

The second block of the SBDO framework is the shape parameterization. This comes after the optimization block because the shape parameterization method will receive the design variable vector $\mathbf{v}$ from the optimizer and will produce the relative shape modification to the geometry $\mathbf{x}(\mathbf{v})$. The choice of the shape parameterization

technique has a large impact on the practical implementation and the success of the optimization process.

Here, we show one of the most method for shape modifications, namely the Free Form Deformation (FFD). The idea is to embed an object within a trapezoidal (or other topology) lattice and modify the object within the trapezoid as the lattice is modified. A local coordinate system is assumed, with origin $\mathsf{x}_0 \in \mathcal{R}^3$ at one of the trapezoid vertices. Any point within the trapezoid has α, β, and γ coordinates such that

$$\mathsf{x} = \mathsf{x}_0 + \alpha\hat{\mathbf{T}}_1 + \beta\hat{\mathbf{T}}_2 + \gamma\hat{\mathbf{T}}_3 \tag{2.4}$$

with α, β, and γ bounded by $[0, 1]$ and given by

$$\begin{aligned}
\alpha &= \frac{\hat{\mathbf{T}}_2 \times \hat{\mathbf{T}}_3 \cdot (\mathsf{x} - \mathsf{x}_0)}{\hat{\mathbf{T}}_2 \times \hat{\mathbf{T}}_3 \cdot \hat{\mathbf{T}}_1}, \\
\beta &= \frac{\hat{\mathbf{T}}_1 \times \hat{\mathbf{T}}_3 \cdot (\mathsf{x} - \mathsf{x}_0)}{\hat{\mathbf{T}}_1 \times \hat{\mathbf{T}}_3 \cdot \hat{\mathbf{T}}_2}, \\
\gamma &= \frac{\hat{\mathbf{T}}_1 \times \hat{\mathbf{T}}_2 \cdot (\mathsf{x} - \mathsf{x}_0)}{\hat{\mathbf{T}}_1 \times \hat{\mathbf{T}}_2 \cdot \hat{\mathbf{T}}_3}
\end{aligned} \tag{2.5}$$

Control points (CPs) $\mathbf{c}_{ijk} \in \mathcal{R}^3$ are defined as lattice nodes. The number of CPs used in $\hat{\mathbf{T}}_1$, $\hat{\mathbf{T}}_2$, and $\hat{\mathbf{T}}_3$ directions are t_1, t_2, and t_3, respectively with a total number of CP's equals to $t_{\text{tot}} = t_1 + t_2 + t_3$. The coordinates of modified CPs depend on the imposed original-lattice nodes and in order to perform a modification to the geometry the coordinates of the CPs are perturbed by the relative *design variable* vector $\mathbf{v}_{ijk} \in \mathcal{R}^3$, as

$$c_{ijk}(\mathbf{v}_{ijk}) = \xi_0 + \frac{i}{t_1}\hat{\mathbf{T}}_1 + \frac{j}{t_2}\hat{\mathbf{T}}_2 + \frac{k}{t_3}\hat{\mathbf{T}}_3 + v_{ijk} \tag{2.6}$$

The shape modification is achieved by interpolating the CPs' modification over the embedding space. The interpolation can be performed using different polynomial bases. Herein, to generated a geometry a tensor product of trivariate Bernstein polynomial is used [115]

$$\begin{aligned}
\mathbf{x}(\mathbf{v}) &= \mathbf{g}_0 + \sum_{i=0}^{t_1}\sum_{j=0}^{t_2}\sum_{k=0}^{t_3} b_i(\alpha)b_j(\beta)b_k(\gamma)c_{ijk}(v_{ijk}) \\
&= \mathbf{g}_0 + \sum_{i=0}^{t_1}\sum_{j=0}^{t_2}\sum_{k=0}^{t_3} c_{ijk}(v_{ijk})B_{i,j,k}(\alpha, \beta, \gamma)
\end{aligned} \tag{2.7}$$

where $\mathbf{x}(\mathbf{v})$ is a vector containing the Cartesian coordinates of the displaced point, where the generic Bernstein basis polynomials is defined as

$$b_{v,r}(\chi) = \binom{r}{v}\chi^v(1-\chi)^{r-v} \tag{2.8}$$

The design variable vector $\mathbf{v}$ is then reshaped in M-dimensional vector where $M = 3t_{\text{tot}}$.

2.2.3 Physical Solver

In general a physical solver is a computer program that provide an approximation about the behavior of a physical system determined by its governing equations and boundary conditions. Computer simulations are crucial nowadays for carry out scientific research in many domains from fluid dynamics, material science, chemistry and biology. Especially when direct experimentation is too economically expensive, dangerous or even impossible to perform. In fluid dynamics the governing equations are given by the Navier-Stokes equations that we going to briefly introduce in this section. Without considering the energy in the system, the continuity equations and the Cauchy momentum equation are

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathsf{u}) = 0 \tag{2.9}$$

$$\rho \frac{D\mathsf{u}}{Dt} = \nabla \cdot \mathbf{T} + \rho \mathbf{f} \tag{2.10}$$

where is the velocity vector, ρ is the density, $\mathbf{T}$ the second order Cauchy stress tensor, and $\rho\mathbf{f}$ is the volume forces vector. With $\mathbf{T} = p\mathbf{I} + \tau$, for a Newtonian and incompressible fluid the Navier-Stokes equation are given by

$$\rho \frac{\partial \mathsf{u}}{\partial t} + \rho(\mathsf{u} \cdot \nabla)\mathsf{u} = \nabla \cdot [-p\mathbf{I} + \eta[\nabla \mathsf{u} + (\nabla \mathsf{u})^\top]] + \rho \mathbf{f} \tag{2.11}$$

assuming that the stress tensor is a linear function of the strain tensor, the fluid is isotropic, $\nabla \cdot \tau = 0$ for a fluid at rest and η and p is the dynamic viscosity of the fluid and the pressure respectively. The Navier-Stokes completely describe the dynamics of a fluid as the turbulence described by the nonlinear term. In many real world applications, the turbulence effects must to be considered in order to provide an accurate description of the physical process. Still nowadays, the computational effort required for the direct numerical simulation (DNS) of the Navier-Stokes equations is intractable. Most of time a simplified numerical model is considered as the Large Eddy Simulation (LES), Reynolds Averaged Navier-Stokes Equations (RANS).

2.3 Design Space Dimensionality Reduction for SBDO

The curse of dimensionality is one of the major drawback when global optimization algorithms are employed in order to find an optimal solution. In the following sections we show how to reduce the dimensionality of the design vector $\mathbf{v}$ by *learning* a new parameterization for the shape modification. More precisely the objective that we want like to achieve is to perform the optimization respect a new design variable vector $\mathbf{z} \in \mathcal{R}^K$ with $K < M$ in order to improve the convergence speed of the global optimization routine to an optimal solution.

The process is highlighted in Fig. 2.3. The first phase is to generate the dataset $\mathbf{X}$ and this is done by randomly sampling the design variable vector $\mathbf{v}$ from a uniform distribution. As support for the uniform distribution we use the upper bounds and lower bounds for each design variable component. The second phase is to apply a dimensionality reduction model which learns the new parameterization given by the new design variable vector $\mathbf{z}$ and a matrix $\mathbf{U}$. Here the matrix $\mathbf{U}$ is responsible to

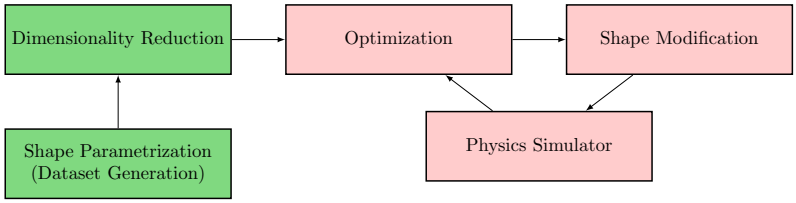


Figure 2.2. SBDO framework with design-space dimensionality reduction phase.

transform (or decode) the design variable $\mathbf{z}$ in a new geometry $\mathbf{x}(\mathbf{z})$. The matrix $\mathbf{U}$ replaces the shape parameterization method (e.g the FFD method) during the SBDO loop.

2.3.1 Dataset Generation

The first step in the DR-SBDO is to generate the dataset that will be used in the next block to train the dimensionality reduction model. This can be simply achieved by sampling from a uniform distribution the full dimensional design variable vector $v_m \sim \mathcal{U}(v_m^{\text{lb}}, v_m^{\text{ub}})\, \forall m = 1, \ldots, M$ and using the sampled vector as input for Eq. 2.7 to produce a random geometry. If we repeat the process N times we can collect all the geometries inside a dataset $\mathbf{X}$ of size $(N \times D)$, where D is the number of nodes L where the geometry is discretized times the number of Cartesian coordinate $(\mathsf{x}, \mathsf{y}, \mathsf{z})$ considered in the application. In this step, is crucial that the designer carefully selects the value for the upper and lower bounds for the design variables. The first reason is that a restricted range for the design variables will not allow diversity but redundancy (i.e low variance) in the generated dataset. A large range for the design variables, produces many undesired geometries as many of them will not satisfy the geometrical constraints. Once the dataset is generated we are ready to perform the next step as described in the next section.

2.3.2 The Principal Component Analysis

Principal Component Analysis (PCA) becomes during the last decades one of the most famous statistical tool for feature extraction and dimensionality reduction. It appeared for the first time in a paper [93] published by Karl Pearson in 1901 and then independently developed from a different point of view by Harold Hotelling in early '30's [64]. PCA finds a new optimal basis of dimension K such that the variance of the projected points is maximized and the mean squared error between the original data and their projection is minimized. The first step in the PCA is the computation of the sample mean vector $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$ and the computation of the sample covariance matrix $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$. The second step is to solve an eigendecomposition problem respect to the matrix $\mathbf{S}$

$$\mathbf{S}\mathbf{u} = \lambda \mathbf{u} \tag{2.12}$$

of size $(D \times D)$. To reduce the dimensionality of our dataset we should choose a subset of K eigenvectors which corresponds to the top-K largest variance eigenvalues. Then we can project a data vector $\mathbf{x}$ into the subspace defined by top-K

orthonormal eigenvectors of the covariance matrix $\mathbf{U}$ (of size $D \times K$) also called *principal components*

$$\mathbf{z}_n = \mathbf{U}^\top(\mathbf{x}_n - \bar{\mathbf{x}}) \tag{2.13}$$

the matrix $\mathbf{z}_n$ of size K represents the reduced representation of the original data $\mathbf{x}_n$. Usually the random variable $\mathbf{z}$ is called *latent variable* because they are not directly observed but they explain or extract some useful patterns of the observed data $\mathbf{x}$. In case is needed, the reconstruction of the relative data point can be obtained by projecting back to data space $\mathcal{R}^D$ as following

$$\tilde{\mathbf{x}}_n = \mathbf{U}\mathbf{z}_n + \bar{\mathbf{x}} = \mathbf{U}\mathbf{U}^\top\mathbf{x}_n + \bar{\mathbf{x}} = \mathbf{P}\mathbf{x}_n + \bar{\mathbf{x}} \tag{2.14}$$

where $\mathbf{P} = \mathbf{U}\mathbf{U}^\top$ is the symmetric orthogonal projection matrix for the subspace spanned by the first top-K eigenvectors $\mathbf{U}$. Practical implementations of the PCA perform the Singular Value Decomposition [53] of the data matrix $\mathbf{X} = \mathbf{V}\boldsymbol{\Xi}\mathbf{U}^\top$, where $\mathbf{V}$, $\boldsymbol{\Xi}$ and $\mathbf{U}$ are $(N \times K)$, $(K \times K)$ and $(K \times D)$ respectively, with $K \leq \min(N, D)$. In this case $\mathbf{X}$ must be centered before the SVD computation to obtain the correct results. The sample covariance matrix in this setting is given by $\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$ and consequently

$$\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X} = \frac{1}{N}\mathbf{U}\boldsymbol{\Xi}\mathbf{V}^\top\mathbf{V}\boldsymbol{\Xi}\mathbf{U}^\top = \frac{1}{N}\mathbf{U}\boldsymbol{\Xi}^2\mathbf{U}^\top = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top \tag{2.15}$$

the eigenvalues of the covariance matrix are given by $\lambda_i = \frac{\xi_i^2}{N}$. One important property in PCA is that the projection of the data matrix $\mathbf{X}$ in the latent space given by $\mathbf{Z}$ are uncorrelated

Lemma 2.3.1. *Suppose that the data matrix* $\mathbf{X}$ *is already centered and its projection in the subspace spanned by the first* k *principal components is given by* $\mathbf{Z} = \mathbf{X}\mathbf{U}$. *Then the correlation between* $\mathbf{z}_i$ *and* $\mathbf{z}_j$ *for* $i, j = 1, \ldots, K$ *with* $i \neq j$ *is zero.*

Proof. By computing the covariance of $\mathbf{Z}$ and using the SVD decomposition

$$\frac{1}{N}\mathbf{Z}^\top\mathbf{Z} = \frac{1}{N}\mathbf{U}^\top\mathbf{X}^\top\mathbf{X}\mathbf{U} = \frac{1}{N}\mathbf{U}^\top\mathbf{U}\boldsymbol{\Xi}\mathbf{U}^\top\mathbf{U} = \boldsymbol{\Lambda} \tag{2.16}$$

we obtain that the covariance of $\mathbf{Z}$ is diagonal and consequently the columns of $\mathbf{Z}$ are uncorrelated and the variance given by the eigenvalues of $\mathbf{S}$. □

Another important property of PCA is that using as new basis the principal components for the reduced subspace of $\mathcal{R}^K$, the variance along the projections $\mathbf{z}$ is maximized and the euclidean distance from the reconstructed data $\tilde{\mathbf{x}}$ and the original representations $\mathbf{x}$ is minimized

Lemma 2.3.2. *Given the eigenvectors of the covariance matrix* $\mathbf{U}$, *then the squared euclidean distance between* $\tilde{\mathbf{x}}$ *and the original data* $\mathbf{x}$ *is minimized if and only if the variance of the projections* $\mathbf{z}$ *is maximized.*

Proof. Writing the sum of the euclidean distance between $\mathbf{x}$ and $\tilde{\mathbf{x}}$

$$\begin{aligned}
\sum_{n=1}^{N} ||\mathbf{x}_n - \tilde{\mathbf{x}}_n||^2 &= \sum_{i=1}^{N} ||\mathbf{x}_n - \mathbf{U}\mathbf{U}^\top \mathbf{x}_n||^2 \\
&= \sum_{i=1}^{N} ||\mathbf{x}_n||^2 - 2\mathbf{x}_n^\top \mathbf{U}\mathbf{U}^\top \mathbf{x}_n + (\mathbf{U}\mathbf{U}^\top \mathbf{x}_n)^\top (\mathbf{U}\mathbf{U}^\top \mathbf{x}_n) \\
&= \sum_{i=1}^{N} ||\mathbf{x}_n||^2 - \sum_{i=1}^{N} \mathbf{x}_n^\top \mathbf{U}\mathbf{U}^\top \mathbf{x}_n \\
&= \sum_{i=1}^{N} ||\mathbf{x}_n||^2 - \sum_{i=1}^{N} \mathbf{z}_n^\top \mathbf{z}_n \\
&= N(\mathrm{Tr}(\mathbf{S}) - \mathrm{Tr}(\mathbf{\Lambda}))
\end{aligned}$$

Thus the in order to minimize the euclidean distance the variance of the projection must to be maximized. □

Finally a simple and widely used criterion for choose the number components K to retain is given by computing the fraction explained variance respect to the total variance given by $\frac{\sum_{k=1}^{K} \lambda_k}{\mathrm{Tr}\,(S)}$ and fix K to the desired level.

2.3.3 Optimization in the Latent Space

Once we trained the PCA model respect to the dataset $\mathbf{X}$, we can formulate the new optimization model. Given the latent variable $\mathbf{z}$

$$\min_{\mathbf{z}} \quad f(\mathbf{x}(\mathbf{z})) \tag{2.17}$$

$$g_j(\mathbf{x}(\mathbf{z})) \leq a_j \qquad \forall j = 1, \ldots, J \tag{2.18}$$

$$z_k^{\mathrm{lb}} \leq z_k \leq z_k^{\mathrm{ub}} \qquad \forall k = 1, \ldots, K \tag{2.19}$$

where the modification of the geometry $\mathbf{x}$ is performed by the variable $\mathbf{z} \in \mathcal{R}^K$. Also the bounds for the latent variable are computed taking the maximum and the minimum of each column component z_k of $\mathbf{Z}$. Now the optimizer performs its update in the latent space respect to the variable $\mathbf{z}$ and the dimensionality of this subspace is smaller respect to the full dimensionality space defined by $\mathbf{v} \in \mathcal{R}^M$. This fact should increase the capability and the performance of the global optimization algorithm to find the basin of attraction of the global minimum more efficiently.

2.3.4 Decoding from the Latent Space

Once the optimizer perform a new iteration on the reduced dimensionality space it returns as output a new design variable $\mathbf{z}$, that hopefully, will produce a decreased value in the objective function $f(\mathbf{x}(\mathbf{z}))$. To obtain the objective function value the design variable vector $\mathbf{z}$ must be projected back in the data space to obtain its relative modified geometry $\mathbf{x}(\mathbf{z})$ as input for the physical solver. This can be easily achieved by performing a matrix multiplication with the principal components of the data $\mathbf{U}$, namely $\mathbf{x} = \mathbf{U}\mathbf{z} + \bar{\mathbf{x}}$ where to maintain the notation uncluttered we suppose

that $\mathbf{x}(\mathbf{z}) = \mathbf{x}$. The eigenvectors $\mathbf{U}$ represents spatial geometrical components and are responsible to decode back the design variable $\mathbf{z}$ in the space $\mathcal{R}^D$. Once we obtained the geometry from the latent space $\mathbf{x}$, this will entry in the physical solver that computes the objective function as shown in Fig. 2.3.

2.4 Probabilistic Linear Latent Variable Models

In this section we will introduce some important modifications in the framework described before. The SBDO framework with a dimensionality reduction phase performed before the optimization routine allows the optimizer to iterate in subspace of lower dimensionality of $\mathcal{R}^K$ respect to the full dimensionality space of $\mathcal{R}^M$ with $K < M$. An important property that could be highly desiderable when performing the optimization in the latent space is that the shape modification should be *invariant* respect the full dimensionality design space $\mathcal{R}^M$. This means that the design variable vector $\mathbf{z}$ should not produce geometries that the original full dimensional design variable vector $\mathbf{v}$ is not able to compute. This is possible in case the latent space define a subspace that is larger respect to the full dimensionality space. Suppose for a moment that the data $\mathbf{x} \in \mathbf{X}$ is confined inside an hypersphere. The bounds constraint of the optimization model in the latent space in Eq. 2.19 defines an hyperrectangle in data space. Consequently, a decoded geometry in the data space $\mathbf{R}^D$ could be arbitrarily far from the center of the hypersphere. Also, the performance of the optimizer could be weaken because must to explore in a larger space. In the next sections we will describe more in details the problematics and how to solve them using the new framework for shape optimization, precisely we will discuss

- In section 2.4.1 we show that in case the design variables are sampled from a uniform distribution, the geometries produced with the shape parameterization method (e.g. FFD) produce a design space that approximately follows a Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$ as a direct application of the Central Limit Theorem (CLT).

- In section 2.4.2 and 2.4.3 we introduce two probabilistic latent variable models as Probabilistic PCA and Factor Analysis for dimensionality reduction and density estimation for the new SBDO framework. In particular for the density estimation we are interested on the estimate of the inverse of the covariance matrix in order to define the uncertainty.

- In section 2.4.4 we propose a new optimization model which take into account the uncertainty in terms of Mahalanobis distance. Also, we show the effect when the uncertainty is not considered, analyzing the geometrical properties of our new proposed methodology.

2.4.1 Statistical Properties of the Shape Parametrization Method

As we already seen the shape parameterization method as the FFD is used in the DR-SBDO for generate the dataset composed by N geometries which design variables are sampled from a uniform distribution $v_m \sim \mathcal{U}(v_m^{\text{lb}}, v_m^{\text{ub}}), \forall m = 1, \ldots, M$.

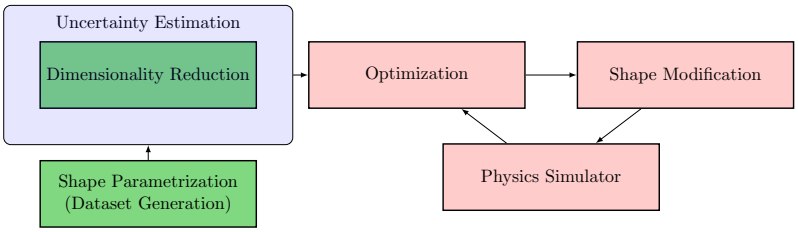


Figure 2.3. SBDO framework with design-space dimensionality reduction and uncertainty estimation phase.

As shown in section 2.3.1, the sampled design variable will produce a uniform random perturbation in the positions of the control point so that the interpolant function will produce a modified geometry $\mathbf{x}$. Looking at Eq. 2.7 the shape parameterization method performs a linear combination of a fixed function of the cartesian coordinates $r(\mathsf{x}, \mathsf{y}, \mathsf{z})$ and a coefficient that represent the design variable responsible for the random perturbation of the geometry around these cartesian coordinates. Consequently, they are a sum of random vectors $\mathbf{h} \in \mathcal{R}^D$

$$\mathbf{x}(\mathbf{v}) = \mathbf{g}_0 + \sum_{m=1}^{M} r(\mathsf{x}, \mathsf{y}, \mathsf{z}) v_m = \mathbf{g}_0 + \sum_{m=1}^{M} \mathbf{h}(r, v_m) \tag{2.20}$$

where $\mathbf{x}(\mathbf{v})$ is a vector containing the Cartesian coordinates of the geometry. Could be interesting to exploit some statistical properties of $\mathbf{x}(\mathbf{v})$. The most interesting one would be understand from which probability distribution the data $\mathbf{x}(\mathbf{v})$ is generated from. The central limit theorem (CLT) [136] shows that under some certain conditions, the probability distribution of the sum of a large number independently and identical distributed random variables converge to a Gaussian distribution. More formally

Theorem 2.4.1. *For a sequence of D dimensional random vectors $\mathbf{h}(r, v_m)$ with finite mean and covariance $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ respectively, we have that*

$$\sqrt{M}\left(\frac{1}{M}\sum_{m=1}^{M} \mathbf{h}(r, v_m) - \boldsymbol{\mu}\right) \xrightarrow{d} \mathcal{N}(0, \boldsymbol{\Sigma}) \tag{2.21}$$

as $M \to \infty$.

which implies that $\sum_{m=1}^{M} \mathbf{h}(r, v_m) \xrightarrow{d} \mathcal{N}(M\boldsymbol{\mu}, M\boldsymbol{\Sigma})$ for $M \to \infty$. For the proof see for example [142]. When the random variables $\mathbf{h}$ are generated by a uniform distribution, their sum converges to a Gaussian distribution very fast (i.e for $M << \infty$). This is an important result because if the probability distribution of data $\mathbf{X}$ is approximately Gaussian we can use this information to force the optimizer to 'stay' in the region confined by the hyperellipsoid defined by the Gaussian distribution. A final observation and important observation is that the geometries $\mathbf{x} \in \mathcal{R}^D$ are produced from a design variable vector of much lower dimensionality $\mathbf{v} \in \mathcal{R}^M$. Consequently, we can expect that the degrees of freedom of the data $\mathbf{x}$ is not D but much lower. We can imagine in this case that the data $\mathbf{x}$ lives in lower dimensional

linear manifold. This means that the sample covariance matrix $\mathbf{S}$ has many zero eigenvalues and then not invertible. In the next section we describe two probabilistic latent variables models that we can use for dimensionality reduction and density estimation. From these two models, we can obtain the inverse of the covariance matrix in order to compute the uncertainty for the new SBDO framework.

2.4.2 Factor Analysis

Factor Analysis [6] is one of the most famous probabilistic latent variable model that during its long history has been applied in many different fields. We start from the assumption that the variable $\mathbf{x}$ can be written as a linear combination of a latent variable $\mathbf{z}$

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{2.22}$$

and a *factor loading* matrix $\mathbf{W}$ of size $(D \times K)$. The latent variable is generated by a standard Gaussian distribution $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ like the noise term $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Psi})$, independent from $\mathbf{z}$. We can write that the conditional distribution is given by

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{2.23}$$

a Gaussian distribution with conditional mean $\mathbb{E}[\mathbf{x}|\mathbf{z}] = \mathbf{W}\mathbf{z} + \boldsymbol{\mu}$ and a diagonal conditional covariance $(D \times D)$ matrix $\boldsymbol{\Psi}$. The FA model is described by the parameters $\boldsymbol{\Theta} = \{\mathbf{W}, \boldsymbol{\Psi}\}$ for a total of $(D \times K + D)$ parameters. We find the expression for the marginal distribution $p(\mathbf{x})$ solving the following integral

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z}) p(\mathbf{z})\, d\mathbf{z} \tag{2.24}$$

where Eq. 2.24 can be computed in closed form because it represents a convolution of two Gaussian distributions. The marginal is again Gaussian, $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$ with the covariance matrix equal to

$$\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \boldsymbol{\Psi} \tag{2.25}$$

The diagonal elements of $\mathbf{C}$ are given by a sum of two terms. The first term, $||\mathbf{w}_i||^2$, $\forall i = 1, \ldots, D$ is called *communality* because it represents the variance explained by the K factors $\mathbf{W}$ respect to the feature x_i. The second term, is the variance relative to the feature x_i that is not explained by the factors and is called *uniqueness*. Given those results we can write down the expression for the posterior distribution $p(\mathbf{z}|\mathbf{x})$

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{W}^\top \mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu}), \mathbf{G}) \tag{2.26}$$

with with $\mathbf{G} = (\mathbf{W}^\top \boldsymbol{\Psi} \mathbf{W} + \mathbf{I})^{-1}$ so that the posterior mean is given by

$$\mathbb{E}[\mathbf{z}|\mathbf{x}] = \mathbf{W}^\top \mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \tag{2.27}$$

that is a linear function of $\mathbf{x}$ while the posterior covariance $\mathbf{G}$ is independent from $\mathbf{x}$. We can find the set of parameters $\boldsymbol{\Theta} = \{\mathbf{W}, \boldsymbol{\Psi}\}$ via the classical Maximum Likelihood Estimation (MLE) respect to the marginal likelihood using the information in our observed data $\mathbf{x}_n \in \mathbf{X}$. Unfortunately, for the FA model a closed form solution of

the marginal likelihood is not available and then numerical optimization methods should be employed in order to find the parameters. A viable and efficient option for find parameters in latent variable models is using the Expectation Maximization (EM) algorithm [31]. The EM algorithm tries to maximize the expectation of the joint log-likelihood $\ln p(\mathbf{X}, \mathbf{Z})$ respect to the posterior $p(\mathbf{z}|\mathbf{x})$. We can compute

$$\ln p(\mathbf{X}, \mathbf{Z}) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{z}_n) + \ln p(\mathbf{z}_n) \tag{2.28}$$

then taking the expectation respect to the posterior $p(\mathbf{z}|\mathbf{x})$ we obtain

$$\begin{aligned}\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\Theta})] &= \sum_{n=1}^{N} \mathbb{E}_{\mathbf{z}|\mathbf{x}}[\ln p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\Theta}) + \ln p(\mathbf{z}_n)] \\ &= \sum_{n=1}^{N} \mathbb{E}_{\mathbf{z}|\mathbf{x}}\Big[-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Psi}| - \\ &\qquad \frac{1}{2}(\mathbf{x}_n - \mathbf{W}\mathbf{z}_n - \boldsymbol{\mu})^\top \boldsymbol{\Psi}^{-1}(\mathbf{x}_n - \mathbf{W}\mathbf{z}_n - \boldsymbol{\mu})\Big]\end{aligned} \tag{2.29}$$

where we omitted the prior distribution since it does not depend on $\boldsymbol{\Theta}$. In the EM algorithm the following two steps are repeated until convergence

- Expectation step (E-step) where we compute the sufficient statistics of the posterior distribution respect to the old parameters $p(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\Theta}_{\text{old}})$

$$\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n] = \mathbf{G}\mathbf{W}^\top \boldsymbol{\Psi}^{-1}(\mathbf{x}_n - \bar{\mathbf{x}}) \tag{2.30}$$

$$\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n \mathbf{z}_n^\top] = \mathbf{G} + \mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n]\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n]^\top \tag{2.31}$$

- Then given the statistics from the E-step, in the new Maximization step (M-step) we evaluate the parameters maximizing $\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\Theta}_{\text{old}})]$ setting its derivatives respect to the parameters $\boldsymbol{\Theta}$ to zero

$$\mathbf{W}_{\text{new}} = \Big[\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n]^\top\Big]\Big[\sum_{n=1}^{N}\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n\mathbf{z}_n^\top]\Big]^{-1} \tag{2.32}$$

$$\boldsymbol{\Psi}_{\text{new}} = \text{diag}\Big\{\mathbf{S} - \mathbf{W}_{\text{new}}\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n](\mathbf{x}_n - \bar{\mathbf{x}})^\top\Big\} \tag{2.33}$$

This iterative procedure described above, efficiently produces at the end of the iterations, a stationary solution.

Suppose now that $\mathbf{R}$ is a $(D \times D)$ orthogonal matrix, if we define the new rotated factor loadings $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$ is possible to notice that the marginal covariance 2.25 with this new reparametrization is given by

$$\mathbf{C} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^\top + \boldsymbol{\Psi} = \mathbf{W}\mathbf{R}\mathbf{R}^\top\mathbf{W} + \boldsymbol{\Psi} = \mathbf{W}\mathbf{W}^\top + \boldsymbol{\Psi} \tag{2.34}$$

which means that performing a rotation of the latent space achieves the same value for the marginal distribution. Consequently, the matrix $\mathbf{W}$ is not uniquely identifiable.

In the next section we are going to discuss a special case of FA where the conditional covariance matrix $\boldsymbol{\Psi}$ is restricted to be isotropic. This will allow to compute the MLE solution of the marginal distribution in closed form.

2.4.3 Probabilistic Principal Component Analysis

Probabilistic PCA (PPCA) introduced independently by [138] and [106] is a probabilistic formulation of classical PCA which assumes a linear relationship between the observed variable $\mathbf{x}$ and the K-dimensional latent variable $\mathbf{z}$ plus a Gaussian noise term $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \epsilon \tag{2.35}$$

where $\mathbf{W}$ is a matrix of size $(D \times K)$ and $\boldsymbol{\mu}$ is a D-dimensional vector mean. Assuming a zero mean and uncorrelated Gaussian latent variable $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, the value of $\mathbf{x}$ conditioned on $\mathbf{z}$ is given by

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \tag{2.36}$$

where the conditional mean equals to $\mathbb{E}[\mathbf{x}|\mathbf{z}] = \mathbf{W}\mathbf{z} + \boldsymbol{\mu}$ and the conditional isotropic covariance $(D \times D)$ matrix given by $\sigma^2\mathbf{I}$. The PPCA model is described by a set of parameters $\boldsymbol{\Theta} = \{\mathbf{W}, \sigma^2\mathbf{I}\}$ for a total of $(D \times K + 1)$ parameters. We can compute the marginal distribution $p(\mathbf{x})$ solving the following integral

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{z} \tag{2.37}$$

that can be compute in closed form since it represents again a convolution of two Gaussian distributions with $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$ and a covariance matrix given by

$$\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I} \tag{2.38}$$

The marginal distribution depends from the parameters $\boldsymbol{\Theta}$ that can be determined maximizing the marginal likelihood

$$\begin{aligned} \ln p(\mathbf{X}|\boldsymbol{\Theta}) &= \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) \\ &= -\frac{ND}{2} - \frac{N}{2}\ln(2\pi) - \frac{N}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^\top \mathbf{C}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}) \end{aligned} \tag{2.39}$$

Setting the derivatives to zero, the likelihood is maximized at

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{2.40}$$

$$\sigma^2 = \frac{1}{D-K}\sum_{i=K+1}^{D}\lambda_i \tag{2.41}$$

$$\mathbf{W} = \mathbf{U}(\boldsymbol{\Lambda} - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \tag{2.42}$$

The marginal likelihood in Eq. 2.39 is a non convex function but all stable stationary points are global minima [138].

The maximum likelihood solution is obtained in closed form, where $\boldsymbol{\mu}$ is the sample mean, $\boldsymbol{\Lambda}$ is a $(K \times K)$ diagonal matrix composed by the top-K largest eigenvalues of the sample covariance matrix $\mathbf{S}$, $\mathbf{U}$ the relative PCA eigenvectors and

σ^2 is the expected value of the residual variance of the discarded $N - K$ principal components. The orthogonal $(K \times K)$ matrix $\mathbf{R}$ can be set to $\mathbf{I}$. In this case, the matrix $\mathbf{W}$ is composed by the principal components $\mathbf{U}$ scaled by a factor of $\sqrt{\lambda_i - \sigma^2}$. In case a numerical optimization algorithm is used for find the maximum likelihood solution, the matrix $\mathbf{R}$ could be arbitrary and the matrix $\mathbf{W}$ not orthogonal at the optimal solution.

To define a projection of a point $\mathbf{x}$ in the latent space we can compute the posterior distribution of $p(\mathbf{z}|\mathbf{x})$ using the Bayes theorem as

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{M}^{-1}\mathbf{W}^\top(\mathbf{x} - \boldsymbol{\mu}), \sigma^2\mathbf{M}^{-1}) \tag{2.43}$$

with $\mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I}$. The latent representation of a data point $\mathbf{x}$ in the latent space is given by the posterior mean

$$\mathbb{E}[\mathbf{z}|\mathbf{x}] = \mathbf{M}^{-1}\mathbf{W}^\top(\mathbf{x} - \boldsymbol{\mu}) \tag{2.44}$$

and can be reconstructed in data space with $\mathbf{W}\mathbb{E}[\mathbf{z}|\mathbf{x}] + \boldsymbol{\mu}$. Differently from PCA, in PPCA and FA we are not performing an orthogonal projection of the data in the latent space. In fact, if we take the limit $\sigma^2 \to 0$ with $\mathbf{W}$ given by the MLE solution we recover the classical PCA, $(\mathbf{W}^\top\mathbf{W})^{-1}\mathbf{W}^\top(\mathbf{x} - \boldsymbol{\mu})$. But for $\sigma^2 \to 0$ the posterior covariance is zero and the density becomes singular and then not defined. Even if an exact closed form solution of the likelihood is provided, could be advantageous to use an iterative procedure to compute the parameters instead to perform the eigenvalue decomposition of the $(D \times D)$ sample covariance matrix $\mathbf{S}$. This can be expensive and not viable option in case of very large D. As for the FA model the EM algorithm involves in the maximization of the expectation of the complete data log-likelihood $\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\Theta})$ respect to the posterior $p(\mathbf{z}|\mathbf{x})$. In case of the PPCA model this corresponds to compute

$$\begin{aligned}
\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\Theta})] &= \sum_{n=1}^{N} \mathbb{E}_{\mathbf{z}|\mathbf{x}}[\ln p(\mathbf{x}|\mathbf{z}, \boldsymbol{\Theta}) + \ln p(\mathbf{z})] \\
&= \sum_{n=1}^{N} \mathbb{E}_{\mathbf{z}|\mathbf{x}}\Big[-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln|\sigma^2\mathbf{I}| - \\
&\qquad \frac{1}{2}(\mathbf{x}_n - \mathbf{W}\mathbf{z}_n - \boldsymbol{\mu})^\top\sigma^2\mathbf{I}^{-1}(\mathbf{x}_n - \mathbf{W}\mathbf{z}_n - \boldsymbol{\mu})\Big]
\end{aligned} \tag{2.45}$$

where we omitted the prior distribution since it does not depend on $\boldsymbol{\Theta}$. Setting $\boldsymbol{\mu}$ as the sample mean, in the EM algorithm the following two steps are repeated until convergence

- Expectation step (E-step) where we compute the statistics of the posterior distribution given the old parameters $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\Theta}_{\text{old}})$:

$$\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}(\mathbf{x} - \boldsymbol{\mu}) \tag{2.46}$$

$$\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n\mathbf{z}_i^\top] = \sigma^2\mathbf{M}^{-1} + \mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n]\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n]^\top \tag{2.47}$$

  Then given the new value of the new Maximization step (M-step) based on the posterior distribution we evaluate the parameters by maximizing

$\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\ln p(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta}_{\text{old}})]$ setting its derivatives respect to the parameters $\mathbf{\Theta}$ to zero

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n]^\top\right]\left[\sum_{n=1}^{N}\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n\mathbf{z}_n^\top]\right]^{-1} \tag{2.48}$$

$$\sigma^2_{\text{new}} = \frac{1}{ND}\sum_{n=1}^{N}(||\mathbf{x}_n - \boldsymbol{\mu}||^2) - 2\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n]^\top\mathbf{W}_{\text{new}}^\top(\mathbf{x}_n - \boldsymbol{\mu}) \tag{2.49}$$

$$+ \text{Tr}(\mathbb{E}_{\mathbf{z}|\mathbf{x}}[\mathbf{z}_n\mathbf{z}_n]^\top\mathbf{W}_{\text{new}}^\top\mathbf{W}_{\text{new}}) \tag{2.50}$$

There is a great computational advantage in computing the model parameters $\mathbf{\Theta}$ with EM algorithm especially when $K << D$ and when the dimensionality of the observed data D and the number of observation N are large. This because the covariance is never computed explicitly which has complexity $\mathcal{O}(ND^2)$ and its eigenvalue decomposition with complexity $\mathcal{O}(D^3)$. The snapshot method [130] for the computing the eigenvalues of the covariance is $\mathcal{O}(N^3)$ which scales poorly in case of large datasets.

A remark regards in the computation of inverse of the covariance matrix $\mathbf{C}$. Instead of directly take the inverse of $\mathbf{C}$, we can use the Woodbury matrix inversion formula [147] obtaining, $\mathbf{C}^{-1} = \sigma^{-2}\mathbf{I} - \sigma^{-2}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^\top$. Using the previous transformation we only need to compute the inverse of $\mathbf{M}$ which is $(K \times K)$ rather than find directly the inverse of $\mathbf{C}$ with cost $\mathcal{O}(D^3)$; the same transformation can be applied to the FA model.

Finally, the PPCA as the FA, is invariant to rotations in latent space which means that there exist a set of matrices $\mathbf{W}$ giving the same marginal distribution.

2.4.4 Exploiting the Uncertainty in the Optimization Model

Once we trained our model (either a PPCA or FA) on the data matrix $\mathbf{X}$ we can start the SBDO process. In this case the optimization problem is given by

$$\min_{\mathbf{z}} \quad f(\mathbf{x}(\mathbf{z})) \tag{2.51}$$

$$\text{subject to:} \quad g_j(\mathbf{x}(\mathbf{z})) \leq a_j \qquad \forall j = 1, \ldots, J \tag{2.52}$$

$$\varphi(\mathbf{x}(\mathbf{z})) \leq \varphi_{\max} \tag{2.53}$$

$$z_k^{\text{lb}} \leq z_k \leq z_k^{\text{ub}} \qquad \forall k = 1 \ldots K \tag{2.54}$$

where we added a new constraint in Eq. 2.53. The term $\varphi(\mathbf{x}(\mathbf{z}))$ represents the uncertainty associated to the geometry $\mathbf{x}(\mathbf{z})$. Given the inverse of the covariance matrix $\mathbf{C}^{-1}$ this is given by (with $\mathbf{x}(\mathbf{z}) = \mathbf{x}$)

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C}) = \frac{1}{2\pi^{D/2}|\mathbf{C}^{1/2}|}\exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top\mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\} \tag{2.55}$$

the marginal distribution estimated through the probabilistic latent variable models, FA and PPCA. A more interpretable metric is given by the exponent of 2.55, namely the squared Mahalanobis distance $d_M^2(\mathbf{x})$. The Mahalanobis distance [85] is defined a as follow

$$d_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top\mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu})} \tag{2.56}$$

The Mahalanobis distance introduced by P.C Mahalanobis in 1936, it's an extension of the Euclidean distance that takes into account the correlation between vectors since for $\mathbf{C} = \mathbf{I}$ Eq. 2.56 reduces to the Euclidean distance. The scalar $\varphi_{\max}$ represents a threshold for the maximum level of uncertainty that the generated shape from the optimizer cannot exceed. The value of the scalar $\varphi_{\max}$ is computed respect to the reconstructed dataset $\tilde{\mathbf{X}}$. For each data point we encode it in the latent space using Eq. 2.27 or Eq. 2.44 and then projected back in data space using the conditional mean in 2.23 or Eq. 2.36 respectively for FA and PPCA. Successively, we can evaluate the uncertainty (for example using Eq. 2.56) respect to every $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ and fix $\varphi_{\max}$ to the maximum value obtained.

Could be also interesting at this point to explore what kind of distribution can have in general the random variable defined by the squared Mahalanobis distance $d_M^2(\mathbf{x})$. Now we will show that under some condition the squared Mahalanobis distance follows a Chi-Squared distribution $d_M^2(\mathbf{x}) \sim \chi_D^2$.

Lemma 2.4.2. *Given a D-dimensional random variable vector* $\mathbf{x}$*, then if* $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *then the distribution of the squared Mahalanobis distance follows a Chi-Squared distribution* $d_M^2(\mathbf{x}) \sim \chi_d^2$*.*

Proof. For the random variable $\mathbf{x}$ the Gaussian distribution is given by

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{2\pi^{D/2}|\boldsymbol{\Sigma}^{1/2}|} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{2.57}$$

consequently the functional dependence respect to $\mathbf{x}$ is given only respect to the squared Mahalanobis distance $d_M^2(\mathbf{x})$ inside the exponent of the Gaussian distribution. Using the eigenvalue decomposition for the covariance matrix $\boldsymbol{\Sigma}\mathbf{u}_i = \mathbf{u}_i\lambda_i$, $\forall i = 1, \ldots, D$, its inverse can be written as follows

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^\top \tag{2.58}$$

and the squared Mahalanobis distance becomes

$$d_M^2(\mathbf{x}) = \sum_{i=1}^{D} \frac{1}{\lambda_i} (\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{u}_i \mathbf{u}_i^\top (\mathbf{x}-\boldsymbol{\mu}) = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i} \tag{2.59}$$

where $y_i = \mathbf{u}_i^\top(\mathbf{x}-\boldsymbol{\mu})$ represents the i-th component of the vector $\mathbf{y}$ in the new coordinate system defined by the eigenvectors $\mathbf{U}$. The random variable $\mathbf{x}$ is Gaussian and its the affine transformation $\mathbf{y}$ is again Gaussian, consequently since for definition the sum of squared value of normal random variables follow a Chi squared distribution then $d_M^2(\mathbf{x}) \sim \chi_d^2$. □

This an important result in our setting because we could use that as an additional proof that random geometries $\mathbf{x}_n \in \mathbf{X}$, $\forall n = 1, \ldots, N$ produced by the shape parameterization are Gaussian distributed. Finally, we'd like to conclude this section what is the effect in case the constraint in Eq. 2.53 inside the SBDO process. The constraints in Eq. 2.53 and 2.54 define an hyperellipsoid and a hyperrectangle in the data space $\mathcal{R}^D$. To simplify the analysis, we can consider instead of an

hyperellipsoid for constraint Eq. 2.53 an hypersphere which corresponds to an isotropic covariance matrix for the Gaussian distribution. Also, let's consider an hypercube for constraint 2.54 always for make the analysis more general. We can derive an expression for the volume of the hypersphere and for the hypercube, using simple tools from geometry of $\mathcal{R}^D$ [71]. The volume of the hypersphere of radius r is given by $V_D = \pi^{D/2} r^D (\Gamma(\frac{D}{2}+1))^{-1}$ (where $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt$). The volume of the hypercube of length $l = 2r$ is given by $H(D) = l^D = 2r^D$. The asymptotic behavior for the ratio of the volume of the hypersphere inscribed in an hypercube is given by

$$\lim_{D \to +\infty} \frac{V_D}{H_D} = \frac{\pi^{D/2} r^D}{2r^D \Gamma(\frac{D}{2}+1)} = 0 \tag{2.60}$$

with

$$\Gamma\left(\frac{D}{2}+1\right) = \begin{cases} (\frac{D}{2})! & \text{if } D \text{ is even} \\ \sqrt{\pi}(\frac{D!!}{2^{d+1/2}}) & \text{if } D \text{ is odd} \end{cases} \tag{2.61}$$

for $D = 2$ the limit in Eq. 2.60 is equal to 78.5%, meaning that the circle covers the 78.5% of the space defined by the hypercube. The limit converges very fast to zero. For $D = 6$, the hypersphere covers only the $\simeq 8\%$ of the entire hypercube! This shows that as the dimensionality increase, the majority of the volume of the hypercube is concentrated in its 2^D corners. Don't consider the constraint in Eq. 2.53 could allow the optimizer to search into a arbitrarily large design space, with a high probability to prduce geometries which are very far from the center of the distribution (i.e the mean).

2.5 Application: Shape Optimization of a Naval Destroyer DTMB 5415

In this section section we will apply the new proposed framework for the optimization of a US naval Destroyer DTMB 5415.

Table 2.1. DTMB 5415 model scale main particulars and test condition.

Description	Unit	Value
Displacement	tonnes	0.549
Length between perpendiculars	m	5.720
Beam	m	0.760
Draft	m	0.248
Longitudinal center of gravity	m	2.884
Vertical center of gravity	m	0.056
Water density	kg/m^3	998.5
Kinematic viscosity	m^2/s	1.09E-06
Gravity acceleration	m/s^2	9.803
Froude number	–	0.280

2.5.1 Design Space Parameterization and Sampling

The number of design variables is chosen based on the quality of the shape modification obtained. The number of the design variables should be minimized as much as possible but at the same time allowing for large variability across the shapes generated. Moreover, designs not satisfying the geometrical constraints

Table 2.2. Hull shape modification, FFD control points and variables setup.

Layer	Layer x-plane	No. CPs	No. active CPs	Variable range
1	$\mathsf{x} = 0.00$	12	1	$-1.0 \leq v_y^{(1,2)} \leq 1.0$
2	$\mathsf{x} = 18.21$	12	2	$-1.0 \leq v_y^{(3,4)} \leq 1.0$
3	$\mathsf{x} = 36.42$	12	2	$-1.0 \leq v_y^{(5,6)} \leq 1.0$
4	$\mathsf{x} = 54.63$	12	2	$-1.0 \leq v_y^{(7,8)} \leq 1.0$
5	$\mathsf{x} = 72.85$	12	2	$-1.0 \leq v_y^{(9,10)} \leq 1.0$
6	$\mathsf{x} = 91.06$	12	2	$-1.0 \leq v_y^{(11,12)} \leq 1.0$
7	$\mathsf{x} = 109.27$	12	1	$-1.0 \leq v_y^{13} \leq 1.0$
7	$\mathsf{x} = 109.27$	12	1	$-2.0 \leq v_y^{14} \leq 2.0$
8	$\mathsf{x} = 127.49$	12	1	$-2.0 \leq v_y^{15} \leq 2.0$
9	$\mathsf{x} = 145.70$	12	1	$-2.0 \leq v_y^{16} \leq 2.0$

Table 2.3. Bulb shape modification, FFD control points and variables setup.

Layer	Layer x-plane	No. CPs	No. active CPs	Variable range
1	$\mathsf{x} = 127.20$	9	0	(-)
2, 3	$\mathsf{x} = 133.20 \wedge \mathsf{x} = 139.20$	18	1	$-1.0 \leq v_y^{17} \leq 1.0$
2, 3	$\mathsf{x} = 133.20 \wedge \mathsf{x} = 139.20$	18	1	$-1.0 \leq v_z^{18} \leq 1.0$
2, 3	$\mathsf{x} = 133.20 \wedge \mathsf{x} = 139.20$	18	1	$-0.3 \leq v_x^{19} \leq 0.3$
4	$\mathsf{x} = 139.20$	9	1	$-1.0 \leq v_z^{20} \leq 1.0$
4	$\mathsf{x} = 139.20$	9	1	$0.0 \leq v_x^{21} \leq 0.5$

are not included in the data matrix. As a results, all designs processed by the dimensionality reduction models are feasible. The idea is to define an optimal basis for the representation of the feasible domain. Nevertheless, in case of PCA there are no guarantees that all geometries are feasible during the optimization while with the formulation that we are proposing based on the satisfaction of the Mahalanobis distance constraint this can be accommodated. The design spaces is sampled following a uniform random distribution obtaining $N = 10000$ hull-form feasible design. The data matrix $\mathbf{X}$ collects a $L = 7,200$ grids points from hull discretization consequently since the design modifications for this application are allowed in all the three cartesian components (only for the bulb shape modification) x, y and z the resulting dimensionality in the data space D is equal to $D = L \times 3 = 21600$. In Tab. 2.2 and in Tab. 2.3 there are the configurations of the FFD's control points for the hull and bulb respectively, is possible to see that the total number of design variable used are $M = 21$ with the relative support for the uniform distribution. The lattices for the hull and the bulb are shown in Fig. 2.4 and in Fig. 2.5 respectively.

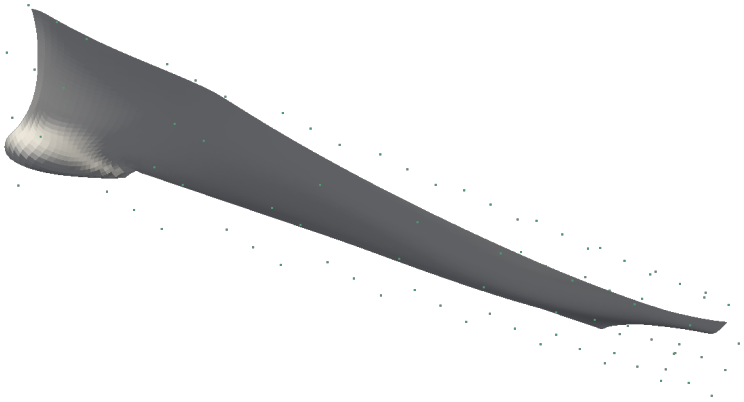


Figure 2.4. FFD control points over the hull.

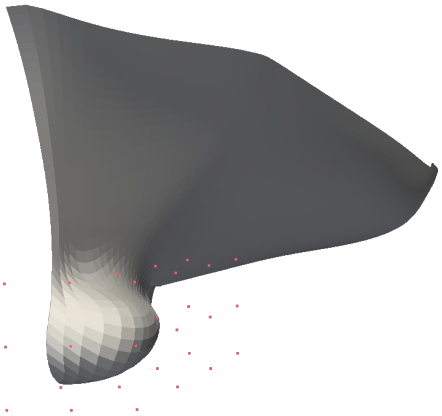


Figure 2.5. FFD control points over the bulb.

2.5.2 Dimensionality Reduction

The first step is to find the right dimensionality of the latent space K. This can be easily assessed respect to the variance that the reparametrization should resolve respect to the original design space representing the explained variance of the model. For this application we fix a threshold of 99% for the explained variance by the PPCA and FA. In Fig. 2.13 and in Fig. 2.14 we show the components $\mathbf{W}$ for the PPCA and FA model directly on the hull form and helps to understand what kind of shape modification we obtain during the transformation given by $\mathbf{Wz} + \bar{\mathbf{x}}$.

In Fig. 2.6 there is the convergence of the explained variance for the two dimensionality reduction models considered in the current analysis namely PPCA and FA. For the PPCA the components are computed using the closed for solution of the marginal likelihood in Eq. 2.42. In this case the explained variance is given by the PCA eigenvalues since $\mathbf{W}$ and $\mathbf{U}$ span the same subspace.

For the FA model we use the EM algorithm to find model parameters. In the FA model the variance explained by each component is given by $||\mathbf{w}_k||^2$, $\forall k = 1, \ldots, K$. As we can see in Fig. 2.6 the PPCA reach the threshold of 99% with $K = 12$ components while the FA with $K = 16$.

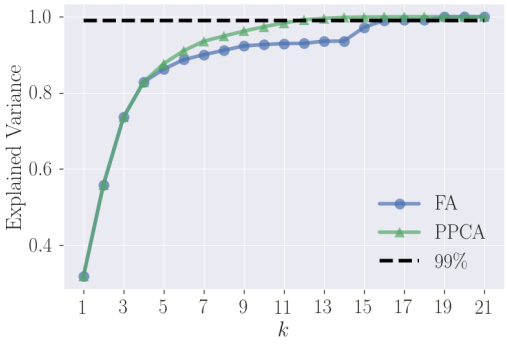


Figure 2.6. Design-space variability retained as a function of the number K-components.

Since the the PPCA components are just scaled PCA eigenvectors, in terms of variance explained is superior respect to FA.

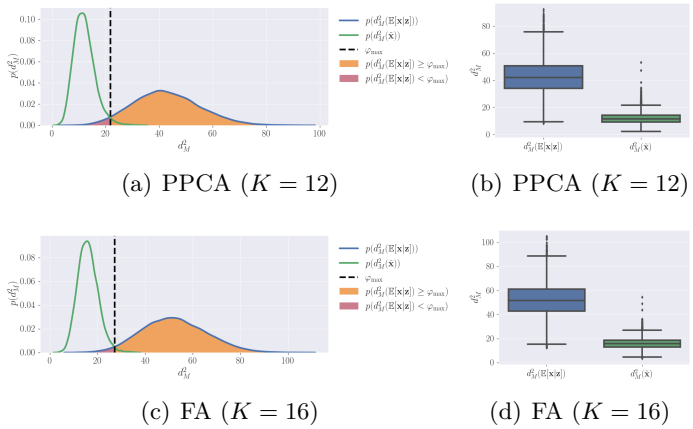
2.5.3 Fixing the Threshold for the Mahalanobis Distance

Before starting the SBDO process we need to fix the constraint for the Mahalanobis distance. The first step is to compute for the K found in the previous section the reconstructed data $\tilde{\mathbf{X}}$. Which means that for all the $\mathbf{x}_n \in \mathbf{X}$ we compute the conditional latent mean for PPCA and FA in Eq. 2.43 and Eq. 2.26 and projecting back all the $\mathbf{z}_n$ in the data space by computing the conditional mean in Eq. 2.36 and Eq. 2.23 for the PPCA and FA respectively. Once we found the matrix $\tilde{\mathbf{X}}$ for the two models we compute the uncertainty associated to each reconstructed geometry $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ as the squared Mahalanobis distance. At the end of this process we have a vector of size N, $\mathbf{d}^2_M(\tilde{\mathbf{x}})$ In this application the threshold $\varphi_{\max}$ is fixed respect to 1.5 times the interquantile range $\text{IQR} = q_3 - q_1$, where q_1 and q_3 are the first and the third quartile of $\mathbf{d}^2_M(\tilde{\mathbf{x}})$. We don't choose for $\varphi_{\max}$ as the maximum value of $\mathbf{d}^2_M(\tilde{\mathbf{x}})$ because during the random sampling procedure could produce outliers. In Tab. 2.4 we summarized the values of the threshold. Finally we performed a simple empirical experiment to show and understand the effect of the density estimation procedure. More precisely we sampled uniformly at random 10000 latent variables $\mathbf{z}$ where as support for the uniform distribution we used the bounds described in Eq. 2.54. Then projected back in data space denoted by$\mathbb{E}[\mathbf{x}|\mathbf{z}]$. This process can be interpreted as performing an optimization procedure using a uniform random sampling as a optimization algorithm. But in this experiment we are only interested in the effect of the constraint in Eq. 2.53. Then we computed the squared Mahalanobis distance denoted by $d^2_M(\mathbb{E}[\mathbf{x}|\mathbf{z}])$ and compared their properties respect to $d^2_M(\tilde{\mathbf{x}})$. The procedure is performed both for PPCA and FA and reported in Fig. 2.7. We can notice is that the center of mass of the distribution of $d^2_M(\mathbb{E}[\mathbf{x}|\mathbf{z}])$ is shifted towards an high value of the Squared Mahalanobis distance respect to

Table 2.4. Dimensionality reduction and density estimation results.

Method	K	$\varphi_{\max}$
PPCA	12	21.74
FA	16	27.13

$d_M^2(\tilde{\mathbf{x}})$. When a latent variable $\mathbf{z}$ is uniformly sampled at random, seems that with



(a) PPCA ($K = 12$) (b) PPCA ($K = 12$)

(c) FA ($K = 16$) (d) FA ($K = 16$)

Figure 2.7. Behavior in terms of the empirical PDF of the squared Mahalanobis distance (left column) and relative box plots (right column).

high probability that this will produce a shape modification far from the mean $\bar{\mathbf{x}}$ of the marginal distribution. Is possible to quantify this effect computing the value of $p(d_M^2(\mathbb{E}[\mathbf{x}|\mathbf{z}]) < \varphi_{\max})$ and $p(d_M^2(\mathbb{E}[\mathbf{x}|\mathbf{z}]) >= \varphi_{\max})$ as shown in Fig. 2.7. The probability that a latent variable sample uniformly at random will be greater than our threshold $\varphi_{\max}$ is ≈ 0.99 and ≈ 0.98 in case of FA and PPCA respectively.

Finally from Fig. 2.7 is possible to notice that the empirical PDF of $p(d_M^2(\tilde{\mathbf{x}}))$ is very similar to a Chi-Square distribution with K degrees of freedom.

2.5.4 Optimization Problem

The problem formulation for the shape optimization of the DTMB 5415 reads

$$\begin{aligned} \min_{\mathbf{v}} \quad & R_{\mathrm{T}}(\mathbf{x}(\mathbf{v})) && \text{with } \mathbf{v} \in \mathcal{R}^M \\ & L_{\mathrm{pp}}(\mathbf{x}(\mathbf{v})) = L_{\mathrm{pp}_0} \\ & \nabla(\mathbf{x}(\mathbf{v})) = \nabla_0, \\ & |\Delta B(\mathbf{x}(\mathbf{v}))| \leq 0.05 B_0, \\ & |\Delta T(\mathbf{x}(\mathbf{v}))| \leq 0.05 T_0, \\ & V(\mathbf{x}(\mathbf{v})) \geq V_0, \\ & v_i^{\mathrm{lb}} \leq u_i \leq v_i^{\mathrm{ub}} && \forall i = 1, \ldots, M \end{aligned} \tag{2.62}$$

where R_{T} is the calm-water resistance at $\mathrm{Fr} = 0.28$ (equivalent to 20 kn for the full-scale ship). Equality constraints are defined for the length between perpendiculars (L_{pp}) and for the displacement (∇). Inequality constraints include 5% of maximum variation of beam (B) and the drought (T) and dedicated volume for the sonar dome (V), corresponding to 4.9 m diameter and 1.7 m length (cylinder). Subscript '0' indicates original-geometry values. Equality and inequality constraints on the geometry deformations are based on [36] Using the reduced-dimensionality design space with probabilistic linear latent variable models PPCA and FA, the optimization

problem (Eq. 2.63) is recast as

$$\begin{aligned} \min_{\mathbf{z}} \quad & R_{\mathrm{T}}(\mathbf{x}(\mathbf{z})) && \text{with } \mathbf{z} \in \mathcal{R}^K \\ & L_{\mathrm{pp}}(\mathbf{x}(\mathbf{z})) = L_{\mathrm{pp0}} \\ & \nabla(\mathbf{x}(\mathbf{z})) = \nabla_0, \\ & |\Delta B(\mathbf{x}(\mathbf{z}))| \leq 0.05 B_0, \\ & |\Delta T(\mathbf{x}(\mathbf{z}))| \leq 0.05 T_0, \\ & V(\mathbf{x}(\mathbf{z})) \geq V_0, \\ & d_M^2(\mathbf{x}(\mathbf{z})) \leq \varphi_{\max} \\ & z_i^{\mathrm{lb}} \leq z_i \leq z_i^{\mathrm{ub}} && \forall i = 1, \ldots, K \end{aligned} \tag{2.63}$$

Where the bounds for the latent variable are computed by taking the maximum and the minimum of each column component z_k of $\mathbf{Z}$ and the value of $\varphi_{\max}$ given in Tab. 2.4.

2.5.5 Hydrodynamic Solver

The calm-water total resistance is evaluated using the linear potential flow code WARP (Wave Resistance Program), developed at CNR-INM. Wave resistance computations are based on the Dawson (double-model) linearization [27]. The frictional resistance is estimated using a flat-plate approximation, based on the local Reynolds number [111]. The ship balance (sinkage and trim) is fixed. Details of equations, numerical implementations, and validation of the numerical solver are given in [9]. Simulations are performed for the right demi-hull, taking advantage of symmetry about the $\xi_1\xi_3$-plane. The computational domain for the free-surface is defined within $1L_{\mathrm{pp}}$ upstream, $3L_{\mathrm{pp}}$ downstream, and $1.5L_{\mathrm{pp}}$ sideways, for a total of 150×44 grid nodes. The associated hull grid if formed by 180×40 nodes.

2.5.6 Numerical Results

The global optimization process is performed using the DIRECT algorithm [69] and Bayesian Global Optimization [90] based on Gaussian Process (GP) [102]. The stopping criteria is the maximum number of function evaluations fixed at $I_{\max} = 500$. For the GP, we used the lower confidence bound (GP-LCB) [133] as utility function, with the parameter that balance the exploitation/exploration is fixed to one. The LCB is optimized with a multi-start Quasi-Newton method BFGS [16]. In this work, a Matérn kernel function [49] with a parameter $\nu = 3/2$ has been used for the GP, while its length scale is optimized during the Maximum Likelihood Estimation (MLE) procedure.

The hidden constraints are treated as following: in case one or more constraints are not satisfied, the geometry will not enter the simulator to obtain the function evaluation, instead the following pseudo value for the objective function is returned back to the optimizer as

$$f(\mathbf{x}) = h + \psi * \sum_{j=1}^{J} \max\{0, g_j(\mathbf{x}) - a_j\} \tag{2.64}$$

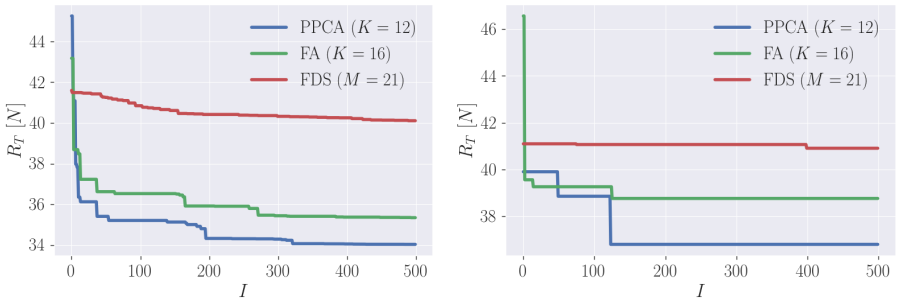
where $h = 50$ and $\psi = 1000$. In Tab. 2.5 the numerical results for each optimization algorithm and for the three design spaces produced by PPCA ($K = 12$), FA ($K = 16$)

and the full dimensional design space FDS ($M = 21$) in terms of the water resistance R_T(N). The overall best function value obtained is given by the DIRECT algorithm iterating in the subspace produced by the PPCA. In Fig. 2.8 we could evaluate the converge speed to the achieved optimal function value. We can notice from these figures that optimizing in the latent space shows a better convergence speed respect to the FDS.

In Fig. 2.9 is possible to evaluate the quality of the optimal geometries (especially the bulb region) produced through FA and PPCA and their relative Mahalanobis distance. In general is possible to notice a good regularity of the shape modification across all the optimal solutions

Table 2.5. Numerical results at the end of the SBDO process.

Method	K	DIRECT R_T(N)	GP-LCB R_T(N)
PPCA	12	34.04	36.81
FA	16	35.35	38.77
FDS	21	40.10	40.91



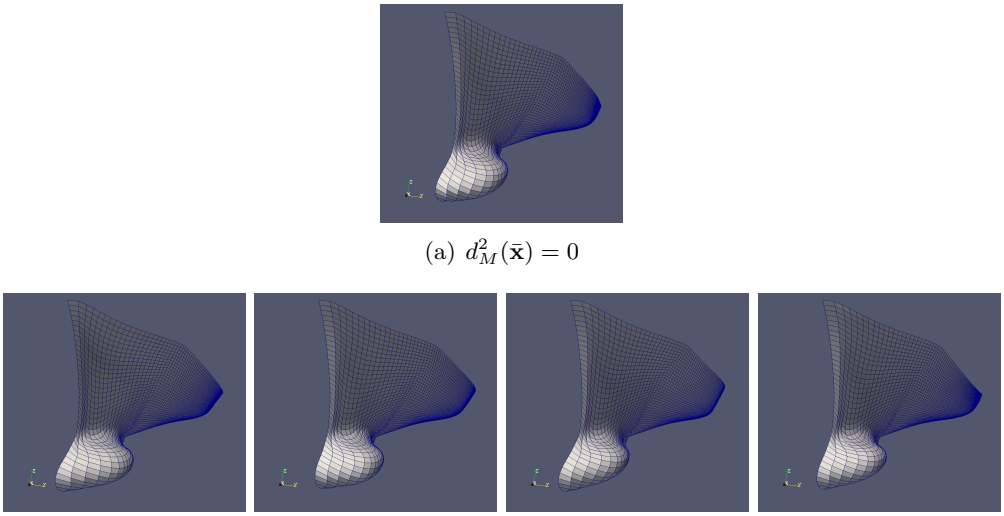
(a) DIRECT (b) GP-LCB

Figure 2.8. Convergence of DIRECT and GP-LCB during the optimization in FA, PPCA and FDS spaces.

Finally the SBDO process is applied also for the PCA with $K = 12$. In Fig. 2.10 we can see the convergence of the two optimizers used in this application.

In order to understand if the geometries produced by the PCA during the SBDO process are out of distribution respect to the training data, we evaluated the squared Mahalanobis distance using the inverse of the covariance matrix $\mathbf{C}^{-1}$ estimated by the PPCA model since the eigenvectors $\mathbf{U}$ and the matrix $\mathbf{W}$ span the same subspace. Most of the latent variable produced by the optimization algorithms when projected back by the PCA for function evaluation don't satisfy the PPCA Mahalanobis distance constraint.

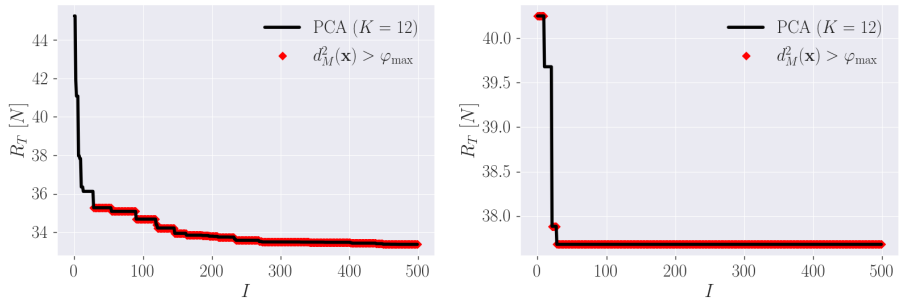
In Fig. 2.11 there is the distribution for the squared Mahalanobis distance computed respect to the geometries sample with PCA during the optimization with DIRECT and GP-LCB. For the GP-LCB the geometrically feasible geometries produced by the PCA that don't satisfy the PPCA Mahalanobis distance constraint



(a) $d^2_M(\bar{\mathbf{x}}) = 0$

(b) $d^2_M(\mathbf{x}) = 21.73$ (c) $d^2_M(\mathbf{x}^\star) = 15.05$ (d) $d^2_M(\mathbf{x}^\star) = 15.14$ (e) $d^2_M(\mathbf{x}^\star) = 27.12$

Figure 2.9. Optimal solutions $\mathbf{x}^\star$ and relative (squared) Mahalanobis distance for PPCA/DIRECT (b), PPCA/GP-LCB (c), FA/DIRECT (d) and FA/GP-LCB (e). The mean geometry $\bar{\mathbf{x}}$ in (a).



(a) DIRECT (b) GP-LCB

Figure 2.10. Convergence of DIRECT and GP-LCB during the optimization in PCA space. In red a geometry which does not satisfy constraint based on the Mahalanobis distance.

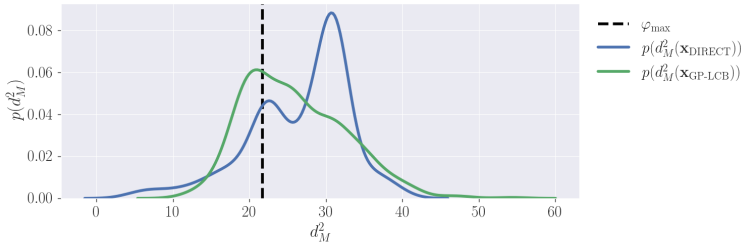
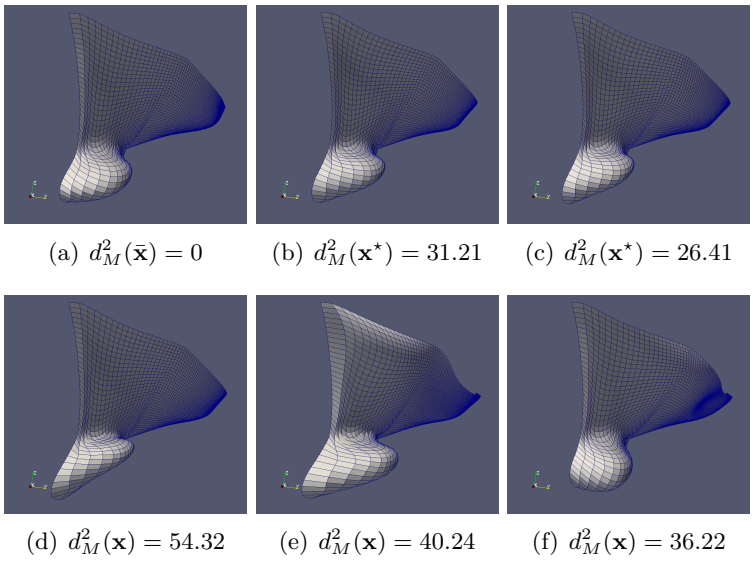


Figure 2.11. Empirical PDF for the squared Mahalanobis distance for the geometries sampled in PCA subspace from DIRECT and GP-LCB.

are the $\approx 67.4\%$ while with the DIRECT algorithm this fraction is approximately $\approx 81.6\%$ of the total number of function evaluations performed.



(a) $d^2_M(\bar{\mathbf{x}}) = 0$ (b) $d^2_M(\mathbf{x}^\star) = 31.21$ (c) $d^2_M(\mathbf{x}^\star) = 26.41$

(d) $d^2_M(\mathbf{x}) = 54.32$ (e) $d^2_M(\mathbf{x}) = 40.24$ (f) $d^2_M(\mathbf{x}) = 36.22$

Figure 2.12. Optimal solutions $\mathbf{x}^\star$ and relative (squared) Mahalanobis distance for PCA/DIRECT (b) and PCA/GP-LCB (c). Examples of three geometries (geometrically feasible) sampled during the optimization routine in (d) (e) and (f). The mean geometry $\bar{\mathbf{x}}$ in (a).

In Fig. 2.12 there are the optimal solutions and the three examples of geometries obtained during the SBDO process. In general, from Fig. 2.12 we obtain a front part of bulb squashed towards the symmetry axis as in 2.12(b), 2.12(c), 2.12(d). This the main reason why geometry 2.12(b) reaches a low value of $R_T(\mathrm{N}) = 33.3$. A similar behavior can be noticed in the optimal solution obtained with DIRECT with the PPCA in Fig. 2.9(b), but in this case the constraint based on the Mahalanobis distance forces the shape of the bulb to be more regular and similar to the mean geometry. Let's notice also, that in 2.12(f) the bulb tends towards a negative value of the z coordinate together with a curious depression obtained over the hull.

To conclude, don't consider the uncertainty of the reduced parametrization could lead to the big disadvantage to waist expensive time consuming function evaluations for meaningless geometries that are far from the original design space carefully defined in our data $\mathbf{X}$.

2.5.7 Conclusions and Future Works

In this chapter we proposed a new framework for SBDO in shape optimization. This is based on coupling a dimensionality reduction of the original design variables with a density estimation of the probability distribution of the data. This is achieved by adding a constraint which measure the uncertainty (in data space) of the shape modification defined in the latent space. The uncertainty is measured in terms of the Mahalanobis distance. Also we showed that when the shape modification method is linear and the design variables are sample uniformly at random, the generated data follows approximately a Gaussian distribution. This is a consequence of the central limit theorem (CLT).

The inverse of the covariance, that is needed to compute the uncertainty, is estimated by using two linear probabilistic latent variable models: Factor Analysis and Probabilistic PCA. These methods, provide also the construction of the latent space, namely the reduced dimensionality space. The new proposed framework is demonstrated at the end for the shape optimization of naval military US destroyer. From the optimization results carried out with two global optimization algorithm DIRECT and GP-LCB, showed that reduce the dimensionality allows a greater reduction in the objective function respect to the full-dimensionality space. At the end we also performed the optimization using the PCA. Most of the time are generated geometries that are 'far' from the original parameterization with the great penalty to have wasted many function evaluations.

In the future work, the possibility to use a finer grid for the hull discretization can be considered since through the EM algorithm is possible to learn the reduced parametrization and the relative uncertainty without storing the covariance matrix in the main memory. Also, high fidelity simulators together and a physics informed formulation could be considered.

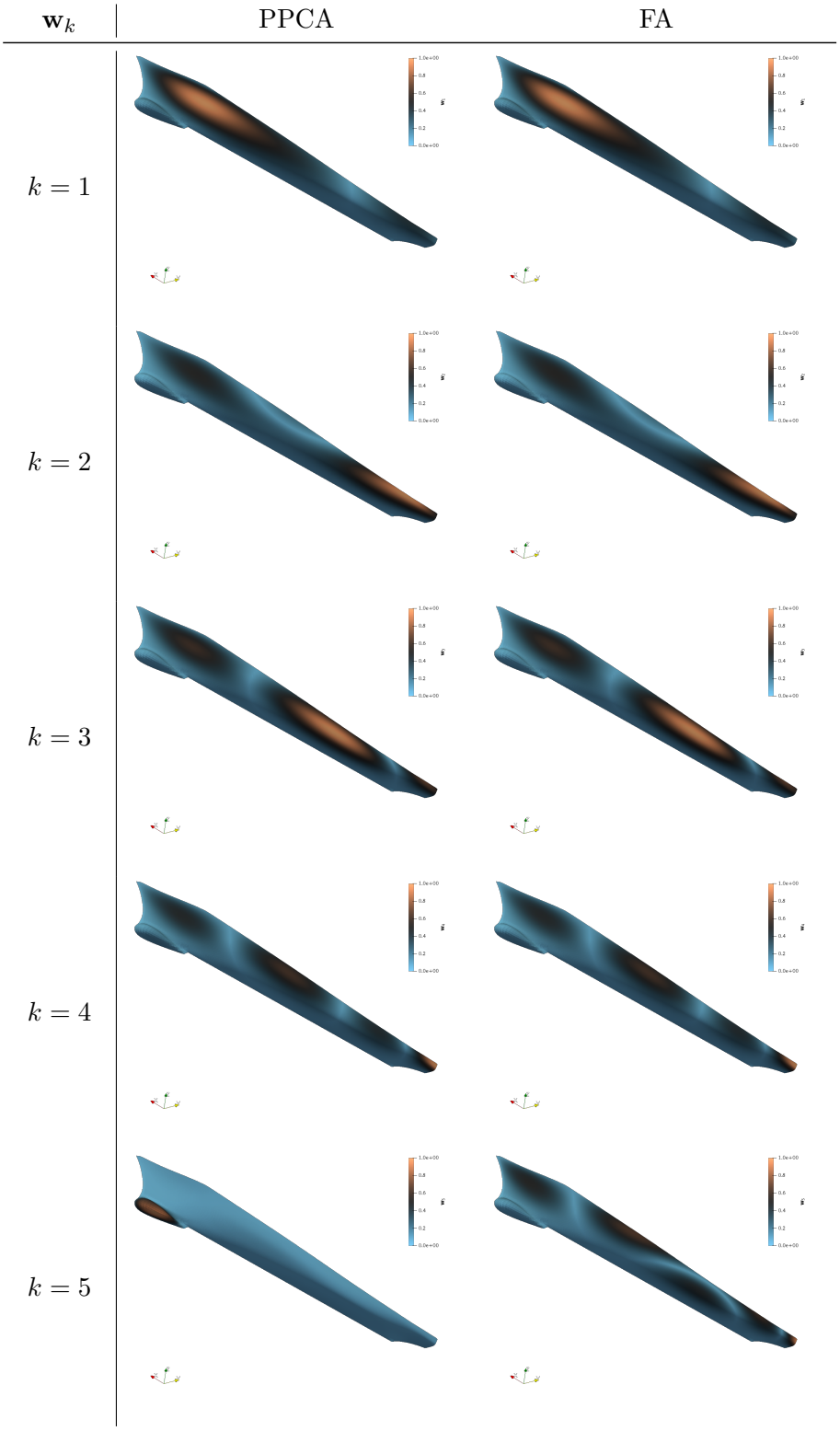


Figure 2.13. Module of the first five PPCA and FA components.

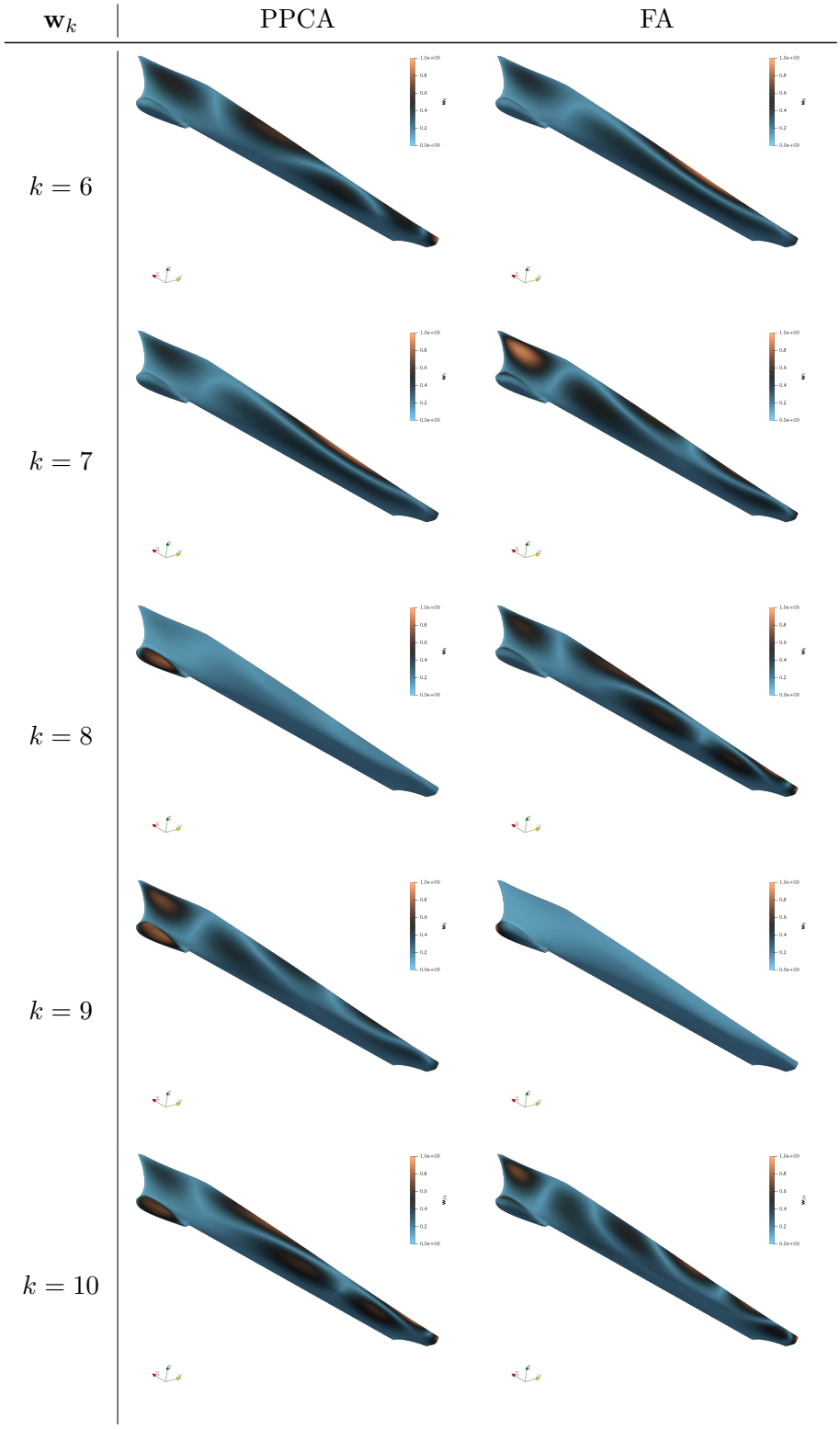


Figure 2.14. Module of the last five PPCA and FA components.

Chapter 3

Data-driven Analysis of Turbulent Flows

In this chapter, we will show a data-driven analysis carried out for two high Reynolds number vortices flows namely for uniform and buoyant jets and 4- and 7-bladed propeller wakes. The methodology is based on the global flow analysis using the Projection Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD) for the Uniform and Buoyant Jet. Also Spatial and snapshot clustering approaches are presented and discussed for particle image velocimetry (PIV) Data clustering is based on the k-means algorithm, along with the identification of the optimal number of clusters based on the energy retained by the dominant POD mode. Spatial clustering for jets flow is based on three sets of clustering variables, namely cross-section velocity profiles, point-wise energy spectra, and point-wise Reynolds stress tensor components. Snapshot clustering of phase-locked propellers wake data is based on the vorticity with a focus on tip vortices regions. POD and t-distributed stochastic neighbor embedding along with kernel density estimation are used to provide a two-dimensional visualization of data clusters for assessment and discussion. The objective of this work is to lay the ground for a systematic data-clustering analysis of PIV data. The examples discussed show how clustering methods can help in achieving physical insights into complex fluid dynamics problems. Most of the results presented in this chapter are based on [41, 120], the experiments for the Buoyant and uniform jet were conducted at George Washington University and for the propeller wakes at the Institute of Marine Engineering of the Italian National Research Council (CNR-INM).

3.1 Introduction

Turbulent flows are very common and simple to observe phenomena from everyday life to many areas of engineering, but at the same time are incredibly difficult to describe and explain. In nature, turbulence are defined as a manifestation of the spatio-temporal chaotic behavior of fluid flows at large Reynolds numbers, i. e. of a strongly nonlinear dissipative system with extremely large number of degrees of freedom (most probably) described by the Navier-Stokes equations [139]. Following [137] and [139] the main characteristics for turbulent phenomena are

briefly summarized in

- *Intrinsic spatio-temporal randomness*: turbulent flows show randomness in the characteristics of the vortices in position and orientation. However this is usually expressed as deterministic chaos since the Navier Stokes equations completely provide a deterministic description of flows.
- *Wide range of strongly interacting scales*: the vortex structures, besides the intrinsical randomness are distinguished by a wide scale of length measuring units
- *Loss of predictability*: this is due to the fact that small perturbations/ disturbances could be be highly amplified over time due to the high nonlinearities.
- *Turbulent flows are highly dissipative*: since turbulences are dissipative processes a source of energy is needed in order to conserve the phenomena over time. This is performed in the main stream and then transferred towards smaller scales
- *Turbulent flows are three-dimensional and rotational*: Vortex structures occur in the space of a turbulent flow field in random locations and with random orientation. The 3D structure of the vector field of velocity fluctuations originates from this situation.
- *Strongly diffusive*: turbulent flows show strongly enhanced transport processes of momentum, energy and passive objects.

As already mentioned above, turbulent flows can be deterministically described by the Navier-Stokes equations. However is important to point out that direct numerical simulation (DNS) of the Navier Stokes equations are usually only possible to be performed in case of simple geometries and a moderate Reynolds number due to the high computational power required. Also the difference in the scale resolution between an experiment ran in the laboratory and a numerical experiment could lead to erroneous results [139]. For these reasons the study and the investigation of turbulent flows are also performed by physical experiments. During the years researchers provided many ways to measure turbulence from nuclear magnetic resonance to the particle image velocimetry. Those methodologies provides data (or snapshots) in space and in time about the quantity of interests (e.g. vorticity).

In the last years the Data Science field has emerged as a cutting-edge research field that develops and applies rigorous methods and algorithms to gain knowledge from data. In recent years, data science methods have been applied in the context of more traditional disciplines to accelerate the experimental/computational analysis process and extract insights from experiments and simulations data. Unsupervised machine learning methods (such as proper orthogonal decomposition formulations, POD, along with linear/nonlinear principal component analysis, PCA) have been used to reduce the problem dimensionality, plan more efficient design- and/or operational-space explorations, and gain insights of complex physical phenomena. POD has been widely used for the identification of coherent structures in turbulent flows [11] and applied to steady/transient uniform/buoyant jets [55, 152] [83]and

marine propeller wakes [43]. In the discrete form, POD is equivalent to PCA and allows to decompose the flow into a linear combination of a subset of orthogonal eigenfunctions, capable of highlighting its spatial/temporal structure and providing a reduced-order/dimensionality model for the flow dynamics. Although POD is a widely used and has well-established global optimality properties, it is based on the linearity, stationarity, and ergodicity assumptions and may not be effective when nonlinear, transient, non-stationary, non-ergodic dynamics are investigated. For this reason nonlinear dimensionality reduction (NLDR) methods have been developed and applied to provide with a deeper understanding of data structures and physical phenomena. A straightforward approach to NLDR with POD/PCA is to use data clustering methods and perform POD/PCA within each cluster. Cluster-based reduced-order and/or dimensionality modeling by POD/PCA can provide with physical insight of complex phenomena and is achieved by local PCA (LPCA), where the data set is divided into clusters and POD/PCA is applied to each cluster, assuming therefore an approximate linear structure within each cluster. The cluster centroids along with the associated modes are used to extract relevant flow feature in the spatial/temporal domains. Applications of spatial clustering via the k-means method with POD/PCA have been presented in [120] for a transient buoyant jet. POD/PCA approaches based on temporal and spatial k-means clustering have been presented in [8] for a swirl-stabilized combustor flow. In general, the number of clusters and the similarity metrics used for data clustering highly affect the quality of the resulting reduced-order/dimensionality model, and therefore the possibility to gain valuable physical knowledge from the clustering analysis. Rigorous data-clustering methods can assist in achieving a deeper understanding of experiments/simulations data and have been proposed in different fields, such as computer vision and speech recognition. A further step towards fully nonlinear dimensionality reduction of data sets has been proposed via t-distributed stochastic neighbor embedding (t-SNE) by [141]. The method provides with the capability of embedding and visualizing high-dimensional data in a low-dimensional space and has been applied to turbulence data sets from simulations in [148]. Data analysis methods are outlined in the following. Specifically, the k-means clustering method is briefly recalled. POD/PCA implementation is described and used both as metric for identifying an optimal number of clusters and a visualization technique. For the same purpose, the t-SNE method is used as visualization technique and briefly discussed along with KDE.

3.2 Data-driven methods for Physical Experimental Data Analysis

In this section we will describe various methodologies for the analysis of turbulent flows used in this work.

3.2.1 Proper Orthogonal Decomposition

The Projection Orthogonal Decomposition[11] is Fluid Dynamics is appllied respect to the Reynolds decomposition of the velocity vector, with

$$u = \overline{u} + u' \qquad v = \overline{v} + v' \tag{3.1}$$

where u and v indicate x (axial/vertical) and y (horizontal) components, respectively. Overbar and prime characters indicate time average and fluctuations, respectively. The POD then follows the same procedure as classical PCA. The $(L \times T)$ data matrix $\mathbf{X}$ is defined as

$$\mathbf{X} = \left[\begin{array}{c|c|c} \mathbf{x}_1 & \dots & \mathbf{x}_T \end{array} \right] \tag{3.2}$$

where $\mathbf{x} = \{u'(\mathsf{x}_1), \dots, u'(\mathsf{x}_P), v'(\mathsf{x}_1), \dots, v'(\mathsf{x}_P)\}^\top$ collects the discretized velocity fluctuations, x_i represents the i-th node of the spatial discretization, P is the spatial discretization size, superscript (t), with $t = 1, \dots, T$, indicates the t-th time realization (snapshot), and finally $L = 2P$. The data matrix $\mathbf{X}$ is reduced in dimensionality through projection of the snapshots into a new linear subspace, formed by the eigenvectors of the $(L \times L)$ sample covariance matrix

$$\mathbf{S} = \frac{1}{T} \mathbf{X}\, \mathbf{X}^\mathsf{T} \tag{3.3}$$

evaluated by

$$\mathbf{S}\mathbf{W} = \mathbf{W}\mathbf{\Lambda} \tag{3.4}$$

where $\mathbf{W}$ and $\mathbf{\Lambda}$ collect the L eigenvectors $(\mathbf{w}_i)$ and eigenvalues (λ_i) of $\mathbf{S}$, respectively. This corresponds to performing the PCA of the matrix $\mathbf{X}$. The problem of Eq. 3.4 may be alternatively solved using the singular value decomposition, SVD [53]. Furthermore, if $L > T$ the dual problem may be solved via the so-called snapshot-POD (equivalent) formulation, see e.g. [20].
POD/PCA eigenvalues represent the variance (under proper assumptions this represents the turbulent kinetic energy) resolved along the corresponding eigenvectors. The linear subspace formed by the M eigenvectors (POD/PCA modes, collected in $\hat{\mathbf{W}}$) associated to the largest M eigenvalues resolves (globally) the largest variance/energy, compared to any other linear subspace of dimension M [12, 62]. The cumulative sum of the eigenvalues is used to assess the variance resolved by the linear subspace of dimension M. Finally, the associated reconstruction of $\mathbf{X}$ is given by $\hat{\mathbf{X}} = \mathbf{W}\mathbf{W}^\top\mathbf{X}$, where by definition the coefficient or latent variable $\mathbf{z}_i = \mathbf{z}_i^\top\mathbf{X}$ is the projection of the data matrix onto the i-th mode.

3.2.2 Dynamic Mode Decomposition

The Dynamic Mode Decomposition DMD introduced by [112], becomes together with POD one of the most useful technique for fluid flows analysis. The DMD provide a modal decomposition highlighting the information about the temporal dynamics of the phenomena. As the POD, the eigenvectors are spatial fields, but in the case of the DMD the relatives eigenvalues give the information about the decay/grow and oscillatory frequencies of each mode.

The core idea of the DMD is to find an approximation of the eigenvectors and eigenvalues of the unknown matrix $\mathbf{A}$ which describe the dynamics $\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t$. There are mainly two approaches that can be used to find the DMD modes: one using the companion matrix and the Arnoldi Algorithm the other, more numerically stable using the SVD. Following the latter approach the procedure starts by computing the matrices $\mathbf{X}_1 = [\mathbf{x}_{t=1} \dots \mathbf{x}_{k=T-1}]$ and $\mathbf{X}_2 = [\mathbf{x}_{t=2} \dots \mathbf{x}_{t=T}]$ then using Singular Value Decomposition (SVD)

$$\mathbf{A} = \mathbf{X}_2\mathbf{X}_1^\dagger = \mathbf{X}_2\mathbf{H}\mathbf{\Sigma}^{-1}\mathbf{W}^\top$$

Where $\mathbf{X}_1^{\dagger}$ is the pseudo-inverse of $\mathbf{X}_1$ and is decomposed with its SVD decomposition. A compressed representation of $\mathbf{A}$ could be obtained using the following decomposition

$$\tilde{\mathbf{A}} = \mathbf{W}^T\mathbf{A}\mathbf{W} = \mathbf{W}^T\mathbf{X}_2\mathbf{H}\boldsymbol{\Sigma}^{-1}$$

The matrix $\tilde{\mathbf{A}}$ is similar to the matrix $\mathbf{A}$ so that they have the same eigenvalues which we compute using

$$\tilde{\mathbf{A}}\mathbf{Z} = \mathbf{Z}\boldsymbol{\Lambda}$$

In order to compute the DMD modes we project back in the original space the eigenvectors $\mathbf{Z}$

$$\boldsymbol{\Phi} = \mathbf{W}\mathbf{Z}$$

The matrix $\boldsymbol{\Phi}$ is composed by the DMD modes. The DMD modes represents coherent spatial components of the flow while their growth/decay rate and the frequency is given by the relative real and imaginary part of the eigenvalues respectively, therefore the temporal dynamic of each dynamic mode is highlighted.

3.2.3 k-Means Clustering

The k-means is a widely used clustering method [67], which allows to build partitions of the original data collected in an $(L \times T)$ matrix $\mathbf{X}$ in k different sets (clusters), defined by representative points (centroids). The Euclidean distance is used to measure both the similarity between data points $\mathbf{x}_j$ and evaluate the associated cluster centroids $\boldsymbol{\mu}_i$, the latter by averaging all data points within the i-th cluster $\mathbf{X}_i$. The assignment of data points to k clusters is achieved by the minimization of the squared Euclidean distance between $\boldsymbol{\xi}_j$ and $\boldsymbol{\mu}_i$ (within-cluster sum of squares, WCSS)

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in \mathbf{X}_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 \tag{3.5}$$

The k-means problem is NP-hard [40]. Therefore, the heuristic approach presented in [83] is used. Results are highly sensitive to centroids initialization. Here, the initialization strategy proposed in [4] is used.

3.2.4 t-Distributed Stochastic Neighbor Embedding

The t-SNE is a machine learning algorithm proposed by [141], which is found very effective for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. The t-SNE first constructs joint probability densities p_{ij} that reflect pairwise similarly among data points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ parameterized by a Gaussian distribution

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{3.6}$$

with

$$p_{j|i} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\varsigma_i^2\right)}{\sum_{k \neq i}^{N} \exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\varsigma_i^2\right)} \tag{3.7}$$

In a similar manner the joint probability densities q_{ij} are defined for the low-dimensional representations $\mathbf{z}_i$ and $\mathbf{z}_j$, parameterized by a t-student distribution

$$q_{ij} = \frac{(1 + ||\mathbf{z}_i - \mathbf{z}_j||^2)^{-1}}{\sum_{k \neq i}^{T} (1 + ||\mathbf{z}_i - \mathbf{z}_k||^2)^{-1}} \tag{3.8}$$

The points (or coefficients) $\mathbf{z}_i$ are determined by minimizing the Kullback-Leibler divergence of the distribution q from the distribution p as

$$\mathrm{KL}(p||q) = \sum_{i \neq j}^{T} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) \tag{3.9}$$

The minimization of Eq. 3.9 with respect to the points $\mathbf{z}_i$ is performed using gradient descent. The parameter ς_i is set in such that the perplexity of the conditional distribution $P_i = \sum_j p_{ji}$ over all data points given $\mathbf{x}^{(i)}$ equals a predefined perplexity $\mathrm{Perp}(P_i)$,

$$\mathrm{Perp}(P_i) = 2^{\left(\sum_j^T p_{j|i} \log_2 p_{j|i} \right)} \tag{3.10}$$

which is solved with a bisection method. The perplexity can be interpreted as a smooth measure of the effective number of neighbors, with typical values ranging from 5 and 50 [141].

3.2.5 Multivariate Kernel Density Estimation

The kernel density estimation (KDE, [128]) is a non-parametric method to estimate the probability density function (PDF) of a random variable, introduced for univariate data. Extending the concept to multivariate data [129], let $\{\boldsymbol{\alpha}\}_{i=1}^{T}$ be a d-variate random vector whose PDF is estimated as

$$\mathrm{PDF}(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} K_{\mathbf{H}} \left(\mathbf{x} - \mathbf{x}_i \right) \tag{3.11}$$

$\mathbf{H}$ is the bandwidth (or smoothing) $[d \times d]$ matrix which is symmetric and positive definite, and K is the kernel function which is a symmetric multivariate density defined as

$$K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2}\mathbf{x}) \tag{3.12}$$

3.3 Application

The objective of the present work is to lay the ground for a systematic data-clustering analysis of particle image velocimetry (PIV) data with the aim of achieving physical insights of complex fluid dynamics problems. Examples are provided for high-Reynolds number uniform/buoyant transient jets along with 4- and 7-bladed propeller wakes.

The velocity fields under investigation (both for jets and propellers) are obtained from experimental tests with large scale, time/phase-resolved, PIV measurements. GWU provided data for the jets, whereas data for the propeller wake were collected at CNR-INM.

Table 3.1. Summary of test cases and clustering approaches.

Test case	Uniform/buoyant jets	4/7-bladed propeller wakes
Data dimension	73,678 × 4,600 / 3,141	6,000 × 1,000 / 10,000 × 500
Data analysis type	Global flow and clustering analysis	Clutering analysis
Clustering domain	Spatial	Snapshots
Clustering variables	Cross-section velocity profiles Point-wise energy spectra Point-wise Reynolds stress	Vorticity

Clustering of PIV data is based on the k-means algorithm. Three clustering approaches are applied to the jet in the spatial domain to identify coherent/self-similar spatial regions using the following clustering variables: (a) cross-section velocity profiles, (b) point-wise energy spectra, and (c) point-wise Reynolds stress tensor components. The resulting clusters and centroids are representative of the local flow, in terms of cross-section profiles and turbulence variables. Data clustering for the propeller wake is applied to phase-locked snapshots to gain knowledge on the topology of wake-instability and its stochastic realizations. The vorticity is used as clustering variable. The resulting cluster centroids identify the topology of the instability, where two or more tip vortices interact and coalesce.

Three metrics are proposed for the identification and assessment of clustering methods, including the selection of the proper number of clusters, namely: (a) within-cluster sum of squares, (b) average silhouette, and (c) within-cluster number of POD modes required to resolve prescribed levels of total variance/energy. Additionally, embedding of data via POD/PCA and t-SNE is used to define and visualize data clusters in a reduced dimensionality space. Finally the kernel density estimation (KDE) is applied to POD/PCA and t-SNE representations to provide with continuous data distributions for assessment and discussion. A summary of test cases and clustering approaches used is presented in Table 3.1.

3.3.1 High-Reynolds Number Uniform and Buoyant Jets

The experiment is conducted at GWU and is the vertical discharge of high-Reynolds number uniform and buoyant transient jets. The latter is discharged in a linearly stratified environment. To enable optical diagnostic deployment, two refractive index matched solutions of different densities are employed; their density difference is 3.00%. Additionally, the dynamic viscosity of the solutions are within 0.7% of each other at 20°C. Details on the refractive index and dynamic viscosity matching, as well as on the linear stratification formation, are reported in [21]. The facility produces a round vertical jet inside a clear acrylic tank (cube of 914 mm-side), Figure 3.1. A linear motor drives a piston in a 203 mm-diameter cylinder, which pushes the fluid through a first contraction section followed by a contoured nozzle with a $D = 6.35$ mm exit diameter.

The fluid in the cylinder is initially at rest and the jet has no initial disturbances. The piston-cylinder and contraction sections lead to a jet with a top-hat velocity profile. The jet Reynolds number is $Re_D = U_\circ D/\nu$ is 2.00×10^4. The run time, limited by the stroke of the piston, is 39 s. The change in height in the tank from

each run is 5.4 mm or about $0.8D$.

During the discharge of the buoyant jet, the stratified environment evolves continuously; the flow might not reach statistical stationarity. Therefore, the whole time history of the velocity fields is recorded in a time-resolved manner: initial circulations in the tank, the entire run, and the settling down after the jet ends. Additionally, the velocity field is recorded from the jet centerline to the wall of the tank. The recorded flow area is nearly 0.7 m horizontally by 0.5 m vertically. The spatial and temporal scales vary greatly over the field of view and to optimize the acquisition system, a multi-camera array is employed. Cameras 1 to 9 are 1.3 MPixel CMOS cameras and record the off-center and far field of the jet. They record either at 64 or 128 Hz depending on their radial location. Camera 10 is a CMOS camera with CoaXpress transfer protocol. It records on the centerline of the jet at 512 Hz at 4 MPixels. The spatial resolution for those cameras is on the order of the Taylor scale. Finally, two other cameras are recording at higher resolution, but the data are not treated here. Three large laser sheets illuminate the fields of view the cameras. They are split from a single cavity of a dual cavity Nd:YLF laser (Photonics DM 527) operated at nearly 30 mJ/pulse. The intensity of each laser sheet is controlled individually by a set of beam splitters. Each laser is configured as a telescope, with a nearly constant 3 mm thickness. Data are processed with Davis 8.4.0 from LaVision. The velocity fields recorded at 64 Hz are first up-sampled to 128 Hz using Davis super-time-sampling function, and data at 512 Hz are down-sampled at 128 Hz. Once all velocity fields are sampled at the same rate, they are spatially stitched together, applying a sliding average over the areas where cameras overlap.

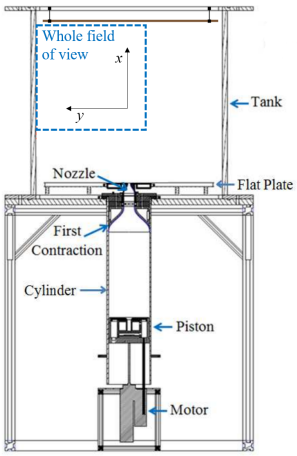


Figure 3.1. Experimental facility for the jets with the location of the PIV whole field of view.

3.3.2 Propeller Wake

The study is based on a comprehensive database of detailed flow measurements of the notional E1658 submarine propeller wake in open water using 2D-PIV (see [43, 98]). The database covers an extensive set of propeller conditions in terms of advance coefficients and blade number configurations providing a wide range of vortex instability and interaction mechanisms that are crucial for the objectives

of the present study. In particular, the present study focuses on two propeller configurations with 4 and 7 blades and one value of the advance ratio, corresponding to a high propeller loading (i.e. $J = 0.56$).

The survey was carried out at the CNR-INM cavitation tunnel (i.e. 2.7 m long by 0.6 m width by 0.6 m height test section, 2% highest free-stream turbulence, mean velocity uniformity within % for the axial component and 3% for the vertical component), measuring the propeller wake flow at the vertical centerplane by a system of multiple, side-by-side, synchronous cameras, 2560×2160 pixels each, and two 200 mJ/pulse Nd-YAG lasers. This arrangement, already adopted in other similar experiments [45, 44, 43] of the propeller wake, allowed the simultaneous reconstruction of a long portion of the wake flow (i.e. from the propeller plane to $3.3D$ downstream, where D is the propeller diameter) without jeopardizing the spatial resolution. More detailed information on the experimental set up are reported in [43].

Camera acquisition was conditioned upon the passage of the propeller reference blade for a selected angular position. This was achieved by synchronizing the four cameras and the two lasers to a TTL OPR (i.e. Once Per Revolution) signal, supplied by a 3600 pulse/sec rotary incremental encoder mounted on the propeller dynamometer.

3.4 Clustering Analysis for Turbulent PIV Data

3.4.1 Spatial Clustering Approach

Clustering approaches in the spatial domain are proposed for the turbulent transient jets, using as clustering variables (1) cross-section velocity profiles, (2) point-wise energy spectra, and (3) point-wise Reynolds stress tensor components, whereas propellers wakes are clusterized in the snapshot domain based on the vorticity only. Here, the original data in $\mathbf{X}$ is rearranged as per the clustering approach and criterion used. In general, we refer to $\mathbf{x}_j$ as one realization (point) of the rearranged data. Note that, generally, $\mathbf{x}_j \in \mathcal{R}^Q$ with $j = 1, ..., H$, where $Q \neq L$ and $H \neq T$. The formulation underlying each approach is described in the following.

Cross-Section Velocity Profiles

Cross-sections are clustered together, based on their velocity profiles. Firstly, cross-section (x-constant) velocity profiles are scaled and secondly used as variables in the clustering process. Specifically, the profiles of the following variables are stitched together to form clustering arrays in Eq. 3.5:

$$\frac{\overline{u}}{u_c}, \qquad \frac{\left(\overline{u'u'}\right)^{1/2}}{u_{c,l}}, \qquad \frac{\left(\overline{v'v'}\right)^{1/2}}{u_{c,l}}, \qquad \frac{\overline{u'v'}}{u_{c,l}^2} \tag{3.13}$$

where u_c is the mean axial velocity at the center line; $u_{c,l}$ is the the mean axial velocity at the center line, assuming idealized profiles from the fully developed region where $1/u_c$ is linear [120]. Profile abscissa are scaled assuming that the velocity

(positive mean axial component) profile follows a Gaussian distribution and using its standard deviation b, evaluated numerically as

$$b(x) = \sqrt{\frac{\int_{\mathsf{y}_{\min}}^{\mathsf{y}_{\max}} (\mathsf{y} - \mathsf{y}_c)^2 \, \max[\overline{u}(\mathsf{x}), 0] \, \mathrm{d}\mathsf{y}}{\int_{\mathsf{y}_{\min}}^{\mathsf{y}_{\max}} \max[\overline{u}(\mathsf{x}), 0] \, \mathrm{d}\mathsf{y}}} \tag{3.14}$$

where y_c is the horizontal coordinate of the center line. It may be noted that, under the Gaussian distribution assumption, the 95% of the (positive) flux is contained within $\pm 2b$. Similarly to u_c, abscissa scaling for turbulence variables in Eq. 3.13 is performed using idealized linearly increasing values of b from the fully developed region, referred to as b_l. Accordingly, $\mathbf{x}_i$ (with $i = 1, ..., H$) arrays are formed as

$$\mathbf{x}_i = \left\{ \begin{array}{cc} \overline{u}(\hat{\mathsf{y}}, \mathsf{x}_i) & [u_c(\mathsf{x}_i)]^{-1} \\ \left[\overline{u'(\hat{\mathsf{y}}, x_i) \circ u'(\hat{\mathsf{y}}, \mathsf{x}_i)}\right]^{1/2} & [u_{c,l}(\mathsf{x}_i)]^{-1} \\ \left[\overline{v'(\hat{\mathsf{y}}, \mathsf{x}_i) \circ v'(\hat{\mathsf{y}}, \mathsf{x}_i)}\right]^{1/2} & [u_{c,l}(\mathsf{x}_i)]^{-1} \\ \overline{u'(\hat{\mathsf{y}}, \mathsf{x}_i) \circ v'(\hat{\mathsf{y}}, \mathsf{x}_i)} & [u_{c,l}(\mathsf{x}_i)]^{-2} \end{array} \right\} \in \mathcal{R}^Q \tag{3.15}$$

where $\hat{\mathsf{y}}$ collects discretized scaled abscissa and '$\circ$' indicates entry-wise product. Here, Q equals four times the size of $\hat{\mathsf{y}}$ and H equals the number of cross sections considered. Variables values at positions $\hat{\mathsf{y}}$ are evaluated by linear interpolation.

Point-Wise Spectra

Spatial points are clustered together based on their energy spectra. Each point is provided with the energy spectrum [51, 10], namely $E(f)$, where f is the frequency. This is used in logarithmic scale to form clustering arrays in Eq. 3.5:

$$\mathbf{x}_i = \left\{ \log\left[E(\mathbf{f})\right] \right\} \in \mathcal{R}^Q \qquad i = 1, ..., H \tag{3.16}$$

where $\mathbf{f}$ is the vector collecting discretized frequencies, Q equals the size of $\mathbf{f}$, and $H = J$.

Point-Wise Reynolds Stress Tensor Components

Spatial points are clustered together based on their Reynolds stress tensor components. Specifically, each point is provided with Reynolds stress tensor components, which are combined together to form clustering arrays in Eq. 3.5:

$$\left(\overline{u'u'}\right)^{\beta}, \qquad \left(\overline{v'v'}\right)^{\beta}, \qquad \left|\overline{u'v'}\right|^{\beta} \tag{3.17}$$

where the exponent β is used as a tuning parameter for the clustering process, and assumed equal to $1/2$. Accordingly, clustering variables collected in $\mathbf{x}_i$ are organized

as

$$\mathbf{x}_i = \left\{ \begin{array}{c} \left[\overline{u'(\mathsf{x}_i)u'(\mathsf{x}_i)}\right]^{1/2} \\ \left[\overline{v'(\mathsf{x}_i)v'(\mathsf{x}_i)}\right]^{1/2} \\ \left|\overline{u'(\mathsf{x}_i)v'(\mathsf{x}_i)}\right|^{1/2} \end{array} \right\} \in \mathcal{R}^Q \qquad i = 1, ..., H \tag{3.18}$$

where $Q = 3$ and $H = P$.

3.4.2 Snapshot Clustering Approach

Snapshot clustering is performed for the propeller wake using vorticity snapshots following clustering variables as

$$\mathbf{x}_i = \boldsymbol{\omega}^{(i)} \qquad i = 1, ..., H \tag{3.19}$$

Data sets are organized in phase-locked snapshots where each phase is typically observed hundreds of times. In this case Q equals the size of the spatial discretization and $H = T$. The POD implementation for the propeller wake follows the same procedure, with

$$\omega = \overline{\omega} + \omega' \tag{3.20}$$

where $\omega = \partial v/\partial x - \partial u/\partial y$ is the vorticity z-component (out of plane) and x and y are the axial/horizontal and vertical coordinates respectively. The $(L \times T)$ data matrix $\mathbf{X}$ is defined in this case as

$$\mathbf{X} = \left[\begin{array}{c|c|c} \boldsymbol{\omega}^{(1)} & \ldots & \boldsymbol{\omega}^{(T)} \end{array} \right] \tag{3.21}$$

where $\boldsymbol{\omega} = \{\omega'(\hat{\mathsf{x}}_1), \ldots, \omega'(\hat{\mathsf{x}}_P)\}^\top$ collects the discretized vorticity fluctuations, x_i represents the i-th node of the spatial discretization, P is the spatial discretization size, superscript (t), with $t = 1, \ldots, T$, indicates the t-th time realization (snapshot), and finally $L = P$.

3.4.3 Data analysis Metrics

Three metrics are used for the assessment of clustering approaches and identification of the optimal number of clusters k, namely: (1) within-cluster sum of squares, (2) average silhouette, and (3) local variation of the kinetic energy. The first and the second metric is used for the propeller wake while latter is used only for buoyant jets. Their definition is included in the following.

Within-Cluster Sum of Squares

The WCSS in Eq. 3.5 is used as evaluation metrics to identify the optimal number of clusters k. Specifically, the elbow method [74] is used with the WCSS metrics.

Average Silhouette

The silhouette method provides a metrics of consistency of data within clusters [105]. Assume a_i as the average Euclidean distance between $\mathbf{x}_i$ and any other data point within the cluster $\mathbf{x}_i$ belongs to. Assume then c_i as the smallest average Euclidean distance of $\mathbf{x}_i$ to all data points in any other cluster $\mathbf{x}_i$ does not belong to. The silhouette associated to $\mathbf{x}_i$ is defined as

$$s_i = \frac{a_i - c_i}{\max[a_i, c_i]} \tag{3.22}$$

and is a measure of how similar the data point is to points in its own cluster as opposed to other clusters. It may be noted that s_i ranges from -1 to 1, where 1 indicates maximum similarity. The average silhouette of all data points is used as a metrics for proper data clustering:

$$s_{\text{avg}} = \sum_{i=1}^{T} s_i \tag{3.23}$$

Note that for $k = 1$ the silhouette is not defined. By convention, for $k = 1$ it is $s_{\text{avg}} = 0$.

Local Variation of Kinetic Energy

In order to define the number of clusters k, we use the information of the variance/energy resolved by the dominant POD mode performed for each cluster. In particular we define the local variation of kinetic energy (LVKE) as

$$\text{LVKE}_k = \sum_{i=1}^{k} \lambda_i - \sum_{j=1}^{k-1} \lambda_j \tag{3.24}$$

which measure the difference in the kinetic energy resolved by the k dominant POD modes minus the kinetic energy resolved by $k-1$ dominant POD modes. Consequently, an high value of the LVKE means that increasing the number of clusters of unit there is a great advantage in terms of variance resolved.

3.5 Numerical Results and Physical Interpretation

In the following sections we will present the numerical results and a physical interpretation for fluid dynamics experiments considered in this work.

3.5.1 Global Flow Analysis: POD and DMD

In this section the POD and the DMD techniques for the buoyant and uniform jet are discussed. In Fig. 3.2 there are the first five POD modes sorted respect to their energy content given by their relative eigenvalues. Interestingly in case of the POD the buoyant and the uniform jets show a quite different behavior in terms of spatial structure which explain most of the energy content inside the physical experiment.

In fact, the first POD mode of the buoyant jet mainly describe the laminar region while in the uniform case seems to highlight the components of the flow near the top of the wall. For the remaining modes the main difference between the two jets is that in the buoyant case the modes they seem to characterize more the reverse component of the flow (especially mode $k = 4$) while in the uniform jet most of the energetic content of the modes is shown always in the same region. In Fig. 3.3 is showed the convergence of the explained variance explained from each POD mode for buoyant and uniform jet. It is possible to observe that the POD in general requires an high number of modes in order to explain a relevant amount of energy inside the system which is mainly due to the fact that high nonlinearities are strongly dominant in both cases.

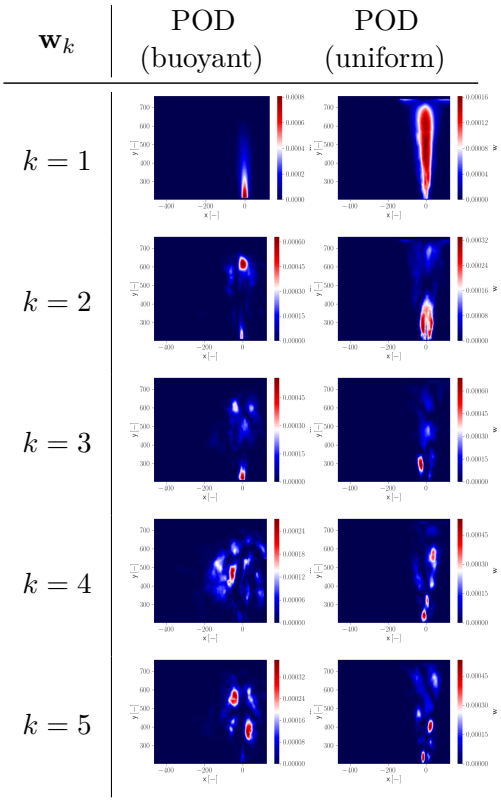


Figure 3.2. Top five most energetic content POD modes for buoyant (left column) and uniform (right column) jets.

For the DMD methodology, identify the modes with the highest energetic content is non trivial as in the POD case, because the eigenvalues relative to the DMD modes don't explain the kinetic energy content but they describe their temporal dynamics. The spectrum of the DMD eigenvalues for the buoyant and the uniform is shown in Fig. 3.4. In this work, in order to provide a sorting

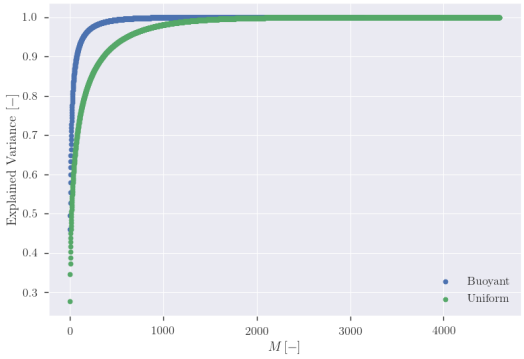
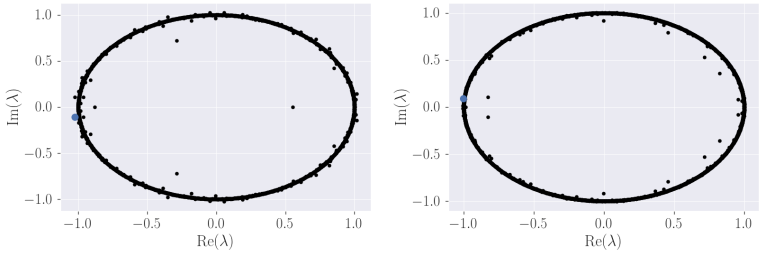


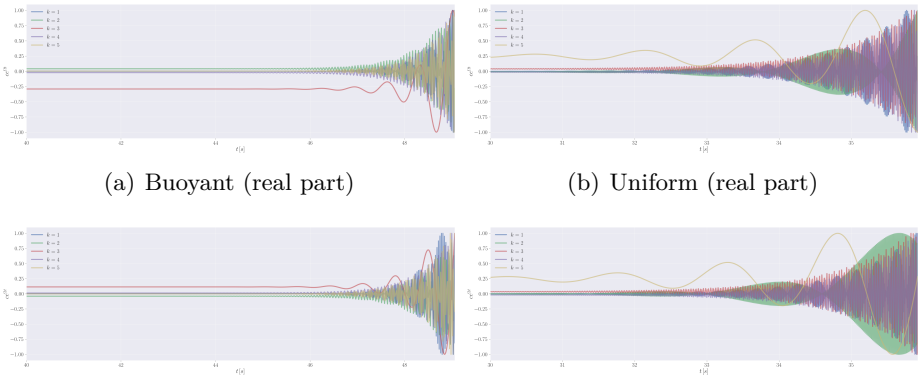
Figure 3.3. Explained variance convergence for the uniform and buoyant jet.



(a) Buoyant (b) Uniform

Figure 3.4. Spectrum DMD eigenvalues.

respect to energetic content of the DMD modes we follow the method proposed in [140] where the DMD modes are scaled by their eigenvalues raised to their $n-1$ power. Then the top five DMD modes respect to their energy content are shown in Fig. 3.6 for the real part and the imaginary part. From the DMD modes, we can notice that for the buoyant jet, the real part of the modes highlight the entry region of the flow while for the uniform jet, the modes span all the y-axis. This is less visible for the imaginary part of the DMD mode where the flow is also highlighted. More physical insights can be investigated by computing the temporal dynamics of



(a) Buoyant (real part) (b) Uniform (real part)

(c) Buoyant (imaginary part) (d) Uniform (imaginary part)

Figure 3.5. DMD Time dynamics.

those DMD modes, as shown in Fig. 3.5 for the real and imaginary part respectively. The time dynamics relative to the mode i is defined by $c_i e^{\omega_i t}$ with ω the continuous DMD eigenvalue is given by the relation $\log(\lambda_i)/\Delta t$.

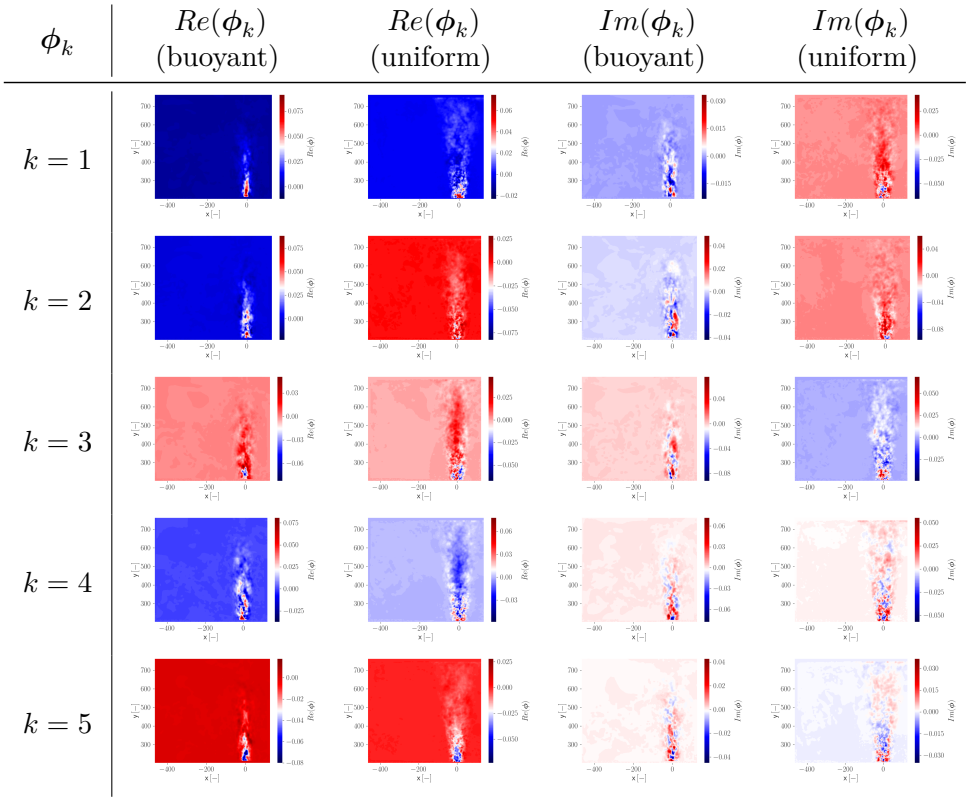
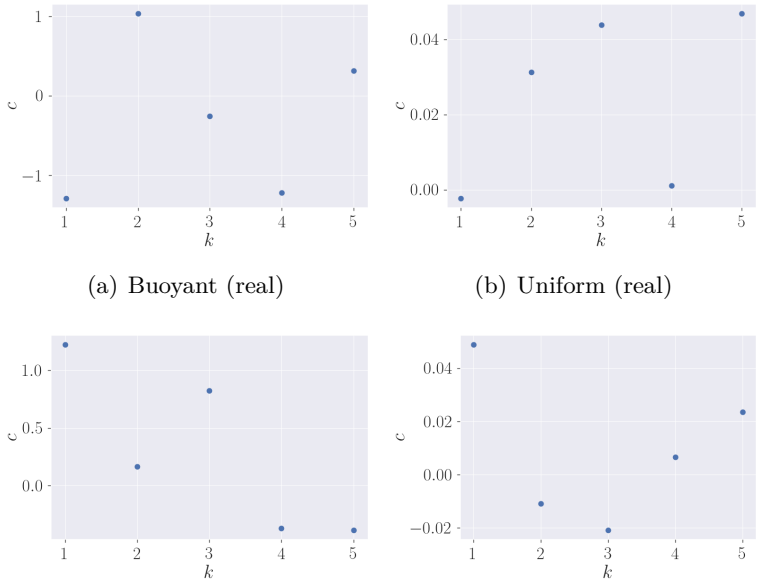


Figure 3.6. Top five energy content DMD modes for buoyant (left column) and uniform (right column) jets (imaginary part).



(a) Buoyant (real)

(b) Uniform (real)

(c) Buoyant (imaginary)

(d) Uniform (imaginary)

Figure 3.7. DMD Coefficients.

3.5.2 Spatial Clustering Results

The whole field of view is discretized with 197 × 187 points. Data rates of 128 and 64 Hz are used for the uniform and buoyant jet respectively. A total of 4,600 snapshots are used for the uniform jet, whereas 3,141 are used for the buoyant jet. For the purpose of current study, a spatial subsampling by a factor of 2 along each spatial direction is used. Coordinates x and y are relative to the jet virtual origin and reported as ratio with respect to the nozzle diameter D.

Clustering by Cross-Section Velocity Profiles

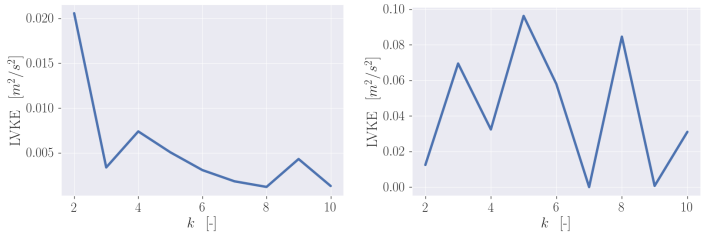
Clustering of velocity profiles produces the LVKE metric presented in Fig. 3.8 for the uniform and buoyant jet. Overall, two clusters may be identified for the uniform jet since the LVKE has a reasonably large value for $k = 2$. For the buoyant jet the situation is totally different, here the maximum of the LVKE is for $k = 5$.

Figure 3.9 shows the uniform-jet cluster centroids for the mean axial velocity (a) and the profiles of Reynolds stress quantities (b–d). The corresponding cross-section clustering is presented in Figure 3.9e. Self-similarity of mean axial velocity profiles is reflected in the centroids (e). Significant differences emerge for the Reynolds stress, where the uu component (normal/axial) presents noticeable differences between the bottom and the top cluster (c), suggesting the turbulence fully develops transitioning from the bottom to the top cluster. The vv component (normal/horizontal) has almost identical profiles for the bottom and middle clusters, whereas it presents a bi-modal shape for the top cluster where impingement occurs (d).

Finally the uv component (shear) produces cluster centroids with significant larger values for the top cluster (e). We may conclude that two regions are identified via k-means clustering, namely bottom (developing), and middle-top (fully developed and impingement) regions. Fig. 3.10 shows the buoyant-jet cluster centroids for the mean axial velocity (a) and the profiles of Reynolds stress quantities (b–d). The corresponding cross-section clustering is presented in Figure 3.10e. Self-similarity of mean axial velocity profiles can be noticed only for the first three clusters (from bottom to top), whereas the top two clusters present not only a remarkable reverse flow but also a different scaling properties with respect to b (a), which may be due to the presence of instabilities and very low velocity values. The uu component (normal/axial) presents noticeable differences between the bottom/middle clusters and the top three clusters (c). The vv component (normal/horizontal) has similar profiles for the bottom and middle clusters, whereas it presents a mild bi-modal shape for the top clusters, which also show reduced values (d). Finally the uv component (shear) produces cluster centroids with very small values for the two top clusters (e). We may conclude that five regions are identified via k-means clustering, namely first from bottom (developed, with almost no reverse flow), second (developed, with significant reverse flow and associated shear stress), third (flow reversing), fourth (jet dome), fifth (top wall) regions.

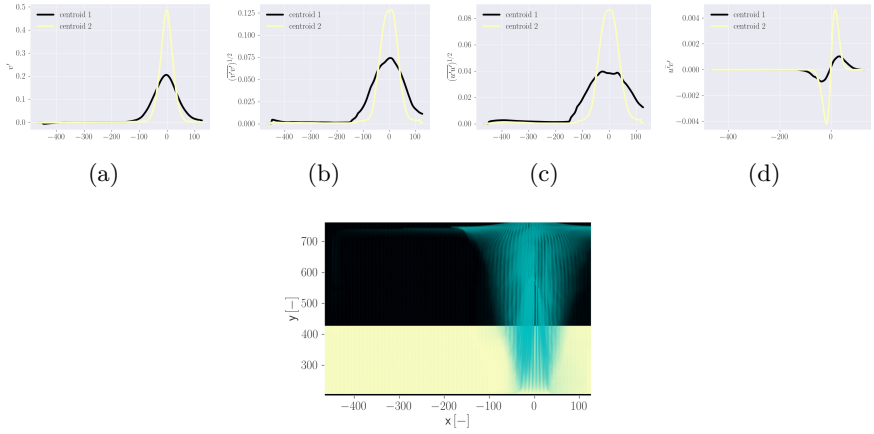
Clustering by Point-Wise Energy Spectra

Clustering by point-wise energy spectra produces the metrics presented in Fig. 3.11 for the uniform and buoyant jet respectively. The number of clusters selected for the



(a) Uniform (b) Buoyant

Figure 3.8. Clustering by cross-section velocity profiles, LVKE uniform (a) LVKE buoyant (b).



(a) (b) (c) (d)

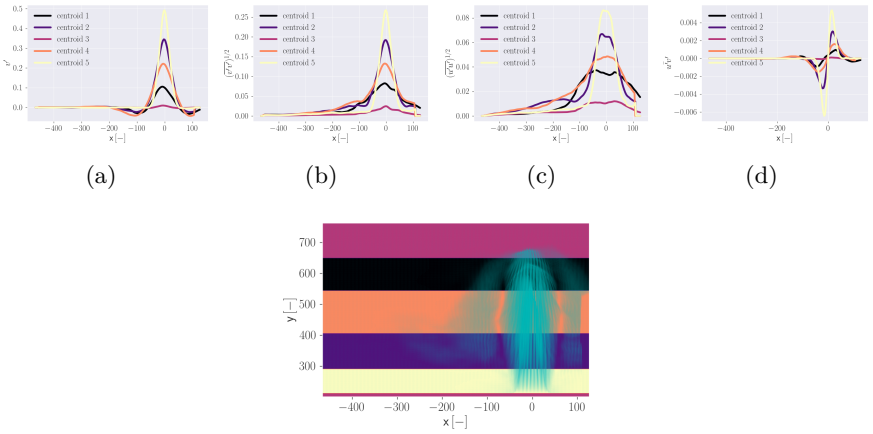
(e)

Figure 3.9. Clustering by cross-section velocity profiles – Uniform jet – Cluster centroids for scaled mean velocity (a) Reynolds stress quantities (b,c,d); cross sections labeled by cluster (e).

uniform jet is 2 since for $k = 2$ (a) the LVKE shows its maximum value. Differently, the number of clusters for the buoyant jet is 4 since he LVKE shows its maximum value at $k = 4$.
Clustering results also emphasizes the differences between uniform and buoyant jets, where the former experiences a reverse flow with associated shear layer due to impingement with the top and left walls, whereas for the latest the reverse flow and shear layer are closer to the jet axis.

Clustering by Point-Wise Reynolds Stress Tensor Components

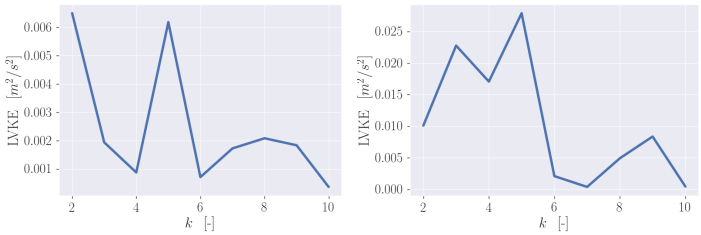
Clustering by point-wise Reynolds stress tensor components gives the metrics presented in Fig. 3.13 for the uniform and buoyant jet.
A number of clusters equal to 3 is selected for the uniform jet, while a number of 5 clusters is selected for the buoyant jet. The spatial decomposition of the domain associated to each cluster are shown in Figure 3.14. It may be observed how the clustering method defines spatial regions mainly by turbulence intensity, highlighting



(a) (b) (c) (d)

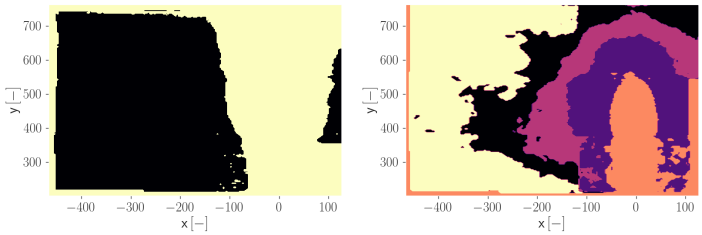
(e)

Figure 3.10. Clustering by cross-section velocity profiles – Buoyant jet – Cluster centroids for scaled mean velocity (a) Reynolds stress quantities (b,c,d); cross sections labeled by cluster (e).



(a) Uniform (b) Buoyant

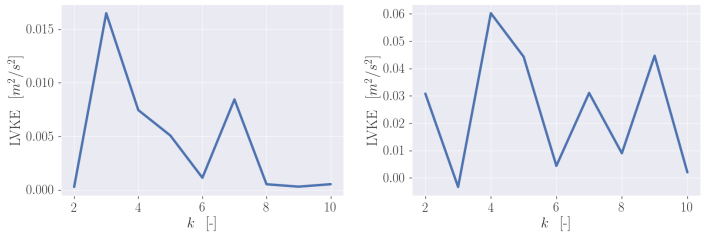
Figure 3.11. Clustering by point-wise energy spectra, LVKE uniform (a) LVKE buoyant (b).



(a) Uniform (b) Buoyant

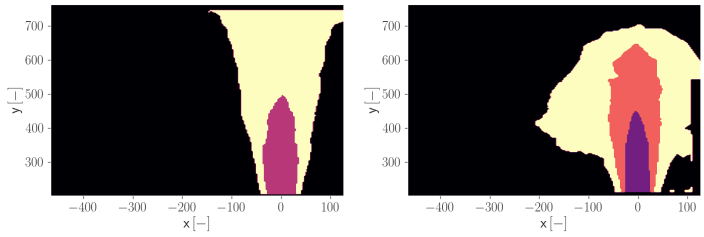
Figure 3.12. Clustering by point-wise energy spectra – Spatial points labeled by cluster for uniform (a) and buoyant (b) jets.

the differences between the two jets.



(a) Uniform (b) Buoyant

Figure 3.13. point-wise Reynolds stress tensor components, LVKE uniform (a) LVKE buoyant (b).



(a) Uniform (b) Buoyant

Figure 3.14. Clustering by point-wise Reynolds stress tensor components – Spatial points labeled by cluster for transient uniform (a) and buoyant (b) jets.

3.5.3 Snapshot Clustering: Propeller Wake Results

4-Bladed Propeller

Four phase-locked vorticity data sets (0, 90, 180, and 270 deg) are used, where each phase is observed 250 times for a total of 1,000 snapshots. The snapshots are organized (subsampled) in a 200×30 array, ranging axially from 0 to 3.3 D, and radially from 0.3 to 0.8 D, focusing on the tip vortex only (see Fig. 3.15). The data matrix has a dimension equal to 6,000 × 1,000. It may be noted that once the data matrix is formed, the information on the phase is lost (as this information is not included in the data matrix).

First, the k-means is applied to the whole field of view (see Fig. 3.16). Fig. 3.17 shows that 4 clusters emerge from the data set. Specifically, WCSS shows a clear elbow corresponding to $k = 4$. The average silhouette exhibits a clear maximum corresponding to $k = 4$. Fig. 3.18a shows the projection of the data set onto the first two POD/PCA modes. Data is labeled both by cluster and phase, showing that the method is able to recover phase information and the data set is clearly clustered by phase. A similar analysis and visualization is shown using t-SNE in Fig. 3.18b, confirming the POD/PCA result. Finally, Fig.3.18c provides joint and marginal probability density functions of POD/PCA and t-SNE coefficients $\boldsymbol{\alpha}$ given by KDE, confirming the data has four clusters of equal size. The corresponding cluster centroids are presented in Fig. 3.19, showing that the mechanism of vortex coupling and convection downstream is globally (for the large scale) deterministic

depending mainly on the phase.

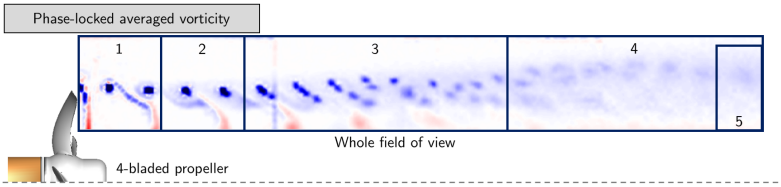


Figure 3.15. 4-bladed propeller – Windows used for clustering of vorticity snapshots.

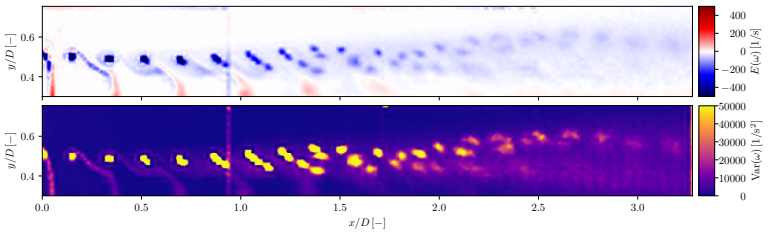


Figure 3.16. Vorticity mean value (top) and variance (bottom) for the 4-bladed propeller.

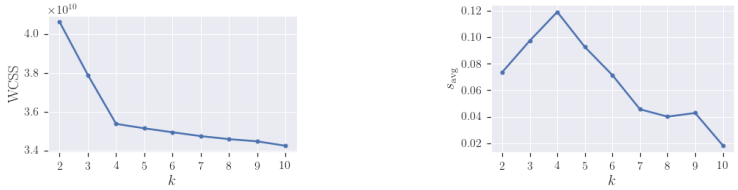
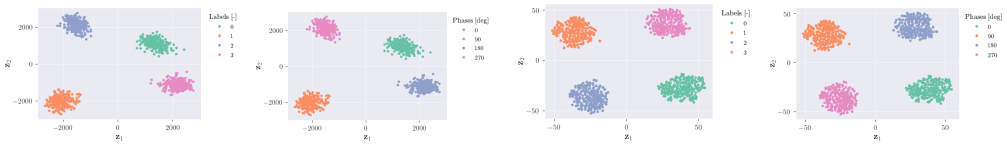


Figure 3.17. Clustering of vorticity snapshots – 4-bladed propeller, whole field of view – Within cluster sum of squares (left) and average silhouette (right).

A second analysis is performed, dividing the filed of view in several windows, based on the vorticity mean and variance associated to each cross section. Fig. 3.20 shows the maximum mean and variance of cross sections along the propeller axis. Four windows are selected, as shown in Fig. 3.15: (1) $0 \leq x/D < 0.4$, where the maximum variance is low and the wake is stable; (2) $0.4 \leq x/D < 0.8$, where the maximum variance starts increasing and the wake destabilizing; (3) $0.8 \leq x/D < 2$, where the maximum variance reaches its own maximum and starts decreasing along with the maximum mean and the wake experiences a fully developed tip vortex interaction; (4) $2 \leq x/D \leq 3.3$ where variance and mean are almost constant and a fully turbulent wake is observed.

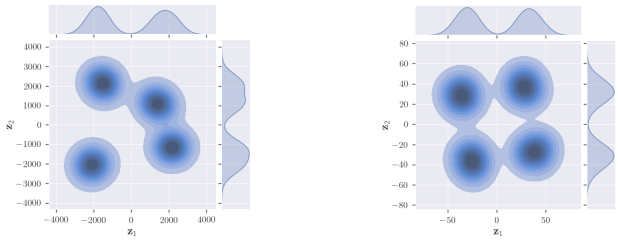
Fig. 3.21, first column, shows the WCSS and silhouette for window 1. Fig. 3.22, first column, shows the POD/PCA and t-SNE coefficients labeled by cluster and phase for the same window. Fig. 3.23, fist column, shows the density functions of the coefficients. As expected, no significant structures are observed. The t-SNE highlights some structure and hints of data clustering. Nevertheless, these are not significant and the data can be interpreted as a single cluster.

Similarly, the second column of Fig. 3.21, 3.22, and 3.23 provides the results for window 2. Clustering results are ambiguous since 4 and 3 clusters are identified by



(a) POD/PCA labeled by cluster (left) and phase (right)

(b) t-SNE labeled by cluster (left) and phase (right)



(c) KDE for POD/PCA (left) and t-SNE (right) coefficients

Figure 3.18. Clustering of vorticity snapshots – 4-bladed propeller, whole field of view – Data-projection on the first two POD/PCA modes (a) and embedding via t-SNE (b), along with joint and marginal probability density functions by KDE (c).

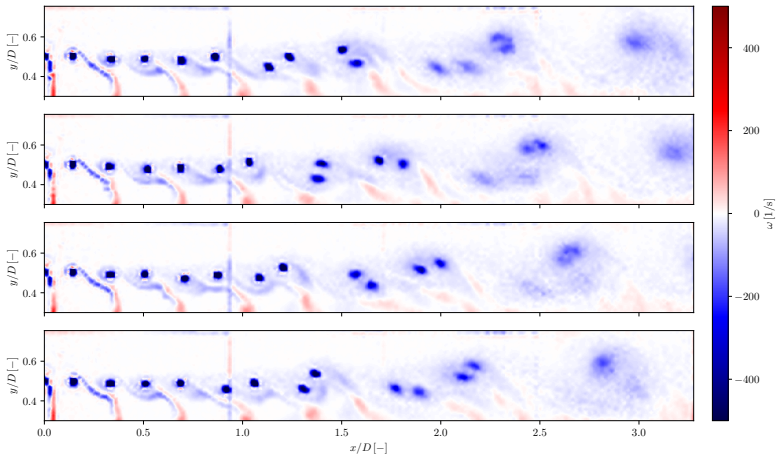


Figure 3.19. Clustering of vorticity snapshots – 4-bladed propeller, whole field of view – Cluster centroids corresponding to phases 0, 90, 180, 270 deg.

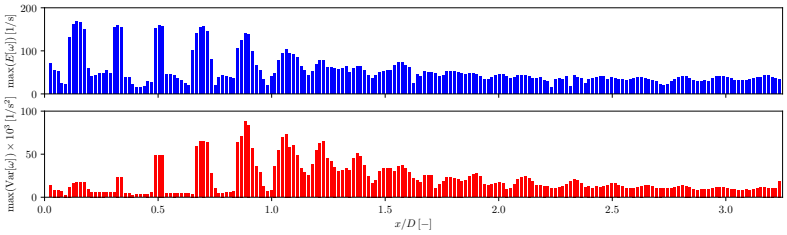


Figure 3.20. 4-bladed propeller, whole field of view – Maximum value of mean (top) and variance (bottom) of the vorticity along cross sections.

WCSS and silhouette, respectively. POD/PCA coefficients show 2 or 3 main clusters whereas the t-SNE clearly identifies 4 clusters associated to the propeller phases.

Results for window 3 and 4 are presented in the third and fourth column of Fig. 3.21, 3.22, and 3.23, respectively where 4 clusters are clearly identified with a one-by-one association to the phase. It may be noted how an high degree of determinism is still present far downstream the propeller. It may be also noted how window 4 t-SNE analysis presents some hints of transition towards a different clustering structure.

Fig. 3.24 shows the cluster centroids associated to windows 2, 3, and 4, where rows represent clusters and columns represent windows. As discussed earlier, these centroids also represent phase-locked averages. The unsupervised association of clusters to phases by k-means indicates that the destabilization (and coupling) of tip vortices progresses following mechanisms governed by deterministic chaos. Finally, the same analysis is performed for window 5, which bounds more closely a single vortex (see Fig. 3.15). Clustering results are ambiguous in this region (see Fig. 3.25), even if some patterns are identified by both POD/PCA and t-SNE coefficients, where a pairwise mixture of phases $0 - 90$ and $180 - 270$ is present (see Fig. 3.26). Cluster centroids are shown in Fig. 3.27.

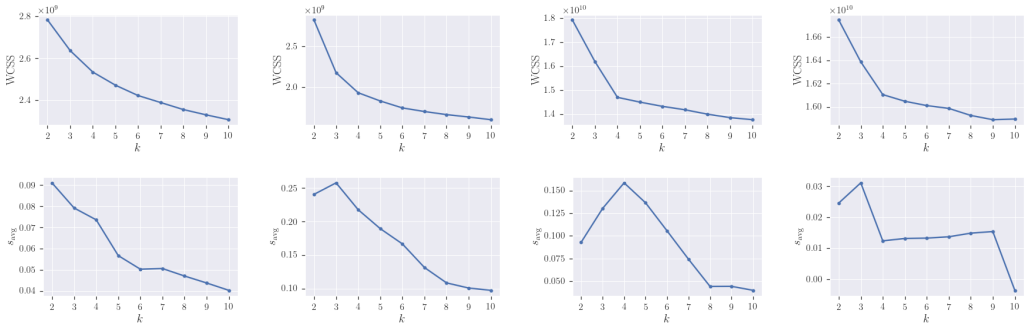
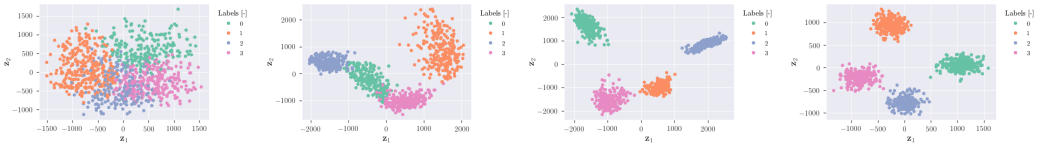


Figure 3.21. Clustering of vorticity snapshots – 4-bladed propeller – Within cluster sum of squares (top) and average silhouette (bottom). From left to right: windows 1, 2, 3, and 4.

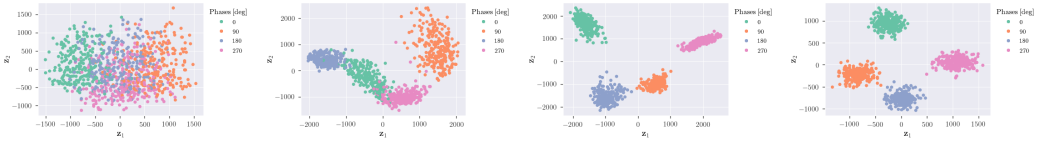
7-Bladed Propeller

A similar analysis is performed for the 7-bladed propeller. In this case, the data set is composed by 500 snapshots coming from a single phase (0 deg). The snapshots are organized (subsampled) in a 200×50 array, ranging axially from 0 to 3.3 D, and radially from 0 to 0.8 D (Fig. 3.28). The data matrix has a dimension equal to $10{,}000 \times 500$.

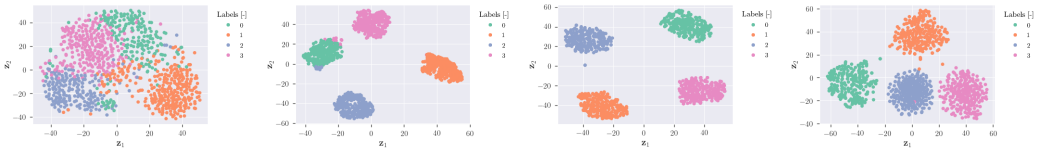
Applying the clustering method to the whole field of view does not reveal any clusters, as shown by WCSS, silhouette (see Fig. 3.29), and POD/PCA and t-SNE coefficients (Fig. 3.30). This is due to the fact that the data set is composed by one phase only and it is consistent with what we found for the 4-bladed propeller, i.e., that overall the data clustering follows the phase. Also window 1 (bounding one vortex) does not show hints of clustering as Fig. 3.31 and 3.32 show.



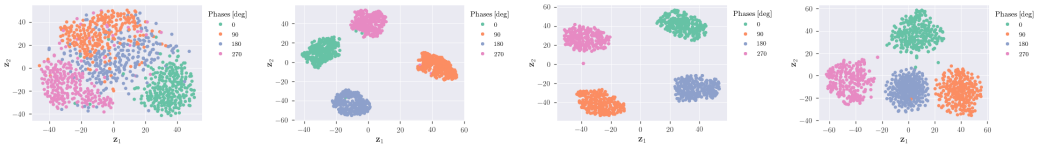
(a) POD/PCA labeled by cluster



(b) POD/PCA labeled by phase

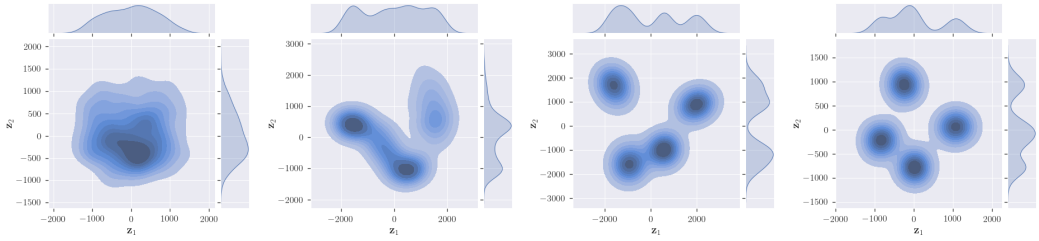


(c) t-SNE labeled by cluster

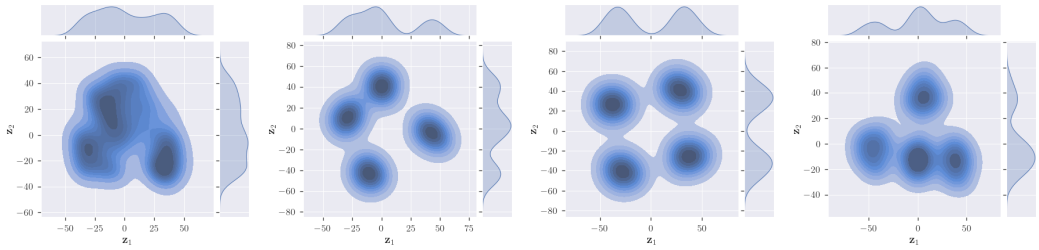


(d) t-SNE labeled by phase

Figure 3.22. Clustering of vorticity snapshots – 4-bladed propeller – Data-projection on the first two POD/PCA modes (a,b) and embedding via t-SNE (c,d). From left to right: windows from 1 to 4.



(a) POD/PCA coefficients



(b) t-SNE coefficients

Figure 3.23. Clustering of vorticity snapshots – 4-bladed propeller – Joint and marginal probability density functions of POD/PCA (a) and t-SNE (b) coefficients by KDE. From left to right: windows from 1 to 4.

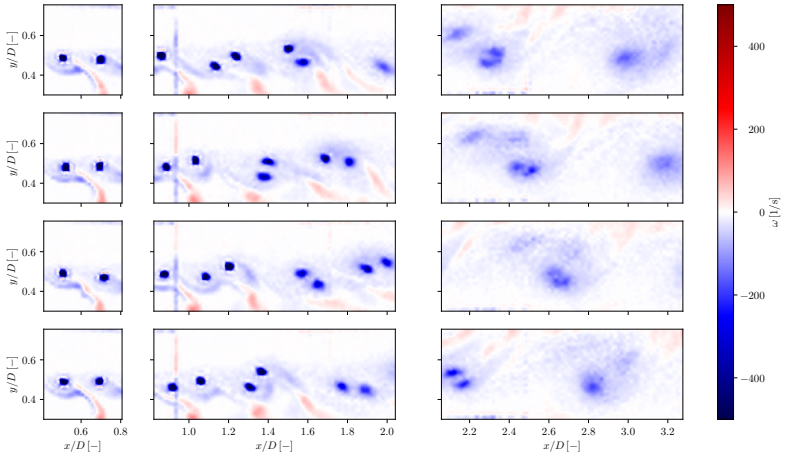


Figure 3.24. Clustering of vorticity snapshots – 4-bladed propeller – Cluster centroids corresponding to phases 0, 90, 180, 270 deg. From left to right: windows 2, 3, and 4.

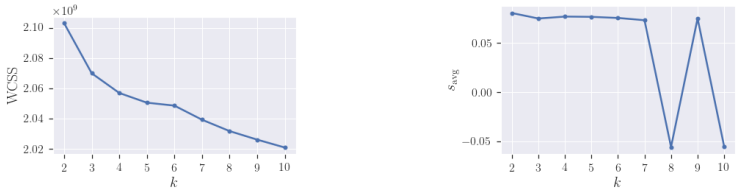
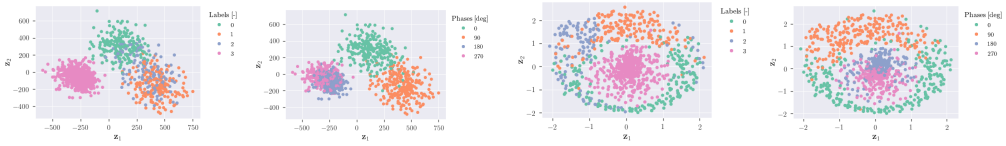
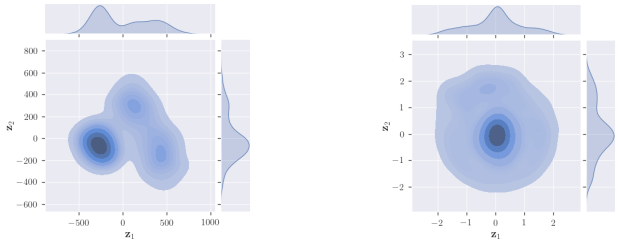


Figure 3.25. Clustering of vorticity snapshots – 4-bladed propeller, window 5 – Within cluster sum of squares (left) and average silhouette (right).



(a) POD/PCA labeled by cluster (left) and phase (right)

(b) t-SNE labeled by cluster (left) and phase (right)



(c) KDE for POD/PCA (left) and t-SNE (right) coefficients

Figure 3.26. Clustering of vorticity snapshots – 4-bladed propeller, window 5 – Data-projection on the first two POD/PCA modes (a) and embedding via t-SNE (b), along with joint and marginal probability density functions by KDE (c).

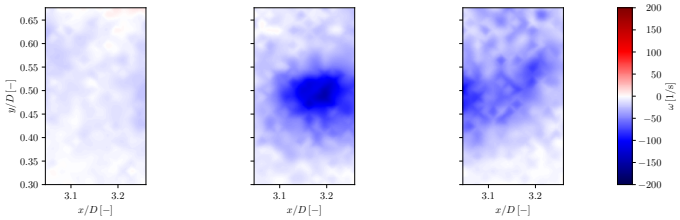


Figure 3.27. Clustering of vorticity snapshots – 4-bladed propeller, window 5 – Cluster centroids.

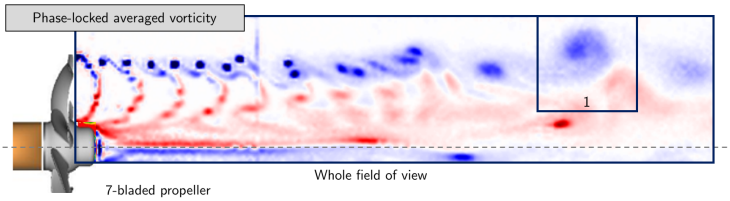


Figure 3.28. 7-bladed propeller – Windows used for clustering of vorticity snapshots.

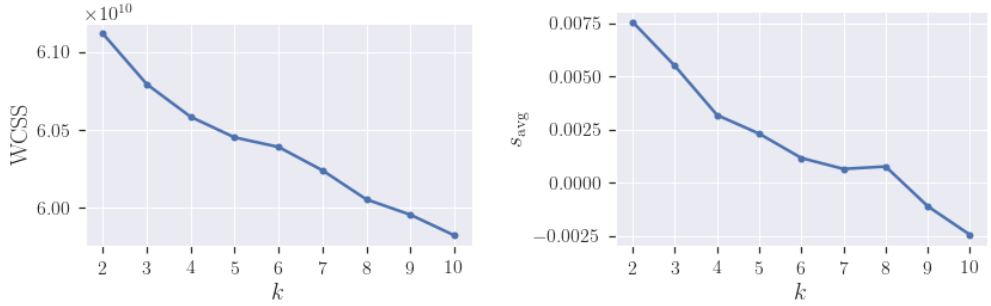
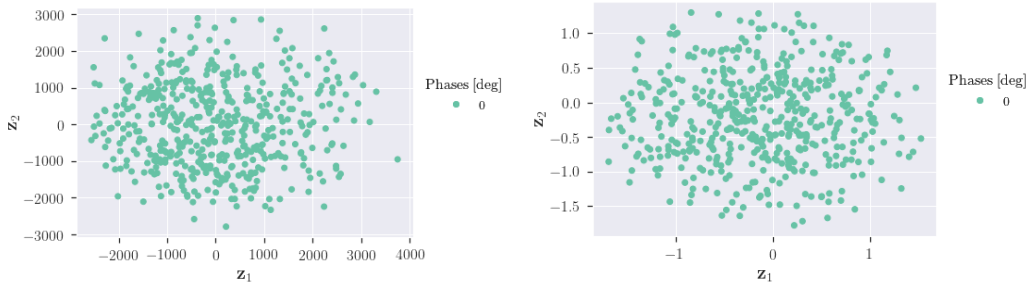
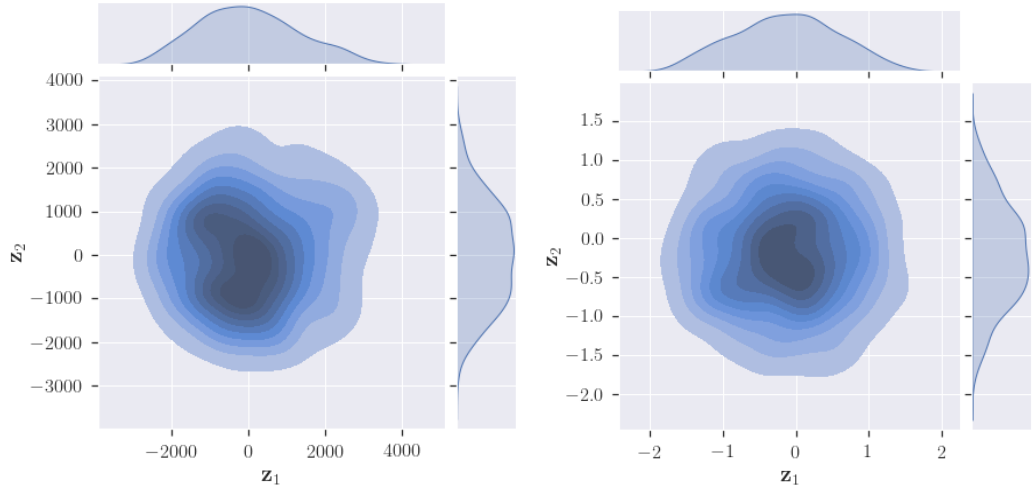


Figure 3.29. Clustering of vorticity snapshots – 7-bladed propeller, whole field of view – Within cluster sum of squares (left) and average silhouette (right).



(a) POD/PCA (b) *t*-SNE



(c) KDE for POD/PCA (left) and *t*-SNE (right) coefficients

Figure 3.30. Clustering of vorticity snapshots – 7-bladed propeller, whole field of view – Data-projection on the first two POD/PCA modes (a) and embedding via *t*-SNE (b), along with joint and marginal probability density functions by KDE (c).

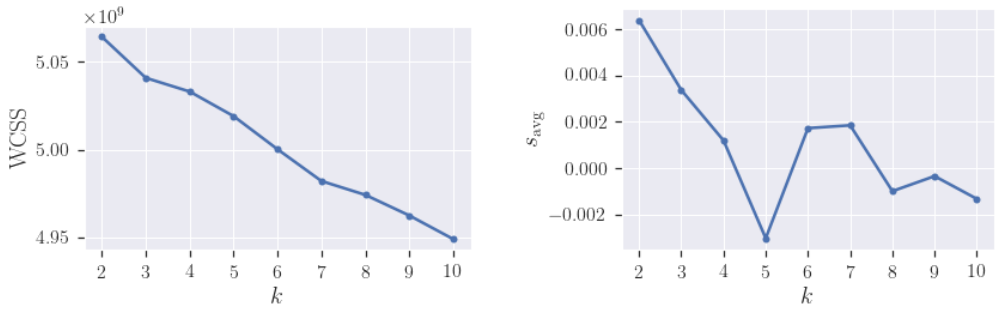
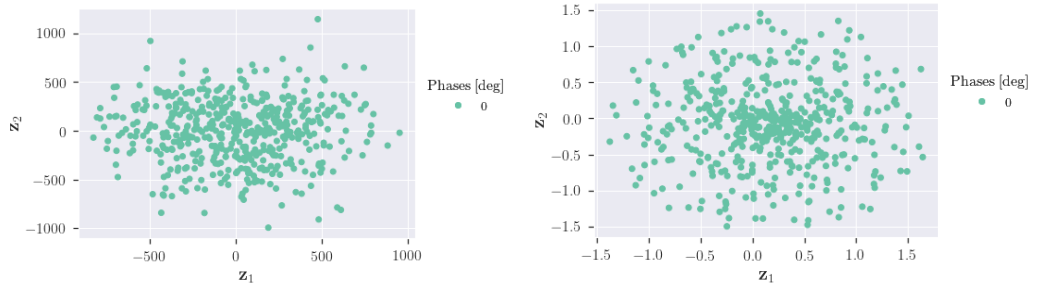
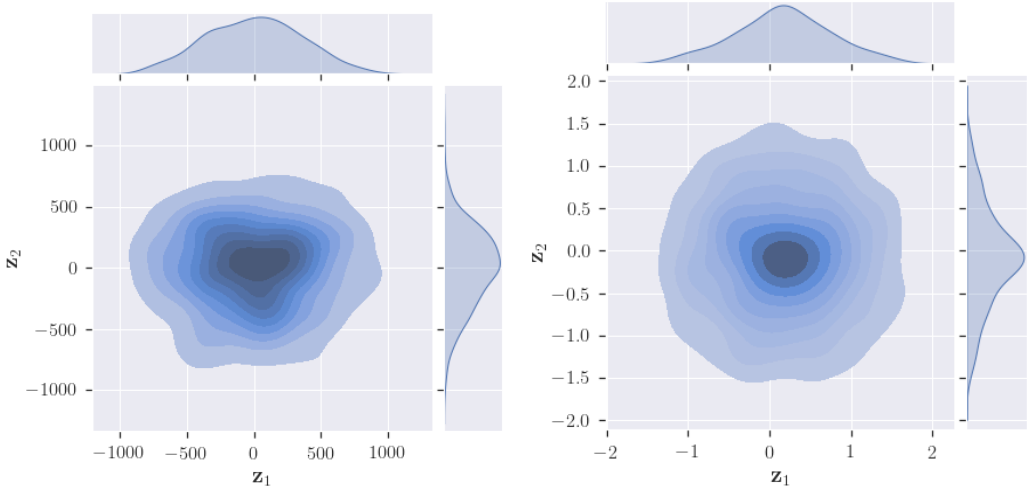


Figure 3.31. Clustering of vorticity snapshots – 7-bladed propeller, window 1 – Within cluster sum of squares (left) and average silhouette (right).



(a) POD/PCA (b) *t*-SNE



(c) KDE for POD/PCA (left) and *t*-SNE (right) coefficients

Figure 3.32. Clustering of vorticity snapshots – 7-bladed propeller, window 1 – Data-projection on the first two POD/PCA modes (a) and embedding via *t*-SNE (b), along with joint and marginal probability density functions by KDE (c).

3.6 Conclusions and future work

A global flow analysis, spatial and snapshot clustering approaches have been presented and discussed for PIV data of high-Reynolds number uniform and buoyant jets and 4- and 7-bladed propeller wakes respectively. Data clustering was based on the k-means algorithm. In order to define the number of clusters, we proposed a metric which measure the energy resolved respect to a particular clustering configuration. Spatial clustering for jet flows was based on three sets of clustering variables, namely the cross-section velocity profiles, point-wise energy spectra, and point-wise Reynolds stress tensor components. Snapshot clustering of phase-locked propellers wake data was based on the vorticity field with focus on the tip vortices. POD/PCA and t-SNE embedding along with KDE were used to provide a two-dimensional visualization of data clusters for assessment and discussion.

Clustering of jet cross-section velocity profiles helped identifying uniform and buoyant jet zones. The analysis of clustering variables allowed to propose new self-similarity laws for the jet, based on (i) actual center velocity and jet width for the velocity profiles and (ii) their idealized linear representation for Reynolds-stress quantities. Two zones were identified for the uniform jet flow, namely bottom (developing) and middle-top (fully developed and impingement) regions. Five zones where identified for the buoyant jet, namely first from bottom (developed, with almost no reverse flow), second (developed, with significant reverse flow and associated shear stress), third (flow reversing), fourth (jet dome), fifth (top wall) regions. Clustering results by point-wise energy spectra emphasized the differences between uniform and buoyant jets, with significantly different clusters topologies. Finally, clustering by point-wise Reynolds stress tensor components produced spatial zones driven mainly by the turbulence intensity.

In conclusion, for a effective characterization, the buoyant jet needs a number of clusters that is greater respect to the uniform flow.

4-bladed propeller wake clustering of phase-locked snapshots produced no clusters (meaning only one cluster) for the near-field data window. Clusters with a one-to-one association to the phase were found for other data windows. Specifically, this was clearly observed for the whole field of view as well as for windows covering the region where the wake transitions to a unstable regimes and windows in the far field. Clustering results suggested that the wake instability and subsequent progression of tip vortices is characterized by mechanisms governed by deterministic chaos also in the far field. Results were confirmed by clustering of a single phase from the 7-bladed propeller data sets, where only a mild clusterization was found.

Ongoing and future work includes extending the analysis of clustering methods and results covering both spatial and temporal clustering for all jet and propeller wake cases with comparison and discussion of the results. The idea is to combine spatial/temporal clustering to fully exploits data reduction and visualization techniques to provide physical characterization of zones and intervals in space and time domain respectively.

Chapter 4

Variational Recurrent-Type Deep Neural Networks for Ship Motion Prediction

Time series forecasting problems arise in many applications. In Fluid Dynamics, the prediction of the dynamics of the ship motion, especially at high sea state level is severely challenging due to the high non-linearities present inside the system. In this chapter, we propose to tackle this hard real-world problem through Deep Learning. We briefly discuss the basic theory of Neural Networks (NN's), from gradient computation and optimization to regularization and uncertainty estimation which is crucial in time series problems. Then we describe Recurrent-type NN's and Encoder-Decoder architectures for sequence modeling and multi-step ahead forecasting. Performance is assessed and compared on a data set formed by computational fluid dynamics simulations of a self-propelled destroyer-type vessel in stern-quartering sea state 7. Incident wave, ship motions, rudder angle, as well as immersion probes time series, are the variables used for multiple time series now-casting problems. The objective is to obtain about 20 seconds ahead prediction.

4.1 Introduction

The prediction of the seakeeping and maneuverability performance of naval ships constitutes one of the most challenging problems in naval hydrodynamics and is important from both an operational and safety point of views, specially in heavy weather conditions. Heavy weather seakeeping of naval ships has traditionally been investigated by means of scale model testing in large seakeeping basins. From a safety point of view a large number of conditions needs to be investigated. Furthermore, in order to reduce the statistical uncertainty in the results a large number of wave encounters has to be met during the tests, including so-called rare events. This makes scale model testing time consuming and expensive. During the last decades low- to high-fidelity simulation methods have been developed for investigating ships seakeeping and maneuvering. Nevertheless, a complete solution of the seakeeping and maneuverability problem involves resolving complex nonlinear wave-body interactions that may require hundreds of computational CPU hours, especially if statistical

indicators are sought after. For this reasons, to alleviate the computational burden associated with numerical simulations, machine learning methods, such as neural networks (NNs) could provide decision support to captains in choosing route, heading, and speed, contributing to the safety of vessels, cargo, and crews, by using both historical and computational fluid dynamic (CFD) data, up to real-time data.

Classical NNs treat each observation or data point in the same way. This means that the NN does not take into account the correlation across the data points, assuming that they are independent and identically distributed (*i.i.d.*). Nevertheless, in several application, such as in time series fore- and nowcasting, the value of the target variable is usually strongly correlated to the past values of the target variable at the previous time step. This correlation is lost in a classical NN model. In order to solve this limitation, recurrent NNs (RNNs) have been developed with the objective to learn the dependencies of the data across time and to improve the prediction accuracy in case of sequential data. Nevertheless, RNN model presents some difficulties in the computation of the gradients. To overcame this issue, different mathematical models have been developed creating *gates* along the time steps where the derivatives information could flow without numerical issues. Among them the long-short term memory (LSTM)[61] and the gated recurrent unit (GRU) [19] have shown quite effective performance for modeling sequences in several research fields.

In the ship hydrodynamics context, the development and the assessment of machine learning methods in fore- and nowcasting of ship motions and (possibly) loads have become of certain interest and a cutting-edge topic in the ocean engineering community. Short-term prediction based on radial basis NN has been presented in [28]. LSTM and GRU have been investigated for the prediction of 2 and 3 degrees of freedom (DoF) of a catamaran in sea state 1 and the DTMB model in sea state 8, based on CFD computations in [29].

The objective of the present work is to assess the sequence modeling capability of recurrent-type NNs for real-time short-term prediction (nowcasting) of ship motions in high sea state. Specifically the performance of RNN, LSTM, and GRU models are assessed for the nowcasting of a self-propelled destroyer-type vessel, sailing in stern-quartering sea state 7.

The data set is formed by free-running CFD simulations of a destroyer-type vessel with appendages (skeg, twin split bilge keels, twin rudders and rudder seats slanted outwards, shafts, and struts), that have been assessed for course keeping in irregular stern-quartering waves (sea state 7) at target Froude number equal to 0.33 in [143]. RNN, LSTM, and GRU are assessed and compared in predicting wave elevation, ship motions, rudder angle, and immersion probes time histories.

4.2 A Brief Introduction to Deep Neural Networks

Neural Networks are a very popular Machine Learning models used for long time in many tasks (e.g., classification, regression) achieving generally good results.

In the last 15 years, with the rising of the computer computational power and the amount of data available, many improvements have been made by researchers. Nowadays NN's became one of the most powerful tool in Machine Learning.

In this section we highlight briefly the main components of Neural Networks.

4.2.1 Model Definition

The most simple architecture for a Neural Network is given by an input layer composed by D neurons (or nodes) and an output layer composed by one node. The computation is performed at the output node, where a linear combination of the data input vector $\mathbf{x} \in \mathcal{R}^D$, the parameters $w_1, \ldots, w_d$ and a bias scalar value w_0 is performed, followed by the application of a nonlinear activation function $h(\cdot)$

$$f(\mathbf{w}, \mathbf{x}) = h\Big(\sum_{i=1}^{D} x_i w_i + w_0\Big) \tag{4.1}$$

this architecture is called Perceptron, one of the first learning machines, developed by Frank Rosenblatt [104].

An example of a more common (and powerful) architecture is given by a Neural Network with an input layer followed by an hidden layer and an output layer which (called Multilayer Perceptron). Fixing the number of hidden units to M and the number of output units to K, the first computation is performed at the hidden layer and the output given by

$$z_j = h_1\Big(\sum_{i=1}^{D} w_{ij} x_i\Big) \quad \forall j = 1, \ldots M \tag{4.2}$$

we call the output of the hidden layer $z_1, \ldots, z_M$ hidden variables or latent variables. The output is given by

$$f_k(\mathbf{x}, \mathbf{w}) = h_2\Big(\sum_{j=1}^{M} w_{kj} z_{kj}\Big) \quad \forall k = 1, \ldots K \tag{4.3}$$

by expliciting the latent variables z_j we can highlight that the final functional form is given by chaining activation functions

$$f_k(\mathbf{x}, \mathbf{w}) = h_2\Big(\sum_{j=1}^{M} w_{jk} h_1\Big(\sum_{i=1}^{D} w_{ij} x_i\Big)\Big) \quad \forall k = 1, \ldots K \quad \forall j = 1, \ldots M \tag{4.4}$$

since in the coming chapters we'll focus on time series forecasting application, we always suppose that the activation function applied to the output variables is linear. Notice that, in this example since we have K outputs, so that this architecture could be used for multi-output regression problems. The possibility to add many hidden layers, allows to increase the representation power of the network, allowing the model to learn more complex concepts. This is simply due to the fact the network's output is given by a repeated composition of nonlinear functions.

NN's are also referred as universal function approximators, since a Multilayer Perceptron with linear outputs and a nonlinear activation can approximate any function, providing that the number of neurons in the hidden layer are sufficiently large [63]. But in practice, adding many neurons could leads the network to overfit (i.e low error on observed data and large error on unseen data). A possible solution to this issue is that we could add layers with limited width (i.e. number of neurons) in order to increase the capacity of the network without increasing much the number

of parameters. This is one of the main underlying motivation of the power of Deep Neural Networks [1]. In the next section we introduce the main ingredients used for train NN's, namely the definition of the loss function and how the gradient, respect to the network parameters can be computed in an efficient way.

4.2.2 The Error Backpropagation Algorithm

Given a dataset of input matrix $\mathbf{X} = \{\mathbf{x}\}_{n=1}^{N}$ with the random variable $\mathbf{x} \in \mathcal{R}^D$ and a target matrix $\mathbf{Y} = \{\mathbf{y}\}_{n=1}^{N}$ with the output vector is $\mathbf{y} \in \mathcal{R}^K$, we are interested to learn a function $\mathbf{f}(\mathbf{x}, \mathbf{w})$ such that the squared loss is minimized

$$e(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n, \mathbf{w})||^2 \tag{4.5}$$

From a probabilistic point of view, the squared error objective function can be obtained also by minimizing the negative log-likelihood of the conditional distribution of the output variables supposing that $p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{f}(\mathbf{x}, \mathbf{w}), \sigma^2\mathbf{I})$ follows a Gaussian distribution.

In regression setting we average the squared loss in Eq. 4.5 across all data points and we minimize the *empirical risk*, obtaining the Mean Squared Error (MSE).

The presence of nonlinear activation functions inside the network makes Eq. 4.5 non convex respect to the network parameters, making the whole optimization process non banal. One positive aspect, is that the gradient of Eq. 4.5 can be computed exactly and efficiently by applying the simple chain rule from differential calculus through dynamic programming. This means that we could use the gradient information in order to find a stationary point of Eq. 4.5.

The gradient vector in neural networks is computed by the *error backpropagation* algorithm proposed by [109] where the name backpropagation comes from the fact that the prediction errors are propagated backwards inside the network starting from the output nodes.

The first stage is called the forward pass because in order to compute the objective function Eq. 4.5, its computation starts from the input layer as we showed in the previous chapter. The computation performed at the unit (or neuron) u_j before the activation is applied, is given by a linear combination of the weights and the outputs from the previous layer

$$u_j = \sum_i z_i w_{ij} \tag{4.6}$$

is worth to notice that z_i could an output from a particular neuron or directly a data input x_i, while u_j could be hidden unit or an output unit as well. We are interested in the partial derivative of $\partial e(\mathbf{w})/\partial w_{ij}$, which can be decomposed using the chain rule

$$\frac{\partial e(\mathbf{w})}{\partial w_{ij}} = \frac{\partial e(\mathbf{w})}{\partial u_j} \frac{\partial u_j}{\partial w_{ij}} \tag{4.7}$$

Suppose that u_j is an output neuron, since a linear activation function in the output layer is used we can write

$$\frac{\partial e(\mathbf{w})}{\partial w_{ij}} = 2 \sum_{n=1}^{N} (u_j - y_n) z_i \tag{4.8}$$

the terms given by $\frac{\partial e(\mathbf{w})}{\partial u_j} = 2\sum_{n=1}^{N}(u_j - y_n)$ are called errors. Those terms will propagate back through the network as we will show in the next steps. Now let's consider u_j as an hidden layer unit, consequently in this case the term $\frac{\partial e(\mathbf{w})}{\partial u_j}$ must to consider that a variation in u_j will cause a variation also to all K units where u_j has connection with u_k

$$\frac{\partial e(\mathbf{w})}{\partial u_j} = \sum_{k=1}^{K} \frac{\partial e(\mathbf{w})}{\partial u_k}\frac{\partial u_k}{\partial u_j} \tag{4.9}$$

again u_k could be an hidden unit or an output unit. The second term in Eq. 4.9 is given by

$$\frac{\partial u_k}{\partial u_j} = z_j' \sum_{k=1}^{K} w_{jk} = h'(a_j)\sum_{k=1}^{K} w_{jk} \tag{4.10}$$

then Eq. 4.9 becomes

$$\frac{\partial e(\mathbf{w})}{\partial u_j} = z_j' \sum_{k=1}^{K} \frac{\partial e(\mathbf{w})}{\partial u_k} w_{jk} \tag{4.11}$$

where the derivative of the activation z_j' can be computed exactly since is an elementary function. We can explicit Eq. 4.11 supposing that u_k is an output neuron

$$\frac{\partial e(\mathbf{w})}{\partial w_{ij}} = \frac{\partial e(\mathbf{w})}{\partial u_j}\frac{\partial u_j}{\partial w_{ij}} = \frac{\partial e(\mathbf{w})}{\partial u_k}\frac{\partial u_k}{\partial u_j}\frac{\partial u_j}{\partial w_{ij}} = z_i z_j' \sum_{k=1}^{K}\sum_{n=1}^{N}(u_k - y_n)w_{jk} \tag{4.12}$$

highlighting that we start to compute derivatives from the output layer and recursively proceeding backwards towards the input layer.

Once the gradient is computed, the natural way to proceed is to use a first order unconstrained optimization method to find a new set of parameters such that the loss function will decrease.

4.2.3 Optimization and Regularization

Once the gradient of Eq.4.5 respect to the parameters is computed, we are ready to perform a step s using for example the gradient descent algorithm

$$\mathbf{w}_{s+1} = \mathbf{w}_s + \alpha\nabla e(\mathbf{w}) = \mathbf{w}_s - \frac{\alpha}{n}\sum_{n=1}^{N}\nabla e_n(\mathbf{w}_s) \tag{4.13}$$

where $\alpha \in (0, 1]$ is the step size or *learning rate*. Optimize the empirical error is one the most challenging tasks in NN's training because we facing a nonconvex optimization problem. Since the gradient can be computed exactly and because nowadays Deep Neural Networks could contain millions of parameters a global optimization algorithm could be not best viable option. For those reasons usually local first order methods for unconstrained optimization are the natural way to attack this problem.

The particular form of Eq. 4.13, allows especially in big data setting, to compute the gradient respect to one or a subset of data drawn independently and uniformly at random with replacement. This is the main idea of the Stochastic Gradient Descent

(SGD) method initially introduce by [103] which makes an update of the weight vector respect to only one data point

$$\mathbf{w}_{s+1} = \mathbf{w}_s - \alpha \nabla e_n(\mathbf{w}_s) \tag{4.14}$$

When every data point is used for updating the parameters the optimization algorithm has computed one *epoch*. Since the gradient can be computed independently for each data data point allows the usage of massive parallel computations which could be not exploited from Eq.4.13. In between of stochastic and full batch Gradient Descent there is the possibility to use a mini batch of size n_b for the parameters update

$$\mathbf{w}_{s+1} = \mathbf{w}_s - \frac{\alpha_s}{n_b} \sum_{n=1}^{n_b} \nabla e_n(\mathbf{w}_s) \tag{4.15}$$

which is usually more used in practice.

The advantages to have an estimate of the gradient during the optimization are analyzed in depth in [15]. Is not hard to imagine that stochastic approaches, in case of redundancy (i.e low variance) in the dataset the update becomes much more efficient. Also in [73] shows that using noisy estimates of the gradient improves the generalization accuracy, in particular that using large batch sizes, the optimizer tends to converges into sharp minima with a consequent degradation of the generalization accuracy. However the SGD has some disadvantages, the first one is that the step size becomes an important hyper parameter that must be estimated with cross validation and second the loss function could be very sensitive in some directions respect to the others (i.e ill-conditioned Hessian matrix) [54]. Recently some algorithms have been developed with the idea to overcome this two issues. Among others there is the *Adam* [75] algorithm which stands for adaptive moments estimates. Adam estimate the first and the second moment (i.e. mean and variance) of the gradient respectively, computing the exponential moving averages of the gradient and the squared gradient. The parameter update rule is given by

$$\begin{aligned}
\mathbf{m}_{s+1}^{(1)} &= \frac{\rho_1 \mathbf{m}_s^{(1)} + (1-\rho_1^s)\nabla e_n(\mathbf{w}_s)}{1-\rho_1} \\
\mathbf{m}_{s+1}^{(2)} &= \frac{\rho_2 \mathbf{m}_s^{(2)} + (1-\rho_2)\nabla e_n(\mathbf{w}_s) \odot \nabla e_n(\mathbf{w}_s)}{1-\rho_2^s} \\
\mathbf{w}_{s+1} &= \mathbf{w}_s - \alpha \frac{\mathbf{m}_{s+1}^{(1)}}{\sqrt{\mathbf{m}_{s+1}^{(2)}} + \epsilon}
\end{aligned}$$

where the suggested parameters for moments estimates ρ_1 and ρ_2 and for the step size α are 0.9, 0.999 and 0.001 with ϵ chosen as a small constant. The greatest challenging task, that is common for every Machine Learning model is that we should obtain an accurate prediction not only on the training data but also on the unseen data.

The *overfitting* problem occurs when our model achieve a low value of the loss function at training time but when we test the model on new unseen data (i.e test set) we obtain an high value of the loss function.

Many aspect should be taken into account to improve the generalization behavior of Deep Neural Networks. The first one is the number of parameters presents inside the network. An high number of parameters usually let the model to overfit the data very easily. Surprisingly this doesn't happen all the time, in fact most of recent Deep Neural Networks models especially in Computer Vision are *overparametrized* which means that the number of parameters are much larger than the number of the observations and they show a very accurate generalization performance [2].

A first simple method for improve generalization is called *early stopping*, where at each epoch we evaluate the validation loss, that is the value of the objective function computed respect another subset of the dataset (i.e validation set). The training will be considered terminated when the loss function respect to the validation set doesn't improve anymore. After that the test error will be evaluated using the optimal parameters found by the optimizer.

Another way to control the complexity of the network is to add a penalization term to the loss function respect to the norm of the parameter vector $\mathbf{w}$. A common choice is to use an L_2 for the norm obtaining

$$r(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_n - f(x_n, \mathbf{w}))^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{4.16}$$

which is also called *Tikhonov Regularization* or *weight decay*. The parameter λ controls the penalization level to inject in the objective function in Eq. 4.16 and should be treated as another hyperparameter to estimate. The parameters update of the SGD algorithm are given by

$$\mathbf{w}_{s+1} = (1 - \alpha\lambda)\mathbf{w}_s - \alpha\nabla e_n(\mathbf{w}_s) \tag{4.17}$$

As we can see the parameters in the previous step are shrunken by a factor of $(1 - \alpha\lambda)$. It's interesting to notice that even if the weight decay and the early stopping methods could seem quite different, in reality they are very similar if we analyze a bit more in details the networks behavior using these two methodology. Let's write down the quadratic approximation of the unregularized objective function around it's minima $\mathbf{w}^\star$

$$e_q(\mathbf{w}) = e(\mathbf{w}^\star) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\top\mathbf{H}(\mathbf{w} - \mathbf{w}^\star) \tag{4.18}$$

adding the regularization term in 4.18 and writing the gradient gives to us

$$(\mathbf{H} + \lambda\mathbf{I})\mathbf{w} = \mathbf{H}\mathbf{w}^\star$$
$$\mathbf{w} = (\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}\mathbf{w}^\star$$

we can see that for $\lambda \to 0$, the regularized $\mathbf{w}$ and unregularized $\mathbf{w}^\star$ solutions are the same. More insights can be obtained if we use the spectral decomposition for the real and symmetric Hessian matrix $\mathbf{H} = \mathbf{U}\boldsymbol{\Xi}\mathbf{U}^\top$

$$\mathbf{w} = \mathbf{U}(\boldsymbol{\Xi} + \lambda\mathbf{I})^{-1}\boldsymbol{\Xi}\mathbf{U}^\top\mathbf{w}^\star \tag{4.19}$$

each eigenvector is scaled by a factor of $\frac{\xi_i}{\lambda+\xi_i}$, $\forall i = 1, \dots D$. Consequently, for directions where $\lambda \ll \xi_i$, the corresponding parameter dimension will be similar to

the maximum likelihood solution $\mathbf{w} = \mathbf{w}^\star$. While for $\lambda \gg \xi_i$, the relative dimension it'll be close to zero $w_i \simeq 0$.

A similar behavior can be obtained quantitatively in case of early stopping. If we write down the gradient descent update for Eq. 4.16 denoting with $\hat{\mathbf{w}}$ parallel to the eigenvectors of the hessian matrix $\mathbf{H}$

$$\begin{aligned}\mathbf{w}_{s+1} &= \mathbf{w}_s - \alpha\mathbf{H}(\mathbf{w}_s\mathbf{w}^\star)\\ \mathbf{U}^\top\mathbf{w}_{s+1} &= \mathbf{U}^\top\mathbf{w}_s - \alpha\mathbf{U}^\top\mathbf{H}(\mathbf{w}_s - \mathbf{w}^\star)\\ \hat{\mathbf{w}}_{s+1} &= \hat{\mathbf{w}}_s - \alpha\mathbf{\Xi}_p(\hat{\mathbf{w}}_s - \hat{\mathbf{w}}^\star)\end{aligned}$$

taking into account only the j-th dimension of $\hat{\mathbf{w}}_s$ and the relative eigenvalue ξ_j, by induction is possible to prove that after $s = N + 1$ steps we have

$$\hat{w}_{j,N+1} = (1 - (1 - \alpha\xi_j)^{N+1})w_j^\star \tag{4.20}$$

if $|1 - \alpha\xi_j| < 1$ for $s \to \infty$ then $\hat{w}_j = w_j^\star$, meaning that without early stopping we obtain the unregularized solution. For a finite number of steps, if the eigenvalue is much smaller than the reciprocal of $s\alpha$ as $\xi_j \gg (s\alpha)^{-1}$, then means that the number of iterations s are still large, then $\hat{w}_{j,s+1} \simeq w_j^\star$ supposing that $|1 - \alpha\xi_j| < 1$. Otherwise, in case $\xi_j \ll (s\alpha)^{-1}$, using the Taylor expansion for the term $(1 - \alpha\xi_j)^N \simeq 1 - s\alpha\xi_j$ and substituting back to Eq. 4.20 we have that $|\hat{w}_{j,s}| \simeq s\alpha\xi_j|w_j^\star| \ll |w_j^\star|$. This highlight the fact that the reciprocal of $s\alpha$ play a similar role as the regularization parameter λ in weight decay.

The last regularization method that we'll discuss is called *dropout* proposed in [134]. The dropout technique allows to combine by averaging the prediction of an exponentially number of thinned NN's by randomly drop some of the units in the network, simply associating to them a Bernoulli random variable, for example in the layer l we have

$$\mathbf{r}^l \sim \text{Bernoulli}(p) \tag{4.21}$$

$$\hat{\mathbf{z}}^l = \mathbf{r} \odot \mathbf{z}^l \tag{4.22}$$

where $\mathbf{r}^l$ is a vector of independent Bernoulli random variables at the layer l, successively a component wise multiplication is performed against the outputs units $\mathbf{z}^l$. This mechanism is applied for each layer with the error backpropagation performed respect to this subnetwork. This process of randomly drop the units in the networks has showed a great performance in reducing the overfitting problem in NN's training.

4.2.4 Uncertainty Estimation in Neural Networks

In many applications we are interested in the *uncertainty* associated to a prediction provided by our model. In particular the main quantity of interest is the *posterior distribution* $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$ which can be evaluated by the *Bayes Theorem*

$$p(\mathbf{w}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{Y}|\mathbf{X})} \tag{4.23}$$

at inference time we integrate over the parameter space to obtain the *predictive distribution* for a particular test point $\mathbf{x}^\star$

$$p(\mathbf{y}^\star|\mathbf{x}^\star, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^\star|\mathbf{x}^\star, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\mathbf{w} \tag{4.24}$$

where the prediction is averaged over all possible realizations of the parameters given the observables. The main challenge here, is the computation of the posterior due to the difficulty to evaluate the evidence at the denominator of 4.23. Marginalizing out the parameters

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \mathbf{w}) p(\mathbf{w}) \mathrm{d}\mathbf{w} \tag{4.25}$$

this integral rarely can be computed analytically especially in the case of Neural Networks, because of the nonlinear dependence of the NN's function and the parameters.

In the last decades many effort has been dedicated in the efficient approximation of Eq. 4.23. A Bayesian treatment of Neural Networks has been applied and developed by many researchers using various techniques from variational inference [60, 5], Laplace approximation [84] and recently in [48] showing that applying the dropout technique is the same as applying a particular form of variational inference. Since the last approach is the one that we used in our application we will show the main concepts.

Variational inference (VI) is a deterministic method to approximate the posterior distribution when the integral can't be solved in closed form. The underlying idea about VI is to provide an analytical approximation to the posterior in terms of factorized distributions (usually Gaussians). Let's begin observing that the *marginal distribution* can be decomposed as

$$\ln p(\mathbf{Y}|\mathbf{X}) = \mathcal{L}(q) + \mathrm{KL}(q||p) \tag{4.26}$$

where the first term is called the *evidence lower bound* (ELBO) and the second term is *Kullback Leibler divergence* between the variational distribution $q(\mathbf{w})$ and the posterior

$$\begin{aligned} \mathcal{L}(q(\mathbf{w})) &= \int q(\mathbf{w}) \ln\left\{\frac{p(\mathbf{Y}, \mathbf{w}|\mathbf{X})}{q(\mathbf{w})}\right\} \mathrm{d}\mathbf{w} \\ \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}|\mathbf{X}, \mathbf{Y})) &= -\int q(\mathbf{w}) \ln\left\{\frac{p(\mathbf{w}|\mathbf{X}, \mathbf{Y})}{q(\mathbf{w})}\right\} \mathrm{d}\mathbf{w} \end{aligned}$$

the main idea here is to minimize the KL-divergence (or maximize the ELBO) respect to the variational distribution $q(\mathbf{w})$ such that we can compute the predictive distribution as

$$p(\mathbf{y}^\star|\mathbf{x}^\star, \mathbf{X}, \mathbf{Y}) \approx \int p(\mathbf{y}^\star|\mathbf{x}^\star, \mathbf{w}) q^\star(\mathbf{w}) \mathrm{d}\mathbf{w} \tag{4.27}$$

the KL-divergence term can be further approximated taking the sum respect to size

of dataset

$$\begin{aligned}\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}|\mathbf{X},\mathbf{Y})) &\approx -\int q(\mathbf{w})\ln p(\mathbf{Y}|\mathbf{X},\mathbf{w}) + \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}))\mathrm{d}\mathbf{w} \\ &= -\sum_{n=1}^{N}\int q(\mathbf{w})\ln p(\mathbf{y}_n|\mathbf{f}_n,\mathbf{w}) + \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}))\mathrm{d}\mathbf{w} \quad (4.28)\end{aligned}$$

where $\mathbf{f}_n = \mathbf{f}(\mathbf{x}_n)$ is the n-th output of the Neural Network. The first term in Eq. 4.28 can be approximated with Monte Carlo (MC) integration sampling $\hat{\mathbf{w}}_n \sim q(\mathbf{w})$, giving

$$\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}|\mathbf{X},\mathbf{Y})) \approx -\frac{1}{N}\sum_{n=1}^{N}\ln p(\mathbf{y}_n|\mathbf{f}_n,\hat{\mathbf{w}}_n) + \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w})) \quad (4.29)$$

Where for each term inside the summation we sample the the weight matrix from its variational distribution $q(\mathbf{w})$. The variational distribution is defined as mixture of two Gaussians [48] defined respect to each row of the weight matrix $\mathbf{w}_k$

$$q(\mathbf{w}_k) = p\mathcal{N}(0,\sigma^2\mathbf{I}) + (1-p)\mathcal{N}(\mathbf{m}_k,\sigma^2\mathbf{I}) \quad (4.30)$$

with $p \in [0,1]$ and $\mathbf{m}_k$ the variational parameter. In this way the second term can be approximated in terms of L_2 norms, assuming a standard Gaussian prior distribution [47]. The procedure to optimize Eq. 4.29, can be equivalently obtained by optimizing the parameter of the NN's with the scalar p given by the dropout procedure. We can obtain the uncertainty of the predictions applying dropout at test time, where the approximated predictive distribution is given by

$$p(\mathbf{y}^\star|\mathbf{x}^\star,\mathbf{X},\mathbf{Y}) \approx \int p(\mathbf{y}^\star|\mathbf{x}^\star,\mathbf{w})q^\star(\mathbf{w})\mathrm{d}\mathbf{w} \approx \frac{1}{J}\sum_{j=1}^{J}p(\mathbf{y}^\star|\mathbf{x}^\star,\hat{\mathbf{w}}_j) \quad (4.31)$$

where P is the number of the weight realizations using the dropout procedure. Where the predictive mean and variance is simply given by $\mathbb{E}[p(\mathbf{y}^\star|\mathbf{x}^\star)] = \frac{1}{J}\sum_{j=1}^{J}\mathbf{f}_j$ and the model uncertainty $\mathrm{Var}[p(\mathbf{y}^\star|\mathbf{x}^\star)] = \frac{1}{J}\sum_{j=1}^{J}(\mathbf{f}_j - \overline{\mathbf{f}})^2$.

4.3 Recurrent-Type Neural Networks

The classical Neural Network model discussed in the previous chapter treats each observation or data point in the same way, without considering the correlation across the data points as it assumes that they are independent and identically distributed.

In many tasks as in time series forecasting the value of the target variable $\mathbf{y}_t$ is usually strongly correlated to the past values of the target variable at time $\mathbf{y}_{t-1}$. This correlation is lost in a classical NN model.

In order to solve this limitation Recurrent Neural Networks (RNN's) have been developed with the objective to learn the dependencies of the data across the timestamps and to improve the prediction accuracy in case of sequential data.

4.3.1 Recurrent Neural Networks

The main difference respect to standard NN model is that the hidden units $\mathbf{z}_t$ or *states* of the network are allowed to pass at the successive time stamps being a function of the data $\mathbf{x}_t$ and the state at the previous time stamp $\mathbf{z}_{t-1}$ namely $\mathbf{z}_{t+1} = h(\mathbf{x}_t, \mathbf{z}_{t-1})$. Writing down the equations for the forward propagation in case of RNN's we have for $t = 1, \ldots, T$

$$\mathbf{z}_t = \tanh(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{zz}\mathbf{z}_{t-1}) \tag{4.32}$$

$$\mathbf{f}_t = \mathbf{W}_{zf}\mathbf{z}_t \tag{4.33}$$

where T is the time window and also the number of stacked RNN's cells, the tanh function is applied element wise as in a classical NN's, the matrices $\mathbf{W}_{xz}$, $\mathbf{W}_{zz}$ and $\mathbf{W}_{zf}$ have dimension $M \times D$, $M \times M$ and $K \times M$ respectively. A linear activation is used to compute the output vector $\mathbf{f}_t$ since we are interested in time series forecast. The same setting of the matrices parameters are used in each time stamp despite the states that can evolve in time. This *parameter sharing* characteristic allows the network to generalize better even in case of limited number of training data [54].

4.3.2 Error Backpropagation in Time

The RNN will be trained in order to minimize the following loss

$$e(\mathbf{W}) = \frac{1}{T}\sum_{t=1}^{T} ||\mathbf{y}_t - \mathbf{f}(\mathbf{x}_t, \mathbf{W})||^2 \tag{4.34}$$

The set of parameters are shared across in the time window, so that the states will play a special role in the computation of the derivatives.

To highlight the main features of the error backpropagation in RNN's we'll use a matrix/vector notation for the derivatives instead of consider a particular scalar value w_{ij} since the same set of parameters are used through the timestamps. This will also highlight the main problematic in RNN's parameter optimization called the *vanishing* and *exploding gradients* problems.

Let's writing down that

$$\nabla e = \sum_{t=1}^{T} \nabla e_t \tag{4.35}$$

the gradient is given by sum of the gradients at different timestamps. The equations that defines the updates in the RNN depends on three matrices $\mathbf{W}_{xz}$, $\mathbf{W}_{zz}$ and $\mathbf{W}_{zf}$. We will focus in the computation of the derivatives respect to the matrix $\mathbf{W}_{zz}$ because the same procedure can be obtained for $\mathbf{W}_{xz}$ while the computation of the derivatives respect to the matrix $\mathbf{W}_{zf}$ is straightforward. In order to maintain the notation uncluttered we'll refer the matrix $\mathbf{W}_{zz}$ as $\mathbf{W}$. The derivative of the error term in a particular timestamp T' is given by

$$\nabla e_{T'} = \sum_{t=1}^{T'} \frac{\partial \mathbf{e}_{T'}}{\partial \mathbf{f}_{T'}} \frac{\partial \mathbf{f}_{T'}}{\partial \mathbf{z}_{T'}} \frac{\partial \mathbf{z}_{T'}}{\partial \mathbf{z}_t} \frac{\partial \mathbf{z}_t}{\partial \mathbf{W}} \tag{4.36}$$

the parameters $\mathbf{W}$ occurs many times and we have to sum from the first time stamp to the time stamp in which we are considering to compute the derivatives (i.e T'). This is the reason why this procedure is called backpropagation in time.

We can expand even more the right term of Eq. 4.36

$$\nabla e_{T'} = \sum_{t=1}^{T'} \frac{\partial \mathbf{e}_{T'}}{\partial \mathbf{f}_{T'}} \frac{\partial \mathbf{f}_{T'}}{\partial \mathbf{z}_{T'}} \frac{\partial \mathbf{z}_{T'}}{\partial \mathbf{z}_t} \frac{\partial \mathbf{z}_t}{\partial \mathbf{W}} = \sum_{t=1}^{T'} \frac{\partial \mathbf{e}_{T'}}{\partial \mathbf{f}_{T'}} \frac{\partial \mathbf{f}_{T'}}{\partial \mathbf{z}_{T'}} \prod_{j=t+1}^{T'} \left(\frac{\partial \mathbf{z}_j}{\partial \mathbf{z}_{j-1}} \right) \frac{\partial \mathbf{z}_{j-1}}{\partial \mathbf{W}} \tag{4.37}$$

where is possible to notice that the derivative of $\frac{\partial \mathbf{z}_{T'}}{\partial \mathbf{z}_t}$ is given by a consecutive multiplications of Jacobians as

$$\prod_{j=t+1}^{T'} \left(\frac{\partial \mathbf{z}_j}{\partial \mathbf{z}_{j-1}} \right) = \prod_{j=t+1}^{T'} \mathbf{W}^\top \mathrm{diag}(\tanh'(\mathbf{z}_j)) \tag{4.38}$$

The derivative in Eq. 4.38, is obtained by multiplying the same matrix of parameters $\mathbf{W}$ multiple times. This could lead, depending on the magnitude the largest eigenvalue of $\mathbf{W}$, to expose at the end of the computation, the gradient to vanish or to diverge. In general, we have the following result

Lemma 4.3.1. *Let* $\mathbf{A}$ *a square matrix, given* ξ *the largest eigenvalue of* $\mathbf{A}$ *then the value of* $||\mathbf{A}^t||_2 \to 0$ *for* $t \to \infty$ *when* $\xi < 1$ *and the value of* $||\mathbf{A}^t||_2 \to \infty$ *for* $t \to \infty$ *when* $\xi > 1$.

Proof. Diagonalizing the matrix $\mathbf{A}^t = \mathbf{U}\boldsymbol{\Xi}^t\mathbf{U}^\top$ we have that $\xi \to \infty$ when $\xi > 1$ for $t \to \infty$, also $\xi \to 0$ when $\xi < 1$ for $t \to \infty$. □

This problem will be much severe as the time window increase, making very difficult for the network to learn *long-term dependencies* in the time series. For many years, this was the main reason why RNN's were difficult to train.

Is important to remind that as the classical NN's models the RNN's are also universal approximators and then very powerful. Consequently, many modifications are presents in literature in order to overcome this kind of issues are proposed like *gradient clipping* [91] and *batch normalization* [66] or the usage of other variants of recurrent units as we'll show in the next sections.

4.4 Long Short Term Memory and Gated Recurrent Units

In the following sections, we will discuss extensions to the current RNN model. The underlying idea os these new models is to overcome the problems that with discussed in the previous section about the computation of the gradient vector in the case of the RNN. Those extensions have in common that they try to create, through a different mathematical model, *gates* along the time stamps where the derivatives information could flow without numerical issues.

4.4.1 Long Short Term Memory

The Long Short-Term Memory (LSTM) network has been introduced for the first time by [61] and become during the last years one of the most powerful tool for modeling sequences in various domains. The LSTM cell or unit is composed by three main gates called the *input* $\mathbf{i}$, *forget* $\mathbf{g}$, *output* $\mathbf{o}$ and the *cell-state* $\mathbf{c}_t$, all M-dimensional vectors which each of them cover a particular role in the network. Those are given by

$$\begin{bmatrix} \mathbf{i} \\ \mathbf{g} \\ \mathbf{o} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{bmatrix} \mathbf{W} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{z}_t \end{bmatrix} \tag{4.39}$$

where the matrix $\mathbf{W}$ is $4M \times (M + D)$. Then the cell state $\mathbf{c}_t$ and the state $\mathbf{z}_t$ update are given by

$$\mathbf{c}_t = \mathbf{g} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{c} \tag{4.40}$$

$$\mathbf{z}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t) \tag{4.41}$$

let's discuss more in detail the underlying reason of those equations: the vector is called forget $\mathbf{g}$ because it multiplies respect to the cell state at the previous time stamp $\mathbf{c}_{t-1}$, and because the forget vector has values that are between 0 and 1 this can be interpreted as the amount of information that we want allow to pass into the next cell state. The intermediate cell state vector $\mathbf{c}$ is multiplied respect to the input vector $\mathbf{i}$ which can be seen as what kind of new information could be relevant for the current cell state update. Finally the state vector $\mathbf{z}_t$ is updated filtering the cell state vector $\mathbf{c}_t$ with a multiplication respect to output gate $\mathbf{o}$.

4.4.2 Gated Recurrent Units

Another popular RNN's extension is the Gated Recurrent Unit (GRU) proposed by [19]. The mathematical model describing the state updates is similar respect to the LSTM network but in this case we have only two gates: the update gate $\mathbf{d}$ and the reset gate $\mathbf{r}$. In this case we have

$$\begin{bmatrix} \mathbf{d} \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} \text{sigm} \\ \text{sigm} \end{bmatrix} \mathbf{W}_1 \begin{bmatrix} \mathbf{x}_t \\ \mathbf{z}_t \end{bmatrix} \tag{4.42}$$

where the matrix $\mathbf{W}_1$ is $2M \times (M + D)$. Then the state $\mathbf{z}_t$ update is given by

$$\mathbf{z}_t = \mathbf{d} \odot \mathbf{z}_{t-1} + (1 - \mathbf{d}) \odot \tanh\left(\mathbf{W}_2 \begin{bmatrix} \mathbf{x}_t \\ \mathbf{r} \odot \mathbf{z}_{t-1} \end{bmatrix}\right) \tag{4.43}$$

where the matrix $\mathbf{W}_2$ is $M \times (M + D)$ and the length of the update gate $\mathbf{d}$ and of the reset gate $\mathbf{r}$ is M. As we can observe the reset gate decide which information we should retain from the previous hidden state $\mathbf{z}_{t-1}$.

4.4.3 Sequence Modeling in Recurrent-Type Neural Networks

Recurrent Neural Networks are extremely powerful models not only for their capability to obtain an accurate prediction but also because, dependent on the application, one can modify the inside structure in order to achieve the desired output. In this case we are interested to predict, at each particular instant time t, multiple time stamps T' of the target variable in the future. Also, we can take into account that the length of the desired output (in this case T') may differ from the length of the input T. This particular problem is called *Sequence To Sequence* learning where the model is trained to map an input sequence of fixed length $\mathbf{x}_t$ for $t = 1, \ldots, T$ which best predicts the target variables $\mathbf{y}_t$ for $t = T, T+1, \ldots, T'$. A particular

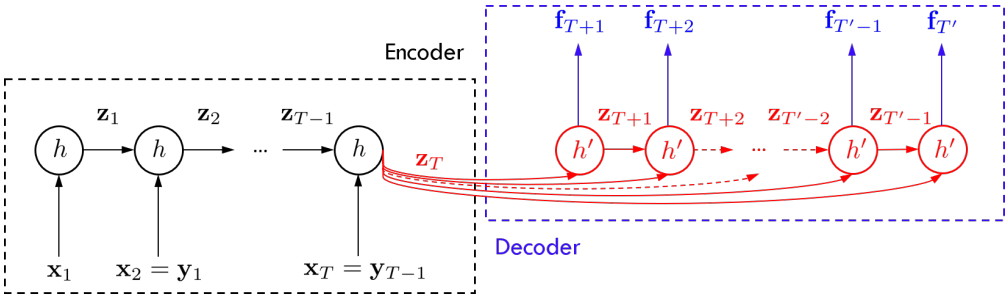


Figure 4.1. Conceptualization of the sequence to sequence learning via encoder-decoder model.

architecture that allows to model this kind of problems is the *Encoder-Decoder* [135] model developed for machine translation. The model is composed by two parts: the encoder network which take all the inputs vector $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and return a latent representation of what the encoder learnt in the time window T, namely the final hidden state $\mathbf{z}_T$ for $t = 1, \ldots, T$

$$\mathbf{z}_T = \tanh(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{zz}^{(1)}\mathbf{z}_{t-1}) \tag{4.44}$$

Given the vector $\mathbf{z}_T$, the decoder network will map into the target space $\mathcal{R}^K$ the latent representations for $t = T+1, \ldots, T'$

$$\mathbf{z}_t = \tanh(\mathbf{W}_{z_T z}\mathbf{z}_T + \mathbf{W}_{zz}^{(2)}\mathbf{z}_{t-1}) \tag{4.45}$$

$$\mathbf{f}_t = \mathbf{W}_{zf}\mathbf{z}_t \tag{4.46}$$

in this case we are considering RNN's cells in the model but this architecture can be also used in the same way with LSTM or GRU cells. The objective function that we are going to minimize at the end is

$$e(\mathbf{W}) = \frac{1}{(T'-T)} \sum_{t=T+1}^{T'} ||\mathbf{y}_t - \mathbf{f}_t(\mathbf{x}, \mathbf{W})||^2 \tag{4.47}$$

where the summation is performed along the outputs generated by the decoder network.

4.5 Application: Ship Motion Prediction in High Sea State Level

In this last section we will present the numerical results for the variational RNN, GRU and LSTM for the motion prediction of the DTMB 5415M.

4.5.1 Problem Definition

The hull form under investigation is the MARIN model 7967 which is equivalent to 5415M [143]. This is a geosim replica of the DTMB 5415 model with different appendages designed by MARIN. The DTMB 5415 is an open-to-public naval combatant hull geometry. The model was self-propelled and kept on course by a proportional-derivative (PD) controller actuating the rudders angle.

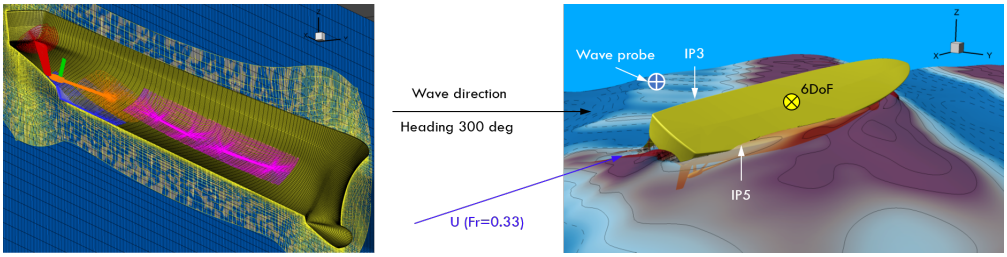


Figure 4.2. Detail of the boundary-layer computational grid (left) and a CFD snapshot with location of the probes (right).

The code CFDShip-Iowa V4.5 [65] is used for the CFD computations. CFDShip-Iowa is an overset, block structured CFD solver designed for ship applications using either an absolute or a relative inertial nonorthogonal curvilinear coordinate system for arbitrary moving but non-deforming control volumes. The free-running CFD simulations were performed with propeller RPM fixed to the self-propulsion point of the model for the envisaged speed. The simulations were conducted in irregular long crested waves, following a JONSWAP spectrum. The turbulence is computed by the isotropic Menter's blended $k-\epsilon/k-\omega$ (BKW) model with shear stress transport (SST) using no wall function. The location of the free surface is given by the "zero" value of the level-set function, positive in the water and negative in the air. The 6 degrees of freedom rigid body equations of motion are solved to calculate linear and angular motions of the ship. A simplified body-force model is used for the propeller, which prescribes axisymmetric body force with axial and tangential components. The total number of grid points is about 45M. Further details can be found in [119].

The data set collects 8 CFD runs (with different random phases) at Fr = 0.33, with nominal peak period $T_p = 9.2$ s and wave heading of 300 deg. It may be noted that the simulation conditions are close to a resonance condition for the roll. The nominal significant wave height is equal to 7 m, corresponding to sea state 7 (high), according to the World Meteorological Organization (WMO) definition. A total of 215 encounter waves have been recorded, with a total run length of about 3323 s and a data rate equal to 129.2 Hz (for the current application the data set has been down-sampled to 8.6 Hz). Wave elevation far from the ship, ship motions (the 6 DoF), rudder angle, and two immersion probes (IP3 and IP5) time series compose

the data set. Figure 4.2 shows a detail of the computational grid (on left) and a snapshot of the ship behavior with the location of signal probes (on right).

The main objective is to obtain an accurate real-time short-term prediction of about 20 s (about one and an half roll periods) of the ten variables ($D = 10$) at the same time.

4.5.2 Numerical Results

For the current analysis the number of cells of the decoder network are fixed in order to produce approximately $20\,s$ ahead prediction.

The NRMSE in this this case is normalized to the data range in order to have a more realistic assessment of the errors obtained across the variables since most of them lies in different ranges; the total NRMSE is given by averaging the NRMSE accross the variables $\mathbf{y} \in \mathcal{R}^D$. For a particular variable the NMRSE is given by

$$\text{NRMSE}_k = \sqrt{\frac{1}{(T' - T)} \sum_{t=T+1}^{T'} \left(\frac{(y_t - f_t)^2}{\max(y) - \min(y)} \right)} \tag{4.48}$$

For the cross-validation procedure the entire dataset has been divided in 70% training set, 20% validation set and 10% test set.

For the current application due to size of the dataset and the limited computational resources a non exhaustive search of the best hyperparameters for the three forecast model has been performed. The hyperparameters are fixed using a grid search by evaluating different: number of cells (width of the network), layers (depth of the network), number of hidden units M, dropout percentage and batch size. The optimization is carried out using the Adam algorithm for a maximum number of epochs fixed at 1000. As regularizer we used the early stopping strategy. The optimal set of hyperparameters are shown in Tab. 4.1. Interestingly, the same optimal set of the hyperparameters are found across all the three models. In Tab. 4.2 the breakdown of NRMSE respect to each target variable is computed for the three networks. Is possible to notice that the Sway and the IP3 are the most challenging variables to forecast as the maximum averaged NRMSE is obtained for those variables. The LSTM model seems to suffer some difficulties for the prediction of the Sway since it achieves the highest overall NRMSE value. On the other side the NRMSE of the Heave and the Pitch is the lowest obtained across the variables where the best models in this case are the GRU and the LSTM respectively.

Table 4.1. Summary of the hyperparameters optimal set found via cross-validation.

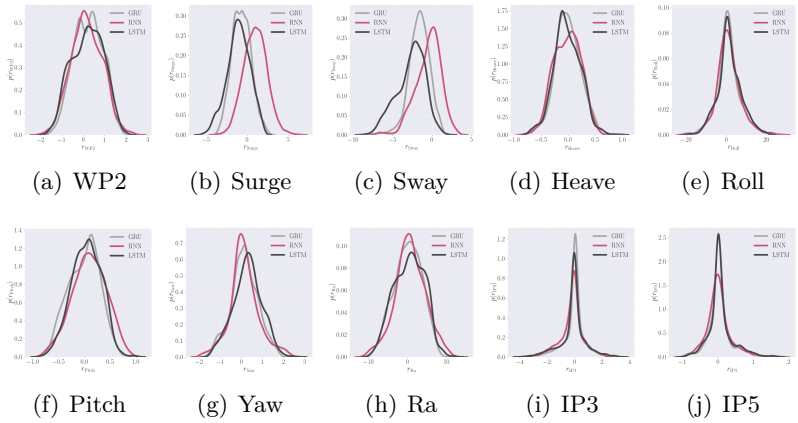
Model	M (encoder, decoder)	Batch Size	Dropout	N. cells (encoder)	N. layers
RNN	(100, 100)	512	0.2	25	1
GRU	(100, 100)	512	0.2	25	1
LSTM	(100, 100)	512	0.2	25	1

In order to obtain more insights about the performance of the proposed methodologies we performed a Kernel Density Estimate (KDE) of the residuals $r = y_t - f_t$ for each variable considered into the analysis. An important property that the

Table 4.2. NRMSE breakdown for each variable for training and test sets.

Data set	Training			Test		
Variable	RNN	GRU	LSTM	RNN	GRU	LSTM
WP2	0.079	0.057	0.052	0.186	**0.175**	0.191
Surge	0.008	0.005	0.005	0.028	**0.022**	0.029
Sway	0.022	0.012	0.011	**0.059**	0.075	0.115
Heave	0.086	0.050	0.049	0.140	**0.132**	0.140
Roll	0.017	0.010	0.009	0.026	**0.025**	0.026
Pitch	0.062	0.036	0.035	0.121	0.105	**0.102**
Yaw	0.046	0.031	0.028	0.146	**0.135**	0.151
Rudder	0.024	0.016	0.014	0.037	**0.033**	0.036
IP3	0.068	0.041	0.038	0.111	**0.106**	0.121
IP5	0.100	0.062	0.057	0.156	**0.146**	0.154
Average	0.051	0.032	0.030	0.101	**0.095**	0.107

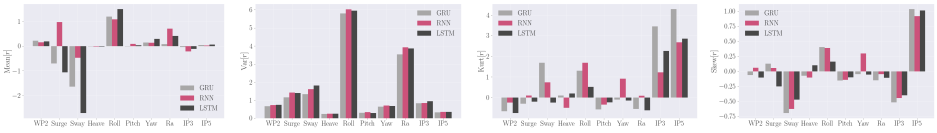
residuals obtained from a forecasting model should satisfy is that they should have a zero mean. In case of residuals with a mean strongly different from zero it means that there is bias in the prediction and the forecasting model could be improved. In Fig. 4.4(a) is possible to observe the mean of the residuals for all the predicted features. The variables: Surge, Sway, Roll and Ra show a mean slightly different from zero, especially from the forecast of the Sway obtained with the GRU and LSTM. The RNN seems more robust in this case. In Fig. 4.4(b) we reported the variance of the residuals to highlight that the variables are in a different range. Other interesting observations can be retrieved by analyzing the kurtosis (Fig. 4.4(c) and skewness Fig. 4.4(d)) of the residuals. A substantial high value of the kurtosis is obtained for the residuals of the IP3 and IP5 forecasts, meaning that the relative distribution has high tails indicating the presence of abnormal high and low values in the residuals as shown in Fig. 4.3(i), and Fig. 4.3(j). This is mainly due to the fact that, the IP3 and IP5 variables present a strong change from zero in some particular instants which seems difficult to model (i.e. high absolute value of the residuals), while for the rest of the time their values are very regular and simple to forecast (i.e. very low value of the residuals). The skewness for univariate distribution, measure whether there is more weight in the right tail of the distribution (i.e. positive skewness) or in the left tail (i.e negative skewness). An high positive skewness is obtained for the residuals of the IP5 variable indicating a systematic underestimate of the forecast while the opposite behavior is obtained for the Sway and the IP3. Finally, in the last figures is possible to observe the forecast obtained through the test set for all the variables considered. Three time intervals are showed, specifically $I_1 \in [4545, 4565], I_2 \in [4554, 4574]$ and $I_3 \in [4565, 4585]$. Is worth noting that the last time instant for which all the three model have been trained on is at 3391.8 s (i.e last time stamp on the training set). As expected the quality of the forecast starts to deteriorate as we will try to predict time windows which are very far from the training data (e.g I_3). From the perspective of a real time scenario, means that the model should retrained to avoid this deterioration of the performances. However,



(a) WP2 (b) Surge (c) Sway (d) Heave (e) Roll

(f) Pitch (g) Yaw (h) Ra (i) IP3 (j) IP5

Figure 4.3. KDE residuals.



(a) Mean (b) Variance (c) Kurtosis (d) Skewness

Figure 4.4. Statistical moments for the residuals.

an overall good forecast can be observed.

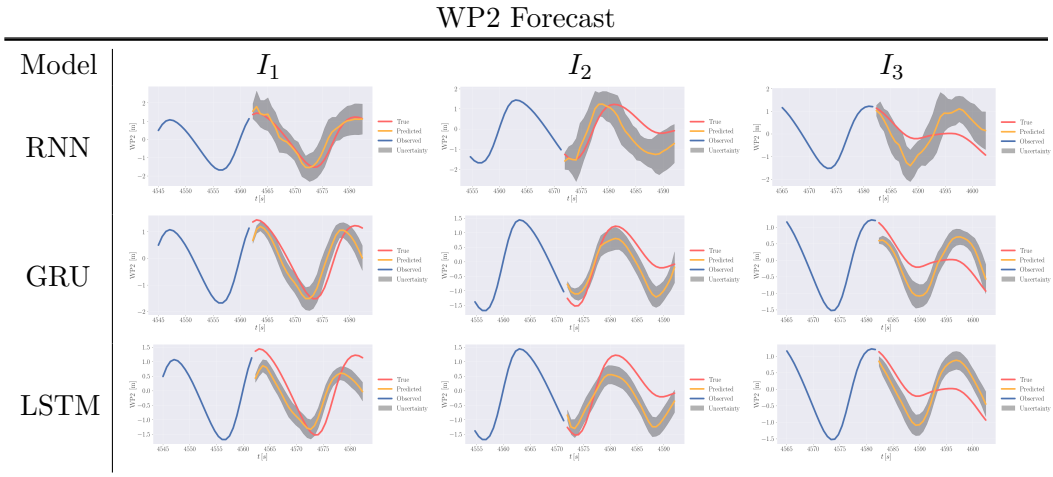


Figure 4.5. Forecast of the WP2 variable for the three chosen time intervals.

4.5.3 Conclusions and Future Work

In this chapter we present the application of three recurrent-type NN's for ship motion prediction. Ten variable have been considered into the analysis namely: WP2, Surge, Sway, Heave, Roll, Pitch, Yaw, Ra, IP3, IP5. All the variables have

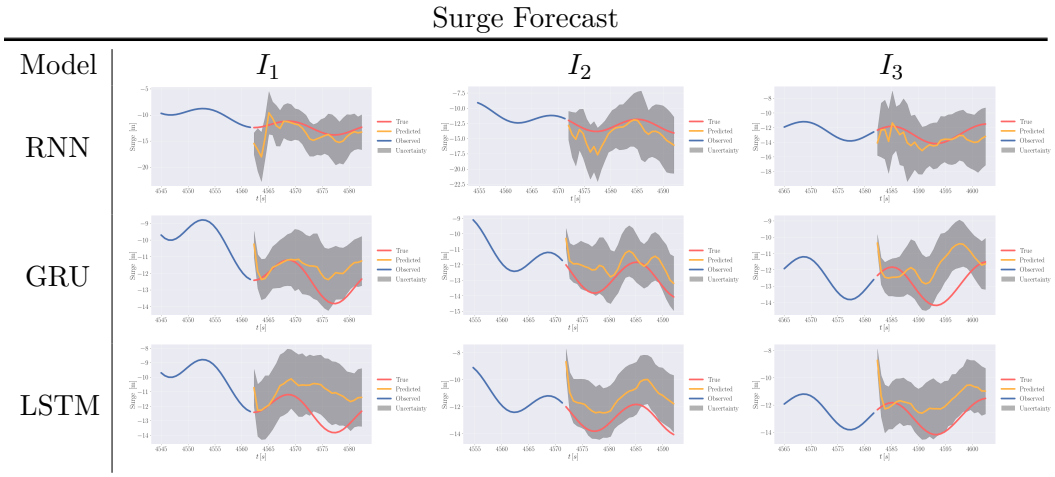


Figure 4.6. Forecast of the Surge variable for the three chosen time intervals.

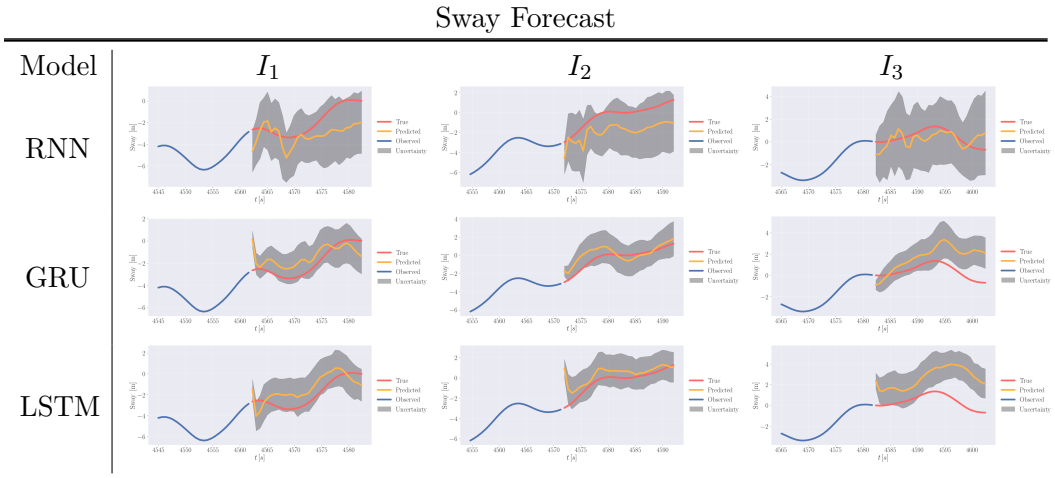


Figure 4.7. Forecast of the Sway variable for the three chosen time intervals.

been used for the forecast defining a multiple time series forecasting problem. The objective was to obtain a 20 sec ahead prediction.

An overall good performance is obtained across all the methodologies considered in this work: RNN, LSTM and GRU.

The Sway and the IP3 are the variables more difficult to predict given their high NRMSE value respect to the other variables. Some problems relative to the bias in the forecast of the Surge and the Sway are noticed especially in GRU and LSTM.

From the current numerical experiment, the GRU model seems performing the best, but to assess the statistical significance of this result more runs should be performed needed. In future work, other NN's architectures and Vector Auto-Regressive type methodologies for time series forecasting will be considered together with a more extensive search of the optimal hyperparameters. Also, the possibility to use GPU architectures in order to speed up the training but also the test time will be addressed. This is crucial in case the proposed methods will be implemented in a real time scenario.

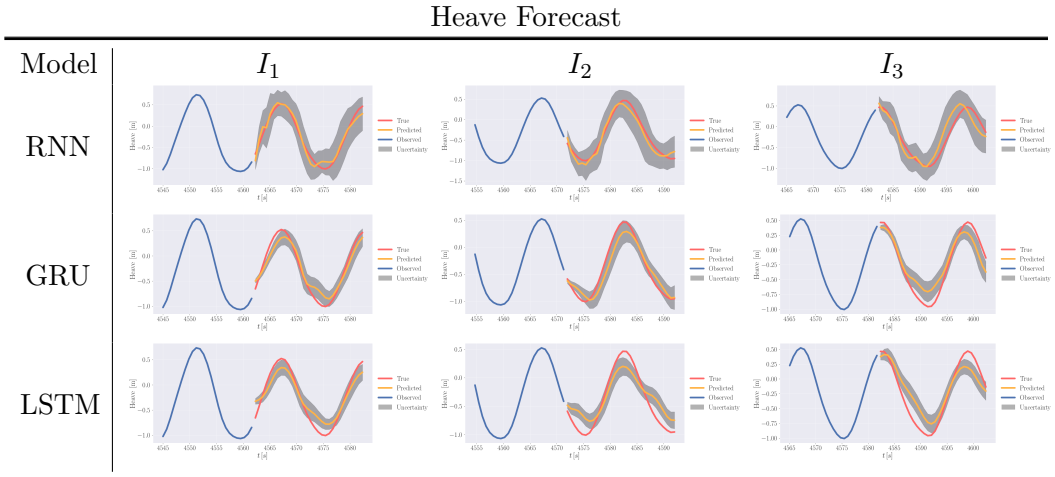


Figure 4.8. Forecast of the Heave variable for the three chosen time intervals.

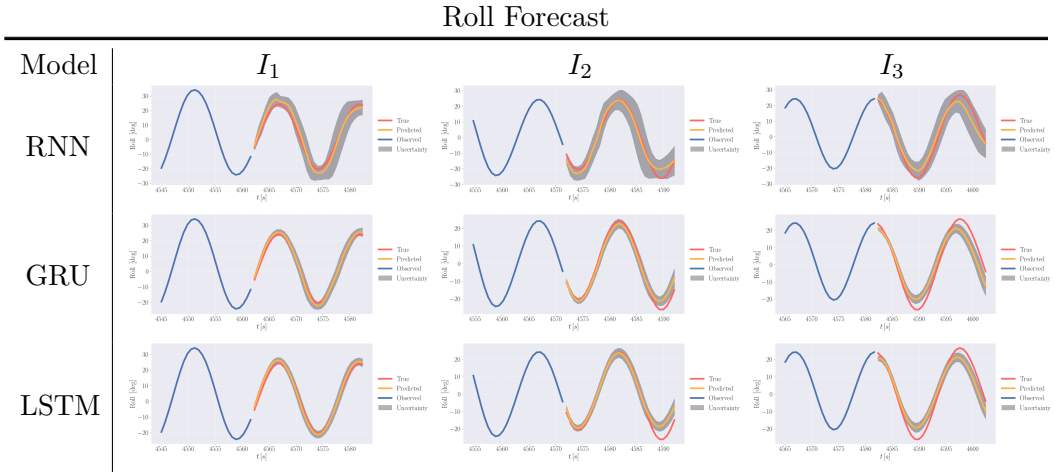


Figure 4.9. Forecast of the Roll variable for the three chosen time intervals.

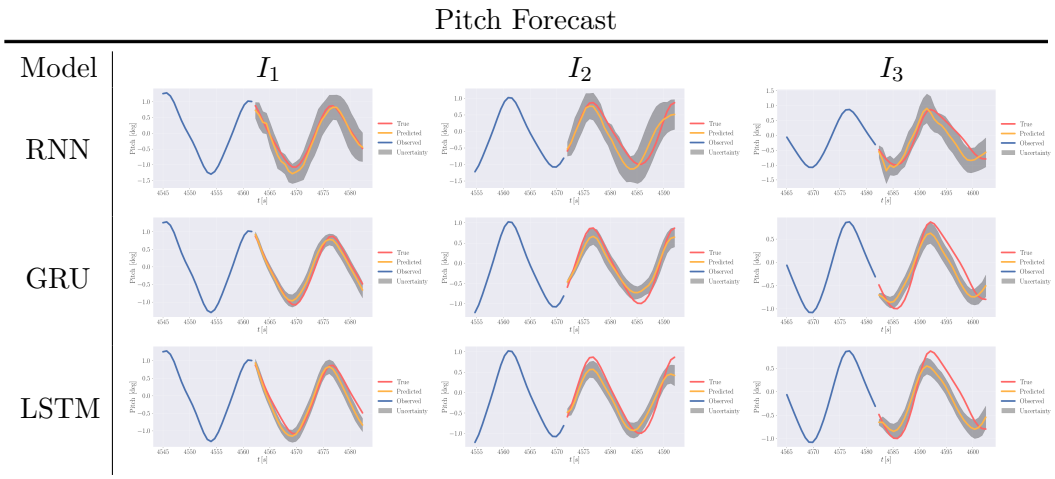


Figure 4.10. Forecast of the Pitch variable for the three chosen time intervals.

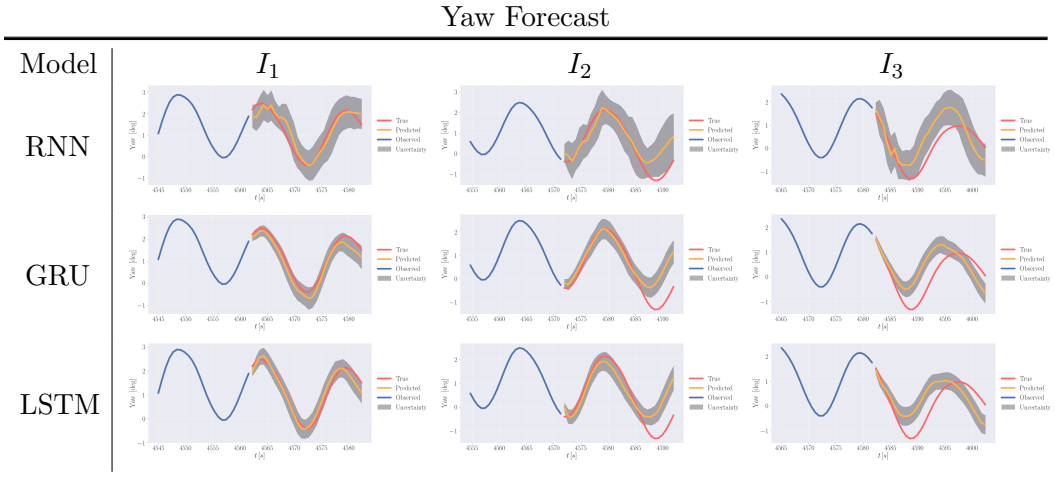


Figure 4.11. Forecast of the Yaw variable for the three chosen time intervals.

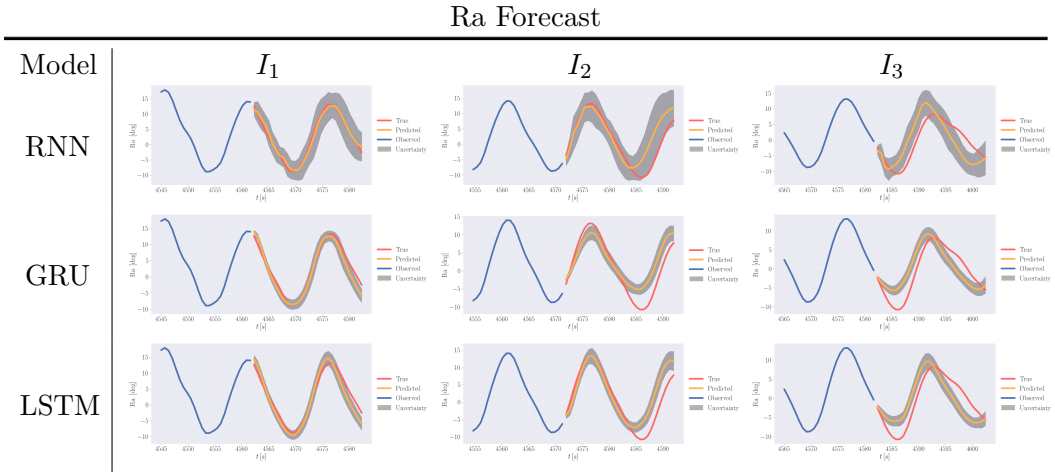


Figure 4.12. Forecast of the Ra variable for the three chosen time intervals.

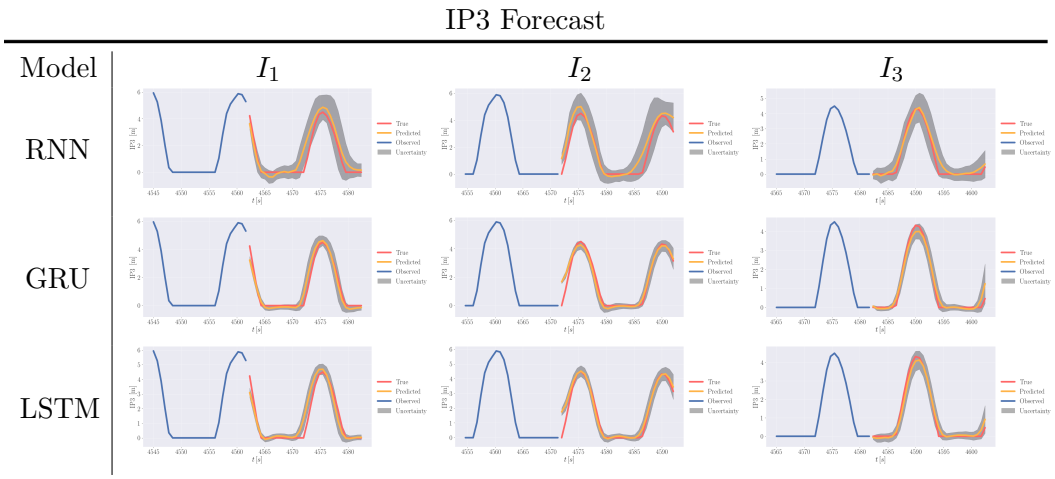


Figure 4.13. Forecast of the IP3 variable for the three chosen time intervals.

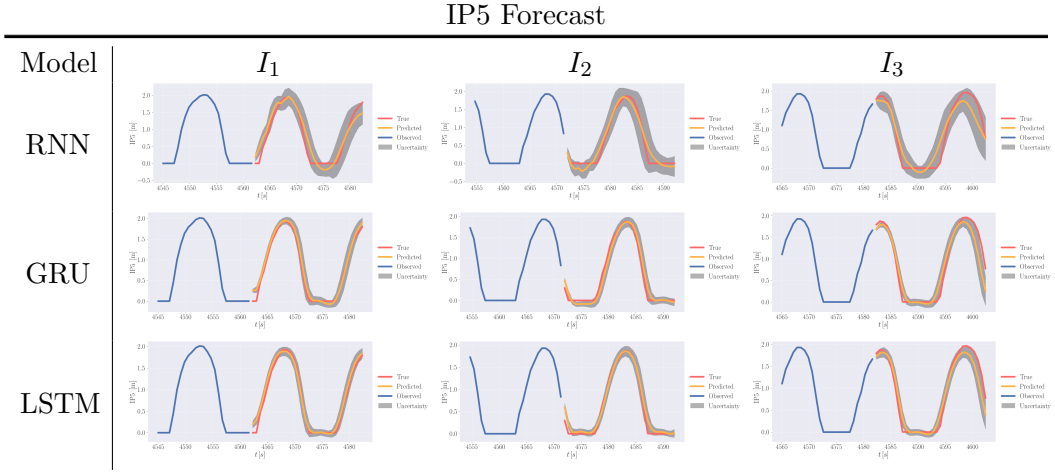


Figure 4.14. Forecast of the IP5 variable for the three chosen time intervals.

Bibliography

[1] Aggarwal, C. C. et al. *Neural networks and deep learning*. Springer (2018).

[2] Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pp. 6158–6169 (2019).

[3] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, **75** (2009), 245–248. Available from: `https://doi.org/10.1007/s10994-009-5103-0`, `doi:10.1007/s10994-009-5103-0`.

[4] Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics (2007).

[5] Barber, D. and Bishop, C. M. Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, **168** (1998), 215.

[6] Bartholomew, D. J. The foundations of factor analysis. *Biometrika*, **71** (1984), 221.

[7] Bartholomew, D. J. Spearman and the origin and development of factor analysis. *British Journal of Mathematical and Statistical Psychology*, **48** (1995), 211.

[8] Barwey, S., Raman, V., and Steinberg, A. M. Data-driven reduction and decomposition via time-axis clustering. In *AIAA Scitech 2020 Forum*, p. 0365 (2020).

[9] Bassanini, P., Bulgarelli, U., Campana, E. F., and Lalli, F. The wave resistance problem in a boundary integral formulation. *Surveys on Mathematics for Industry*, **4** (1994), 151.

[10] Benzi, R., Paladin, G., and Vulpiani, A. Power spectra in two-dimensional turbulence. *Physical Review A*, **42** (1990), 3654.

[11] Berkooz, G., Holmes, P., and Lumley, J. L. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, **25** (1993), 539.

[12] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006). ISBN 0387310738.

[13] Bletzinger, K.-U. and Maute, K. Towards generalized shape and topology optimization. *Engineering Optimization*, **29** (1997), 201.

[14] Bloor, M. I. G. and Wilson, M. J. Efficient parameterization of genetic aircraft geometry. *Journal of Aircraft*, **32** (1995), 1269.

[15] Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *Siam Review*, **60** (2018), 223.

[16] Broyden, C. G. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, **6** (1970), 76.

[17] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, **16** (1995), 1190.

[18] Chen, X., Diez, M., Kandasamy, M., Zhang, Z., Campana, E. F., and Stern, F. High-fidelity global optimization of shape design by dimensionality reduction, metamodels and deterministic particle swarm. *Engineering Optimization*, **47** (2015), 473.

[19] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, (2014).

[20] Cizmas, P. G., Palacios, A., O'Brien, T., and Syamlal, M. Proper-orthogonal decomposition of spatio-temporal patterns in fluidized beds. *Chemical Engineering Science*, **58** (2003), 4417 .

[21] Clément, S. A., Guillemain, A., McCleney, A. B., and Bardet, P. M. Options for refractive index and viscosity matching to study variable density flows. *Experiments in Fluids*, **59** (2018).

[22] D'Agostino, D., Serani, A., Campana, E. F., and Diez, M. Deep autoencoder for off-line design-space dimensionality reduction in shape optimization. In *56th AIAA Aerospace Sciences Meeting, SciTech 2018*. Gaylord Palms, Kissimmee, Florida, USA, January 8-12 (2018).

[23] D'Agostino, D., Serani, A., Campana, E. F., and Diez, M. Nonlinear methods for design-space dimensionality reduction in shape optimization. In *Machine Learning, Optimization, and Big Data. MOD 2017. Lecture Notes in Computer Science, vol 10710* (edited by G. Nicosia, P. Pardalos, G. Giuffrida, and R. Umeton), pp. 121–132. Springer International Publishing, Cham (2018). ISBN 978-3-319-72926-8.

[24] D'Agostino, D., Serani, A., Campana, E. F., and Diez, M. Augmented design-space exploration by nonlinear dimensionality reduction methods. In *Machine Learning, Optimization, and Data Science. LOD 2018. Lecture Notes in Computer Science, vol 11331* (edited by G. Nicosia, P. Pardalos, G. Giuffrida, R. Umeton, and V. Sciacca), pp. 154–165. Springer International Publishing, Cham (2019). ISBN 978-3-030-13709-0.

[25] D'Agostino, D., Serani, A., and Diez, M. On the combined effect of design-space dimensionality reduction and optimization methods on shape optimization efficiency. In *19th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (MA&O), AVIATION 2018*. Atlanta, GA, USA, June 25-29 (2018).

[26] Davis, L. Handbook of genetic algorithms. (1991).

[27] Dawson, C. W. A practical computer method for solving ship-wave problems. In *Proceedings of the 2nd International Conference on Numerical Ship Hydrodynamics*, pp. 30–38. Berkeley (1977).

[28] De Masi, G., Gaggiotti, F., Bruschi, R., and Venturi, M. Ship motion prediction by radial basis neural networks. In *2011 IEEE Workshop On Hybrid Intelligent Models And Applications*, pp. 28–32. IEEE (2011).

[29] del Águila Ferrandis, J., Triantafyllou, M. S., Chryssostomidis, C., and Karniadakis, G. E. Learning functionals via LSTM neural networks for predicting vessel dynamics in extreme sea states. *Proceedings of the Royal Society A*, **477** (2021), 20190897.

[30] Demo, N., Tezzele, M., and Rozza, G. A non-intrusive approach for the reconstruction of pod modal coefficients through active subspaces. *Comptes Rendus Mécanique*, **347** (2019), 873.

[31] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, **39** (1977), 1.

[32] Di Pillo, G., Liuzzi, G., Lucidi, S., Piccialli, V., and Rinaldi, F. A direct-type approach for derivative-free constrained global optimization. (2016).

[33] Diez, M., Campana, E. F., and Stern, F. Design-space dimensionality reduction in shape optimization by Karhunen–Loève expansion. *Computer Methods in Applied Mechanics and Engineering*, **283** (2015), 1525.

[34] Diez, M., Campana, E. F., and Stern, F. Development and evaluation of hull-form stochastic optimization methods for resistance and operability. In *Proceedings of the 13th International Conference on Fast Sea Transportation, FAST 2015*. Washington, D.C., USA (2015).

[35] Diez, M., Campana, E. F., and Stern, F. Stochastic optimization methods for ship resistance and operational efficiency via cfd. *Structural and Multidisciplinary Optimization*, **57** (2018), 735.

[36] Diez, M., Serani, A., Campana, E. F., and Stern, F. Assessing the ability to optimize hull forms of sea vehicles for best performance in a sea environment, STO-TR-AVT-204, chapter 3: INSEAN/UI optimization approach. Tech. rep., NATO (2018).

[37] Diez, M., Serani, A., Campana, E. F., and Stern, F. Stochastic design optimization for naval and aero military vehicles, STO-TR-AVT-252, chapter 7: Reliability-based robust hull-form optimization of a naval destroyer in waves. Tech. rep., NATO (2018).

[38] Diez, M., Serani, A., Campana, E. F., Volpi, S., and Stern, F. Design space dimensionality reduction for single- and multi-disciplinary shape optimization. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization (MA&O), AVIATION 2016*. Washington D.C., USA, June 13-17 (2016).

[39] Digabel, S. L. and Wild, S. M. A taxonomy of constraints in simulation-based optimization. *arXiv preprint arXiv:1505.07881*, (2015).

[40] Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. Clustering large graphs via the singular value decomposition. *Machine learning*, **56** (2004), 9.

[41] D'Agostino, D., Andre, M., Bardet, P., Serani, A., Felli, M., and Diez, M. Observing piv measurements through the lens of data clustering.

[42] D'Agostino, D., Serani, A., and Diez, M. Design-space assessment and dimensionality reduction: An off-line method for shape reparameterization in simulation-based optimization. *Ocean Engineering*, **197** (2020), 106852.

[43] Felli, M. and Falchi, M. A parametric survey of propeller wake instability mechanisms by detailed flow measurement and time resolved visualizations. In *Proceedings of the 32nd Symposium on Naval Hydrodynamics, Hamburg, Germany* (2018).

[44] Felli, M. and Falchi, M. Propeller wake evolution mechanisms in oblique flow conditions. *Journal of Fluid Mechanics*, **845** (2018), 520.

[45] Felli, M., Falchi, M., and Pereira, F. J. A. Distance effect on the behavior of an impinging swirling jet by piv and flow visualizations. *Experiments in Fluids*, **48** (2010), 197.

[46] Gablonsky, J. M. and Kelley, C. T. A locally-biased form of the direct algorithm. *J. of Global Optimization*, **21** (2001), 27–37. Available from: `https://doi.org/10.1023/A:1017930332101`, `doi:10.1023/A:1017930332101`.

[47] Gal, Y. Uncertainty in deep learning.

[48] Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059 (2016).

[49] Genton, M. G. Classes of kernels for machine learning: a statistics perspective. *Journal of machine learning research*, **2** (2001), 299.

[50] Ghoman, S., Wang, Z., Chen, P., and Kapania, R. A pod-based reduced order design scheme for shape optimization of air vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, p. 1808 (2012).

[51] Gibson, M. Spectra of turbulence in a round jet. *Journal of Fluid Mechanics*, **15** (1963), 161.

[52] Gnedenko, B. Sur la distribution limite du terme maximum d'une serie aleatoire. *Annals of Mathematics*, **44** (1943), 423.

[53] Golub, G. H. and Reinsch, C. Singular value decomposition and least squares solutions. *Numerische mathematik*, **14** (1970), 403.

[54] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, vol. 1. MIT press Cambridge (2016).

[55] Gordeyev, S. V. and Thomas, F. O. Coherent structure in the turbulent planar jet. part 2. structural topology via pod eigenmode projection. *Journal of Fluid Mechanics*, **460** (2002), 349. doi:10.1017/S0022112002008364.

[56] Grønlund, A., Larsen, K. G., Mathiasen, A., Nielsen, J. S., Schneider, S., and Song, M. Fast exact k-means, k-medians and bregman divergence clustering in 1d. *arXiv preprint arXiv:1701.07204*, (2017).

[57] Haftka, R. T. and Grandhi, R. V. Structural shape optimization-a survey. *Computer Methods in Applied Mechanics and Engineering*, **57** (1986), 91.

[58] Hansen, N. and Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, **9** (2001), 159.

[59] Hicks, R. M. and Henne, P. A. Wing design by numerical optimization. *Journal of Aircraft*, **15** (1978), 407.

[60] Hinton, G. E. and Van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13 (1993).

[61] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, **9** (1997), 1735.

[62] Holmes, P., Lumley, J. L., Berkooz, G., and Rowley, C. W. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press (2012).

[63] Hornik, K., Stinchcombe, M., White, H., et al. Multilayer feedforward networks are universal approximators.

[64] Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, **24** (1933), 417.

[65] Huang, J., Carrica, P. M., and Stern, F. Semi-coupled air/water immersed boundary approach for curvilinear dynamic overset grids with application to ship hydrodynamics. *International Journal for Numerical Methods in Fluids*, **58** (2008), 591.

[66] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, (2015).

[67] Jain, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, **31** (2010), 651.

[68] Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research*, **37** (1989), 865.

[69] Jones, D., Perttunen, C., and Stuckman, B. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, **79** (1993), 157.

[70] Jones, D. R. and Martins, J. R. The direct algorithm: 25 years later. *Journal of Global Optimization*, (2020), 1.

[71] Kendall, M. A course in the geometry of n-dimensions. *The Statistician*, **12** (1962), 337.

[72] Kennedy, J. and Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, pp. 1942–1948. IEEE (1995).

[73] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, (2016).

[74] Ketchen, D. J. and Shook, C. L. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic Management Journal*, **17** (1996), 441.

[75] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, (2014).

[76] Kotz, S. and Nadarajah, S. *Extreme value distributions: theory and applications*. World Scientific (2000).

[77] Leiva, J. P. and Watson, B. C. Automatic generation of basis vectors for shape optimization in the GENESIS program. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization Conference Proceedings, AIAA*, pp. 1115–1122 (1998).

[78] Liang, J. J., Qu, B. Y., and Suganthan, P. N. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. (2013).

[79] Liu, Q., Yang, G., Zhang, Z., and Zeng, J. Improving the convergence rate of the direct global optimization algorithm. *Journal of Global Optimization*, **67** (2017), 851.

[80] Liu, Q., Zeng, J., and Yang, G. Mrdirect: a multilevel robust direct algorithm for global optimization problems. *Journal of Global Optimization*, **62** (2015), 205.

[81] Liuzzi, G., Lucidi, S., and Piccialli, V. A direct-based approach exploiting local minimizations for the solution of large-scale global optimization problems. *Computational Optimization and Applications*, **45** (2010), 353.

[82] Liuzzi, G., Lucidi, S., and Piccialli, V. A partition-based global optimization algorithm. *Journal of Global Optimization*, **48** (2010), 113.

[83] Lloyd, S. Least squares quantization in PCM. *IEEE transactions on information theory*, **28** (1982), 129.

[84] MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, **4** (1992), 448.

[85] Mahalanobis, P. C. On the generalized distance in statistics. National Institute of Science of India (1936).

[86] Marinò, A. and Bucci, V. Shape optimization by eans of proper orthogonal decomposition and dynamic mode decomposition. In *Technology and Science for the Ships of the Future: Proceedings of NAV 2018: 19th International Conference on Ship & Maritime Research*, p. 212. IOS Press (2018).

[87] Mladineo, R. H. An algorithm for finding the global maximum of a multimodal, multivariate function. *Mathematical Programming*, **34** (1986), 188.

[88] Mockus, J. On the pareto optimality in the context of lipschitzian optimization. *Informatica*, **22** (2011), 521. doi:10.15388/Informatica.2011.340.

[89] Mockus, J., Paulavičius, R., Rusakevičius, D., Šešok, D., and Žilinskas, J. Application of reduced-set pareto-lipschitzian optimization to truss optimization. *Journal of Global Optimization*, **67** (2017), 425.

[90] Mockus, J., Tiesis, V., and Zilinskas, A. The application of bayesian methods for seeking the extremum. *Towards global optimization*, **2** (1978), 2.

[91] Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318 (2013).

[92] Paulavičius, R., Sergeyev, Y. D., Kvasov, D. E., and Žilinskas, J. Globally-biased birect algorithm with local accelerators for expensive global optimization. *Expert Systems with Applications*, **144** (2020), 113052.

[93] Pearson, K. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2** (1901), 559.

[94] Pickett Jr., R. M., Rubistein, M. F., and Nelson, R. B. Automated structural synthesis using a reduced number of design coordinates. *AIAA Journal*, **11** (1973), 489.

[95] Piegl, L. and Tiller, W. Curve and surface constructions using rational b-splines. *Computer-aided design*, **19** (1987), 485.

[96] Piegl, L. and Tiller, W. *The NURBS book*. Springer Science & Business Media (1996).

[97] Poole, D. J., Allen, C. B., and Rendall, T. C. S. High-fidelity aerodynamic shape optimization using efficient orthogonal modal design variables with a constrained global optimizer. *Computers & Fluids*, **143** (2017), 1 .

[98] Posa, A., Broglia, R., Felli, M., Falchi, M., and Balaras, E. Characterization of the wake of a submarine propeller via large-eddy simulation. *Computers and Fluids*, **184** (2019), 138 .

[99] Price, W. L. A controlled random search procedure for global optimisation. *The Computer Journal*, **20** (1977), 367.

[100] Raghavan, B., Breitkopf, P., Tourbier, Y., and Villon, P. Towards a space reduction approach for efficient structural shape optimization. *Structural and Multidisciplinary Optimization*, **48** (2013), 987–1000.

[101] Raghavan, B., Xiang, L., Breitkopf, P., Rassineux, A., and Villon, P. Towards simultaneous reduction of both input and output spaces for interactive simulation-based structural design. *Comput. Methods Appl. Mech. Engrg.*, **265** (2013), 174.

[102] Rasmussen, C. E. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer (2003).

[103] Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, (1951), 400.

[104] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, **65** (1958), 386.

[105] Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, **20** (1987), 53.

[106] Roweis, S. Em algorithms for pca and spca. *Advances in neural information processing systems*, **10** (1997), 626.

[107] Roweis, S. and Ghahramani, Z. A unifying review of linear gaussian models. *Neural computation*, **11** (1999), 305.

[108] Rozvany, G. I., Zhou, M., and Birker, T. Generalized shape optimization without homogenization. *Structural optimization*, **4** (1992), 250.

[109] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, **323** (1986), 533.

[110] Samareh, J. A. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA journal*, **39** (2001), 877.

[111] Schlichting, H. and Gersten, K. *Boundary-Layer Theory*. Springer-Verlag, Berlin (2000).

[112] Schmid, P. J. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, **656** (2010), 5.

[113] Schmit Jr, L. A. and Miura, H. Approximation concepts for efficient structural synthesis. (1976).

[114] Schneider, I. Abraham de moivre, the doctrine of chances (1718, 1738, 1756). In *Landmark Writings in Western Mathematics 1640-1940*, pp. 105–120.

[115] Sederberg, T. W. and Parry, S. R. Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, **20** (1986), 151.

[116] Seneta, E. *Regularly varying functions*, vol. 508. Springer (2006).

[117] Serani, A., D'Agostino, D., Campana, E. F., and Diez, M. Assessing the interplay of shape and physical parameters by nonlinear dimensionality reduction methods. In *Proceedings of the 32nd Symposium on Naval Hydrodynamics, Hamburg, Germany* (2018).

[118] Serani, A., D'Agostino, D., Campana, E. F., and Diez, M. Assessing the interplay of shape and physical parameters by unsupervised nonlinear dimensionality reduction methods. *Journal of Ship Research*, (2019), 1. In press.

[119] Serani, A., Diez, M., van Walree, F., and Stern, F. URANS simulations of a free-running destroyer sailing in irregular stern-quartering waves at sea state 7. *Ocean Engineering*, (2021). Under review.

[120] Serani, A., Durante, D., Diez, M., D'Agostino, D., Clement, S., Badra, J., Andre, M., Habukawa, M., and Bardet, P. Piv data clustering of a buoyant jet in a stratified environment. In *57th AIAA Aerospace Sciences Meeting, SciTech 2019*. Manchester Grand Hyatt San Diego, San Diego, California, January 7-11 (2019).

[121] Serani, A., Fasano, G., Liuzzi, G., Lucidi, S., Iemma, U., Campana, E. F., Stern, F., and Diez, M. Ship hydrodynamic optimization by local hybridization of deterministic derivative-free global algorithms. *Applied Ocean Research*, **59** (2016), 115 .

[122] Sergeyev, Y. D. On convergence of "divide the best" global optimization algorithms. *Optimization*, **44** (1998), 303. doi:10.1080/02331939808844414.

[123] Sergeyev, Y. D. and Kvasov, D. E. Global search based on efficient diagonal partitions and a set of lipschitz constants. *SIAM Journal on Optimization*, **16** (2006), 910.

[124] Sergeyev, Y. D. and Kvasov, D. E. Global search based on efficient diagonal partitions and a set of lipschitz constants. **16** (2006), 910–937. Available from: https://doi.org/10.1137/040621132, doi:10.1137/040621132.

[125] Shubert, B. O. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, **9** (1972), 379.

[126] Sieger, D., Menzel, S., and Botsch, M. Rbf morphing techniques for simulation-based design optimization. *Engineering with Computers*, **30** (2014), 161.

[127] Sieger, D., Menzel, S., and Botsch, M. *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, chap. On Shape Deformation Techniques for Simulation-Based Design Optimization, pp. 281–303. Springer International Publishing, Cham (2015). ISBN 978-3-319-06053-8.

[128] Silverman, B. W. *Density estimation for statistics and data analysis*. Routledge (2018).

[129] Simonoff, J. S. *Smoothing methods in statistics*. Springer Science & Business Media (2012).

[130] Sirovich, L. Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of applied mathematics*, **45** (1987), 561.

[131] Sobieszczanski-Sobieski, J. and Haftka, R. T. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, **14** (1997), 1.

[132] Spearman, C. " general intelligence" objectively determined and measured. (1961).

[133] Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, (2009).

[134] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, **15** (2014), 1929.

[135] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, **27** (2014), 3104.

[136] Tchebycheff, P. Sur les résidus intégraux qui donnent des valeurs approchées des intégrales. *Acta Mathematica*, **12** (1889), 287.

[137] Tennekes, H. and Lumley, J. L. *A first course in turbulence* (2018).

[138] Tipping, M. E. and Bishop, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61** (1999), 611.

[139] Tsinober, A. *An informal introduction to turbulence*, vol. 63. Springer Science & Business Media (2001).

[140] Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., and Kutz, J. N. On dynamic mode decomposition: Theory and applications. *arXiv preprint arXiv:1312.0041*, (2013).

[141] van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, **9** (2008), 2579.

[142] Van der Vaart, A. W. *Asymptotic statistics*, vol. 3. Cambridge university press (2000).

[143] van Walree, F., Serani, A., Diez, M., and Stern, F. Prediction of heavy weather seakeeping of a destroyer hull form by means of time domain panel and cfd codes.

[144] Von Mises, R. La distribution de la plus grande de n valuers. *Rev. math. Union interbalcanique*, **1** (1936), 141.

[145] Wang, H. and Song, M. Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. *The R journal*, **3** (2011), 29.

[146] Wood, G. and Zhang, B. Estimation of the lipschitz constant of a function. *Journal of Global Optimization*, **8** (1996), 91.

[147] Woodbury, M. A. *Inverting modified matrices*. Statistical Research Group (1950).

[148] Wu, J., Wang, J., Xiao, H., and Ling, J. Visualization of high dimensional turbulence simulation data using t-SNE. In *19th AIAA Non-Deterministic Approaches Conference*, p. 1770 (2017).

[149] Wu, X. Optimal quantization by matrix searching. *Journal of algorithms*, **12** (1991), 663.

[150] Yang, X.-S. *Handbook of Solid Modeling*. McGraw-Hill (1995).

[151] Zhang, B. Topics in lipschitz global optimisation. (1995).

[152] Zhou, X. and Hitt, D. L. Proper orthogonal decomposition analysis of coherent structures in a transient buoyant jet. *Journal of Turbulence*, **5** (2004), N28. doi:10.1088/1468-5248/5/1/028.