A SEQUENTIAL ADAPTIVE SAMPLING TECHNIQUE BASED ON A LOCAL LINEAR MODEL FOR COMPUTER EXPERIMENT APPLICATIONS

A Dissertation
Presented to
The Academic Faculty

By

Andrea Garbo

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology

May 2019

A SEQUENTIAL ADAPTIVE SAMPLING TECHNIQUE BASED ON A LOCAL LINEAR MODEL FOR COMPUTER EXPERIMENT APPLICATIONS

Approved by:

Professor Brian J. German, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Professor Graeme J. Kennedy
School of Aerospace Engineering
Georgia Institute of Technology

Professor Edgar G. Lightsey
School of Aerospace Engineering
Georgia Institute of Technology

Professor Dimitri N. Mavris
School of Aerospace Engineering
Georgia Institute of Technology

Professor Daniel W. Apley
School of Engineering
Northwestern University

Date Approved: March 27, 2019

To my Family,

for all the support and motivation they gave me during my first thirty-one years of life.

“Family is not an important thing. It’s everything.” (Michael J. Fox)

Alla mia famiglia,

per il supporto e le motivazioni che mi avete dato in questi primi trentuno anni di vita.

“La famiglia non e’ importante, e’ tutto.” (Michael J. Fox)

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Brian German who decided in Spring 2013 to gave me the opportunity of joining his research group. I found on him an extremely valuable advisor and an authentic friend that I will keep, I am sure, for the rest of my life. He was a priceless guide for my academic and personal growth during my period as a PhD student at Georgia Tech. He continuously supported my study and research with passion, motivation and knowledge, and I cannot omit to mention his extreme patience in reviewing my Italianized articles and thesis. If I succeed in my future professional career, I will be grateful for his advices and knowledge that he gave me during the last six years at Georgia Tech.

I have to thank the incredible group of friends that made the Weber 107 a fantastic place to work. Collin, Michael, David, Youngjun, Marc & Rachel, Xiaofan, Emre, Mark, and Matthew, I am glad that I had the opportunity of sharing with you my experience at Georgia Tech. A special mention to the Italians Matteo & Francesca, Principio, Marc, and Antonia, with whom I spent real Italian-style moments.

A very special gratitude goes to my family: my mom Gigliola, my dad Giancarlo, my brother Francesco, my sister-in-law Chiara, and my aunt Ives. They accepted my decision of spending a long period of time on the other side of the World, and they continuously supported and encouraged me throughout these six years at Georgia Tech. This successfully dissertation would not have been possible without their excellent work as a "family" that they did during my first thirty-one years of life.

Last but not the least, I need to thank my girlfriend Giada who has been a precious companion during this six-year-long journey as a Georgia Tech PhD student. I shared with her all the positive and negative moments of this unforgettable experience, and this incredible accomplishment would have been impossible without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this dissertation research is to develop a model independent sequential adaptive sampling technique for surrogate model (SM) applications based on a local linear model. This technique, called Nearest Neighbors Adaptive Sampling (NNAS), is conceived to be conceptually simple, computationally robust, and easy to apply, all characteristics that are crucial for effective surrogate modeling application during early phases of the engineering design process. SMs are now regarded as powerful engineering tools for the approximation of expensive responses – obtained either from computer simulations or real experiments – via less computationally expensive mathematical models. The use of SMs is especially valuable during the preliminary design phase when engineers need fast and accurate tools to assess the performance of different configurations and to define the top-level specifications that will guide the entire design process. Due to the increasing importance of SMs, new strategies are continuously being devised to build more flexible SM formulations, to rigorously select an SM technique from a set of candidates, and to efficiently sample the design space to collect the data required to train an SM.

The considerable influence of the sample distribution on SM accuracy motivates efforts to develop advanced strategies to improve for the sampling process. In particular, the adoption of sequential adaptive sampling techniques has been empirically shown to reduce the number of samples required to obtain an SM of specified accuracy. However, these techniques are typically challenging to implement, limited by assumptions about the response, and dependent on the SM formulation selected to supervise the sampling process (e.g. cross validation and Kriging based strategies), making them impracticable for most engineering design applications. In particular, *model dependence* – a common characteristic of most state-of-the-art adaptive sequential sampling techniques – may decrease the sampling efficiency if the guiding SM is inappropriately chosen.

The proposed NNAS technique avoids the limitations of model dependence by introducing a new refinement metric – the Non Linearity Index (NLI) – which estimates the local nonlinear characteristics as the difference between the actual response value $f(\mathbf{x}_{T,i})$ and the local function approximation represented by the hyperplane obtained via weighted least squares regression of the closest $D + k$ points in the neighborhood of $\mathbf{x}_{T,i}$, where D is the domain dimensionality. The use of local linear models to assess the nonlinear characteristics of the response without the need for a global SM is the key characteristic of NNAS that makes this strategy *model independent*. Additionally, NNAS introduces a new stochastic Pareto-ranking-based selection criterion to simultaneously maximize the refinement and exploration of the design space search, thereby ensuring a balance between the two behaviors. The initial NNAS and NLI formulations have also been expanded to include a form of *directional sampling* in which the algorithm identifies both region and direction of sampling.

NNAS embodies the capabilities of sampling multi-response design spaces, working in batch-mode (i.e. adding more than one sample at time), and continuing the sampling process even in the event of a critical error in the f evaluation, e.g. the lack of convergence of a computational model at points in the design space. These characteristics together with its ease of implementation make NNAS a valuable, efficient and robust sampling strategy to use during the early phases of engineering design.

A comprehensive set of test cases from low to high complexity and from low to high dimensionality is used to compare the performance of the proposed technique with that of other state-of-the-art sampling strategies, specifically a standard Latin hypercube design, a model dependent technique, and a model independent strategy. Results show the sampling effectiveness and the reduced computational cost of NNAS even in multi-response applications or in situations of function evaluation failure.

CHAPTER 1
INTRODUCTION

The use of high fidelity numerical models and simulations in place of costly and time consuming real life experiments is now a common practice in engineering applications. Computer simulations such as computational fluid dynamics (CFD) [1, 2, 29, 88] and finite element method (FEM) [57, 106] are useful throughout the design process and particularly during the preliminary design phases [15, 101]. The field of research that is active in proposing new methodology to integrate high-fidelity numerical models into the design process is usually called simulation-based design (SBD) ([9, 15, 17, 91, 100]). Unfortunately, although computational power is continually growing, developers have correspondingly increased the fidelity and associated complexity of these numerical models, making their use still expensive for the conceptual and preliminary phases of complex engineered system design processes [30, 47]. A famous example is reported in [36, 40], where Ford Motor Company describes how a single crash simulation of a car can take from 36 to 100 hours. The need to find a tradeoff between accuracy and computational cost of models for early design phase usage has led to the active field of research in surrogate models (SM) [46, 87, 98, 103], which are now popular and powerful tools for engineering modeling applications. The set of operations needed to "create" an SM is called *the surrogate modeling process* which requires the completion of several steps including training data collection, SM selection and training, and SM validation. As widely reported in the literature [30, 31, 87, 102, 105], the techniques adopted to execute each of these steps can substantively affect the accuracy of the final SM, and therefore, each phase must be carefully carried out.

This dissertation is focused on the data collection phase (*sampling*) that is responsible for collecting pertinent information about the high-fidelity response at different locations in the design space. Previous studies [18, 20, 30, 31, 37] have shown how an inappropriate

design space sampling may lead to a completely inaccurate SM. As a consequence, several sampling strategies have been developed with the objective of efficiently collecting as much information as possible with the fewest required high-fidelity simulation runs to reduce the overall surrogate modeling time. However, these techniques are typically conceptually and computationally complex, limited by assumptions about the response, and dependent on the SM formulation selected to supervise the sampling process (model dependence). *Model dependence* is a common characteristic of most of state-of-the-art techniques that has been shown [30, 31] to deteriorate the sampling efficiency if the guiding SM is inaccurately chosen. A comprehensive literature review has revealed the need for a sampling strategy that is simple, robust, and computationally inexpensive in order to be suitable for early design phase applications.

The proposed technique called **Nearest Neighbors Adaptive Sampling (NNAS)** removes the model dependence limitation by introducing a new refinement metric – the Non Linearity Index (NLI) – which estimates the local nonlinear characteristics of the response from the training data. NLI at every training point $\mathbf{x}_{T,i}$ is computed as the prediction error between the actual response value $f(\mathbf{x}_{T,i})$ and the local function approximation represented by the hyperplane obtained via weighted least squares regression of the closest $D+k$ points in the neighborhood of $\mathbf{x}_{T,i}$ (where D is the domain dimensionality). The use of local linear models to assess the nonlinear characteristics of the response without the need of a global SM is the key characteristic of NNAS that makes this strategy *model independent*. Additionally, NNAS introduces a new stochastic Pareto-ranking-based selection criterion to simultaneously maximize the refinement and exploration of the design space search, thereby ensuring a balance between the two behaviors. The initial NNAS and NLI formulations are have also been expanded to include a form of *directional sampling* where the algorithm identifies both the region and the direction of sampling. NNAS also includes capabilities for sampling multi-response design spaces, working in batch-mode (i.e. adding more than one sample at time), and continuing the sampling process even in the event of a critical

error in the f evaluation, e.g. a computational simulation that does not converge at a point in the design space. These characteristics together with its ease of implementation make NNAS a valuable, efficient and robust sampling strategy for early phases of engineering design.

Even though design space sampling is the primary objective of this work, a description of the entire surrogate modeling process is necessary to infer the requirements for a good sampling strategy and to understand its effects on the efficiency of the overall process. The first part of this chapter (Sections 1.1 to 1.5) provides the definition of a SM itself and a description of all the SM creation phases; particular attention is focused on aspects related to the final accuracy of the SM and to the overall modeling efficiency. The second half of the chapter briefly introduces the characteristics of the proposed technique (section 1.7) and the objectives of this work (section 1.6). Chapter 2 describes the current state-of-the-art in sampling strategies, with an assessment of the positive and negative aspects of the predominant techniques in terms of surrogate modeling efficiency. Chapter 3 states the objectives of this dissertation, including the motivations for the development of the proposed technique based on deficiencies of current methods identified in chapter 2. Chapter 4 presents a comprehensive study which highlights the effect of model dependence on sampling processes conducted by using state-of-the-art model dependent sequential adaptive strategies (MDSASs). Chapter 5 describes the NNAS formulations, algorithms, and an estimation of its algorithmic complexity. Chapter 6 presents a comprehensive set of results and analyses which stress key aspects of the NNAS strategy, including the sampling efficiency for multi-response applications, the batch operational mode and the capability of continuing the sampling even in the event of a critical error in the computer simulation solver. Finally, chapter 7 summarizes the key outcomes of this dissertation and provides suggestions for future development of the NNAS technique.

1.1 Definition and Uses of Surrogate Models

This section provides a description of surrogate models (SMs) that is sufficient to understand the material presented in this dissertation. SMs and the adopted nomenclature are defined in section 1.1.1, followed by a description of the principal SM classes for engineering applications in section 1.1.2.

1.1.1 Definition

High fidelity simulation models are typically excessively computationally expensive to be used in preliminary design phase activities such as trade space exploration, evaluation of alternative concept architectures, and design optimization. For this reason, SMs have gained considerable attention [60] because, if used judiciously, they have the potential to efficiently and accurately approximate the response of high fidelity simulations by means of simpler mathematical models. An SM has been defined by Simpson as "an efficient approximation of the analysis codes that yields insight into the functional relationship between the inputs and the outputs" [80, 103]. A similar definition is provided by Kleijnen [52] who sees a SM as "an approximation of the input/output (I/O) function that is defined by the underlying simulation model (high-fidelity model)". Likewise, surrogate modeling has been described as "the practice adopted in recent times for extracting relevant information from the outcomes of costly and lengthy experiments by determining the functional relation between the inputs and the outcome of an experiment" [64, 71]. This computationally inexpensive replacement model can be therefore used for the analysis or the optimization of complex systems while minimizing the required number of expensive simulations [44].

A surrogate model is sometimes designed as a *metamodel*, a word which literally means "model-of-a-model" [36, 50, 82]. Indeed, the metamodel is intended as a simpler representation of the base model, devised at a higher level of abstraction. This concept is represented in the example illustrated in figure 1.1. The globe is the first level of approximation (high

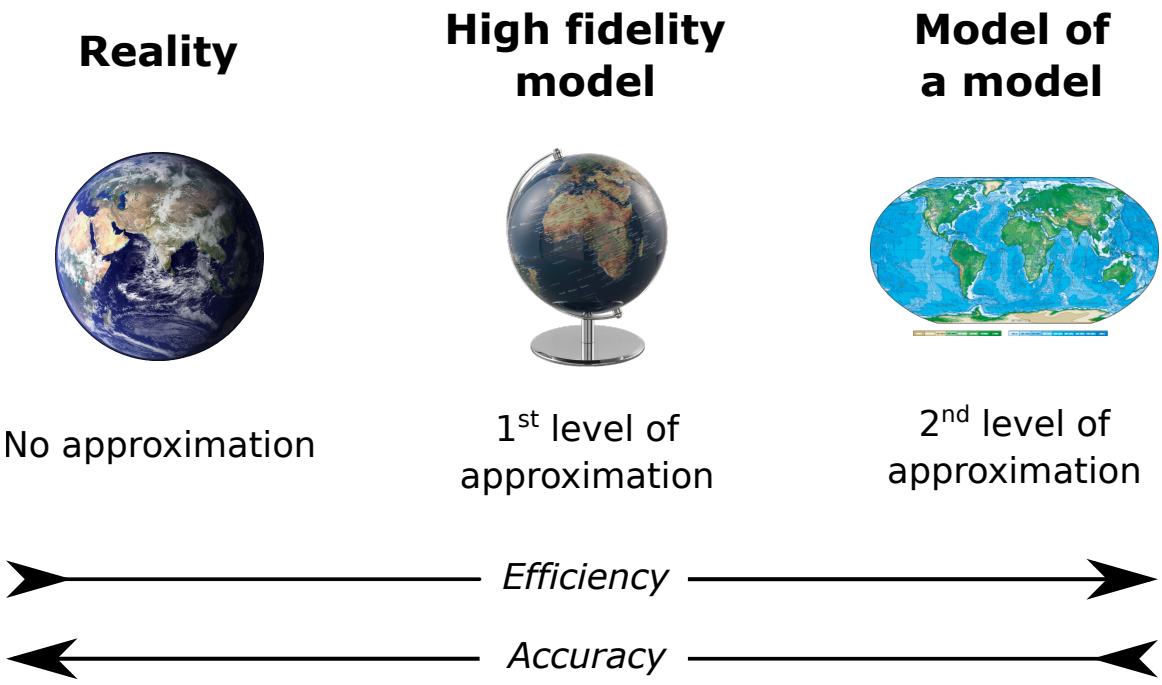


Figure 1.1: Example of "model of a model"

fidelity) of the Earth, where some characteristics such as composition, geology, and atmospheric phenomena are completely lost, but fundamental *geographic* characteristics such as scaled distances and shape are preserved. The globe makes is possible to gather geographic intuition and approximate information about distances in a easier and faster way rather than traveling around the Earth: accuracy is traded to gain efficiency. A second level of approximation is obtained in the world map. As a 2D projection, either distances or angles can be preserved throughout the map but typically not both, and the full character of the 3D spherical shape is lost: another level of accuracy is traded to gain further efficiency. The same idea of trading accuracy to gain efficiency is the motivation in engineering applications of SMs where the expensive code and the SM represent the first and second level approximations of the real physics, respectively [40, 59, 66, 81, 83].

Moving from a qualitative to a more formal mathematical description of a SM, Queipo [87] sees surrogate modeling as "a non-linear inverse problem for which one aims to determine a continuous function of a set of design variables from a limited amount of available data." Presume that the output y of an expensive computer simulation is known

at several design space locations $\mathbf{x}_{T,i}$ with $\mathbf{x} \in \mathbb{R}^D$, establishing the *training dataset* $D_T = \{\mathbf{X}_T = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_T}], \mathbf{y}_T = [y_1, \ldots, y_{n_T}]\}$. An SM is a mathematical model $\tilde{y} = \tilde{f}(\mathbf{x}, \boldsymbol{\theta})$ that is created from the limited data D_T and is used to approximate the true response $y = f(\mathbf{x})$ represented by the computer simulation. The vector $\boldsymbol{\theta}$ contains all the parameters necessary to fully define $\tilde{f}$, and these parameters are identified from D_T through a set of operations called *training process*. This definition introduces the three fundamental "ingredients" required to build an SM : the training dataset D_T (section 1.5, chapter 2), the functional form $\tilde{f}$ (section 1.4), and the training process needed to infer the vector $\boldsymbol{\theta}$ from D_T (section 1.4).

1.1.2 Local vs Global: Surrogate Model Uses in the Engineering Field

SMs are generally characterized as *local* or *global* models depending on their specific application ([37, 52, 58]). Local SMs approximate f in a small restricted region of the design space, and they are mainly used as function representations to guide and speed up optimization algorithms. Usually, local SMs are discarded and retrained multiple times during the optimization process, as the search proceeds in the design space. Examples of applications of local SMs in the context of surrogate-based analysis and optimization (SBAO) or *metamodel assisted optimization* are available in [8, 27, 49, 72, 87].

On the other hand, a global SM has the goal of representing f as accurately as possible over the entire design space, and it can be used, for example, during the preliminary design phase for assessing the performance of several different designs in a limited amount of time, filtering out regions of the design space that violate certain constraints, or identifying regions with high probability of having an optimal design. All the methods discussed in this dissertation are intended to create global SMs.

1.2 Surrogate Model Quality Assessment

As previously described, an SM is a function approximation method that trades representation accuracy to gain model evaluation efficiency. Since an SM is trained starting from limited information (D_T) about the true function f which it is intended to mimic, it is unrealistic to expect that the SM is able to perfectly match f over the entire domain. Indeed, every SM is affected by a certain degree of representation error whose estimation is crucial for a conscious use of SMs in the design process [36, 87, 103].

As described in [30, 71], SM error estimators can be classified from two different perspectives. The first perspective discerns between [69]:

- methods that require additional data (validation data D_V);
- methods that use existing data.

Error estimators of the first kind (like R^2 (A.9) or root mean squared error (RMSE) (A.5)) are usually not convenient when the evaluation of $f(\mathbf{x})$ is computationally expensive and the collection of validation data would lead to a significant additional computational time. On the other hand, error metrics which intensively reuse the available training data (like the cross validation (CV)-based error estimators in equations A.1 and A.13) do not require further function evaluations but may suffer of inductive bias caused by an excessive trust on the data.

The second differentiation is between [34]:

- global error estimator (GEE) methods that provide information about the SM quality over the entire design space;
- local error estimator (LEE) methods that return the accuracy of the SM at specific locations in the domain.

In particular, GEE metrics (e.g. R^2 in eq.(A.9) and RMSE in eq.(A.1)) do not require the specification of the location where the error has to be estimated, while LEE estimators

(e.g. relative absolute error (RAE) in eq.(A.11) and CV-variance (σ_{CV}) in eq.(A.13)) are also function of the design space location where they have to be computed. As described in section 1.4 and chapter 2 and applied in chapter 4, a GEE is a suitable metric for SM functional form selection, whereas a LEE is frequently used in model dependent adaptive sampling strategies to identify the design space locations in which sampling refinement is required.

A complete survey of SM error estimators is unnecessary for the goals of this dissertation (comprehensive reviews are available in [34, 71, 87]), but a brief description of the metrics considered in this work is provided in appendix A.

1.3 Surrogate Modeling Process

The SM definition given in section 1.1 helps to identify the fundamental phases of the surrogate modeling process (table 1.1): the creation of the training dataset D_T in the *sampling phase*, the selection of the SM functional form $\tilde{f}$ during the *SM functional form selection phase*, and the computation of the coefficient vector $\boldsymbol{\theta}$ in the *training phase* [102]. As previously illustrated (section 1.2), it is also imperative to assess the quality of the SM before using it, and this task is accomplished by computing certain error estimators – suitable GEE or LEE metrics – during the *validation phase*. Even though in many practical applications, the SM functional form is selected a priori by the user at the beginning of the surrogate modeling process without performing a formal SM selection phase, several studies [30, 31] show how the use of an inappropriate $\tilde{f}$ can have a strong negative impact on the accuracy of the final SM. Therefore, the SM selection phase should be always included in the surrogate modeling process.

The order in which these four steps are performed in the surrogate modeling process is mainly determined by the specific sampling technique and architecture (chapter 2). For example, if the sampling strategy is model dependent, and therefore it is guided by a global SM, the sampling, the SM selection, and the training phases must be carried out simul-

Table 1.1: Relations between SM creation process elements and phases

Element	Phase
Training dataset D_T	Sampling
Functional form $\tilde{f}$	Functional form selection
Training coefficient vector $\boldsymbol{\theta}$	Training
Error estimators (GEE and LEE)	Validation

taneously (section 2.3.4). With the goal of deriving suitable performance metrics for the surrogate modeling process, consider – without loss of generality – a process with a model independent sampling strategy in which the phases follow the scheme shown in figure 1.2. First, the sampling algorithm creates n_T training sample locations ($\mathbf{X}_T$), and consequentially n_T simulations are run to obtain the system responses $\mathbf{y}_T$ and to create the training set D_T; the sampling phase requires t_{sampl} time, and each simulation runtime is t_{ev}. Second, $\tilde{f}$ is selected within a set of possible candidates by the SM functional form selection phase in t_{sel} time. Third, D_T and $\tilde{f}$ are used in the training process to determine the coefficients $\boldsymbol{\theta}$ required to fully define the SM $\tilde{f}(\mathbf{x}, \boldsymbol{\theta})$, and the SM training process is completed in t_{tr} time. Finally, t_{val} is the time spent in the validation phase to assess the quality of resulting SM; depending on the chosen error estimators, additional simulations (n_V) might be needed to create the validation dataset D_V. Each phase with its relative inputs, outputs, and time is listed in table 1.2, and the overall surrogate modeling process is illustrated in figure 1.2.

Table 1.2: Phases of the surrogate modeling process: input, output, and time required

Phase	Input	Output	Time
Sampling	Design space ranges	$\mathbf{X}_T$	t_{sampl}
Simulation runs	$\mathbf{X}_T$	$\mathbf{y}_T$	$n_T t_{\text{ev}}$
$\tilde{f}$ selection	$D_T = \{\mathbf{X}_T, \mathbf{y}_T\}$	$\tilde{f}$	t_{sel}
Training	D_T,$\tilde{f}$	$\boldsymbol{\theta}$	t_{tr}
Validation	$\tilde{f}$, $\boldsymbol{\theta}$, D_V	GEE	$t_{\text{val}} + n_V t_{\text{ev}}$

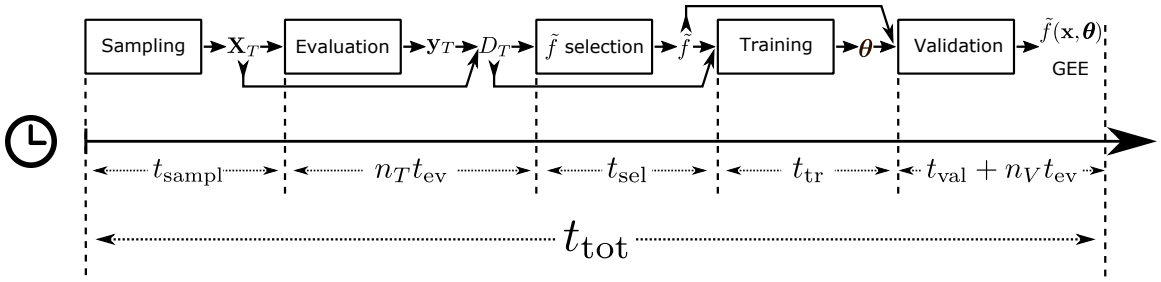


Figure 1.2: Example of surrogate modeling process with timeline

The total time (t_{tot}) required to complete the surrogate modeling process with this particular process architecture is (1.1):

$$t_{\text{tot}} = \underbrace{(t_{\text{sampl}} + n_T t_{\text{ev}})}_{\text{Sampling}} + \underbrace{t_{\text{sel}}}_{\text{Selection}} + \underbrace{t_{\text{tr}}}_{\text{Training}} + \underbrace{(t_{\text{val}} + n_V t_{\text{ev}})}_{\text{Validation}} \tag{1.1}$$

Equation (1.1) can be rearranged to highlight the impact of evaluation time t_{ev} and any other additional time t_{add} on the total process time t_{tot} as shown in equations (1.2) to (1.3):

$$t_{\text{tot}} = \overbrace{(n_T + n_V)}^{n_{\text{tot}}} t_{\text{ev}} + \overbrace{t_{\text{sampl}} + t_{\text{sel}} + t_{\text{tr}} + t_{\text{val}}}^{t_{\text{add}}(n_T, n_V)} \tag{1.2}$$

$$= n_{\text{tot}} t_{\text{ev}} + t_{\text{add}} \tag{1.3}$$

where the notation $t_{\text{add}}(n_T, n_V)$ is intended to underline the dependence of all t_{add} components on n_T and n_V. For example, t_{sampl} and t_{val} are affected by the number of samples n_T that the sampling algorithm has to define and the validation points n_V, respectively. If t_{add} is divided by n_{tot} obtaining the average additional time per single evaluation $\bar{t}_{\text{add}}$, equation (1.3) can be written as:

$$t_{\text{tot}} = n_{\text{tot}} t_{\text{ev}} \left(1 + \frac{\bar{t}_{\text{add}}(n_T, n_V)}{t_{\text{ev}}}\right) \tag{1.4}$$

$$= n_{\text{tot}} t_{\text{ev}} \left(1 + \tau(n_T, n_V)\right) \tag{1.5}$$

where $\tau = \bar{t}_{\mathrm{add}}/t_{\mathrm{ev}}$ is the ratio between the average additional computational time per function evaluation and the single simulation runtime. It is now evident from equations 1.4 and 1.5 that if the t_{tot} required to build a SM with a certain level of accuracy has to be reduced, it is not always true that this objective is achieved by using sampling strategies devised only to decrease n_{tot}, because attention should also be paid to the t_{add} required by all of surrogate modeling process operations. For example, some sophisticated adaptive sampling methods are computationally expensive, and the benefit obtained by the reduction in n_{tot} can be overshadowed by the increase in $\bar{t}_{\mathrm{add}}$, i.e. the computational overhead of the sampling strategy itself, as described by Garbo in [30, 31]. This behavior is more evident when high dimension problems are considered and the "curse of dimensionality" starts to have a significant impact on the scaling of the additional computational time.

The effect of n_{tot} and τ on t_{tot} (Equation 1.4) is one reason why simple SM techniques are usually employed for problems where the t_{ev} is low (such as wind tunnel experiments [22, 74, 76]), and advanced SM strategies are instead used for highly computationally expensive simulations (like CFD [29]). Advanced strategies are beneficial in this context because they usually require fewer n_{tot} than simple strategies at the cost of additional computational time. On the other hand, if t_{ev} is low, the additional $\bar{t}_{\mathrm{add}}$ could lead to such a τ increase that t_{tot} grows even if n_{tot} decreases, thereby making the simple strategies preferable to the advanced ones. An example is the wide use of simple factorial design sampling strategies for current wind tunnel applications [22, 74, 76], where the cost of the experiment is directly connected to the rental cost of the facility – therefore with t_{tot} –, and the "evaluation time" t_{ev} is very low. As shown in equation 1.5, a low t_{ev} requires SM techniques with low $\bar{t}_{\mathrm{add}}$ to mitigate the τ effect on t_{tot}, and given that factorial design sampling techniques have negligible $\bar{t}_{\mathrm{add}}$, they are the most appropriate for these kinds of applications.

The total cost involved in the SM creation can be represented as a linear combination of the evaluation cost and the time cost (Equations (1.6),(1.7)):

$$\$_{\text{tot}} = \overbrace{\$_{\text{ev}} n_{\text{tot}}}^{\text{Evaluation cost}} + \overbrace{\$_t t_{\text{tot}}}^{\text{Time cost}} \tag{1.6}$$

$$= \$_{\text{ev}} n_{\text{tot}} + \$_t \left[n_{\text{tot}} t_{\text{ev}} \left(1 + \tau \right) \right] \tag{1.7}$$

where $\$_{\text{tot}}$ is the total cost, $\$_{\text{ev}}$ is the cost of a single function evaluation, and $\$_t$ is the cost per unit time. When the single function evaluation is economically expensive (high $\$_{\text{ev}}$, negligible $\$_t$) and the goal is the reduction of $\$_{\text{tot}}$ – e.g. combustion experiments – Equation 1.6 indicates that the n_{tot} reduction should be the guiding criterion for the selection of the SM techniques to use. If instead $\$_{\text{ev}}$ is low and $\$_t$ is high, both n_{tot} and τ can play a crucial role to determine $\$_{\text{tot}}$, and two different scenarios are possible:

1. t_{ev} is very high implying a low value of τ (CFD simulations [29]); strategies should be selected to reduce n_{tot} as much as possible.

2. t_{ev} is very low (like in real wind tunnel tests) causing a high value of τ; a trade off between n_{tot} reduction and limited $\bar{t}_{\text{add}}$ has to be found.

As discussed earlier, the increasing interest in SM applications has led to the development of numerous techniques that claim to increase the efficiency of the surrogate modeling process [4, 87, 98, 99, 103]. Focusing on the sampling phase, the majority of previous research [60, 98, 99] mainly focused on engineering applications involving computational expensive simulations (high t_{ev}) with a resulting negligible τ effect on t_{tot}, and therefore they proposed sampling techniques with the unique objective of reducing n_{tot} as much as possible. It is opinion of the author [30, 31] that the additional time $\bar{t}_{\text{add}}$ required by advanced sampling techniques and sophisticated model training processes can significantly lower surrogate modeling efficiency, in particular for mildly computationally expensive and

high dimensional applications. Reducing the total surrogate modeling time by reducing both n_{tot} and $\bar{t}_{\text{add}}$ is one of the primary motivations for the proposed sampling technique.

1.4 Surrogate Model Functional Forms and Training Process

The objective of this section is to provide an overview of some SM classes and their training processes with the goal of discerning the key aspects affecting both the accuracy and the efficiency of the surrogate modeling process. A complete description of all SM functional forms is out of the scope of this dissertation, and good references on this topic include [4, 87, 98, 99, 103].

The first part of this section presents the SM classification based on the training process, and a simple example illustrates the impact of training complexity – and therefore t_{tr} – on the overall surrogate modeling process time t_{tot} (section 1.4.1). Secondly, section 1.4.2 enunciates the No-Free-Lunch theorem with a description of its consequences on the surrogate modeling process. Finally, some approaches for the selection of the SM functional form are briefly described at the end of section 1.4.2.

1.4.1 SM Functional Forms and Training

Commonly, SM functional forms are grouped in families or classes that share similarities in their mathematical formulation; SMs within the same family differ in terms of small adjustments that are introduced to improve the algorithm performance or to create a specific version for a specific application. A list of some of the predominant SM classes is reported in table 1.3.

A more useful SM classification for the analysis of surrogate modeling process efficiency is based on the characteristics of the training technique required by each functional form. It is important to remember that one of the additional computational time (t_{add}) components is the time required by the SM training process (t_{tr}) (Equation (1.1)). Within the available SM techniques, two different classes of training algorithms are recognizable:

Table 1.3: Classes of surrogate models

Class	**Training**	**References**
Kriging	Non-linear	[11, 67, 75, 84, 89, 113, 114]
RBF	Linear and Non-linear	[25, 43, 73]
Spline	Linear and Non-linear	[28]
Artificial Neural Network	Non-linear	[41, 95, 112]
Response surfaces	Linear	[10, 77]

linear and *non-linear* training. A *linear* training process requires the solution of a single linear problem to estimate the coefficient vector $\boldsymbol{\theta}$ from the training dataset D_T, meaning that the training process is simple, robust, and computationally inexpensive (In term of computational complexity, a linear training process can take at most $\mathcal{O}(n_T^3)$ operations). On the other hand, a *non-linear* training requires the solution of a non-linear problem by an iterative procedure, and therefore by algorithms that are more complex, less robust, and more computationally expensive than the linear solvers. This SM classification is important for the purpose of this dissertation because it qualitatively provides insights on the impact that the training process of a particular SM functional form can have on the total process time t_{tot} (Equation 1.2). Extreme attention is needed when non-linear SM are adopted because the high t_{tr} required to solve the non-linear training problem could seriously impact the total surrogate modeling time t_{tot} as studied by Garbo in [30]. Consider an example taken from [30] where a model dependent sequential adaptive strategy (MDSAS) is used to create the D_T of an eight-dimensional test function. As explained later in section 2.3.4, a MDSAS sequentially samples the design space and populates D_T by using a global SM which therefore must be selected and retrained at the beginning of each sampling iteration. In this particular example, five different SM functional forms can be actively selected by the algorithm, specifically four radial basis functions (RBFs) with linear training and an ordinary Kriging (OKRG) SM with non-linear training. Figure 1.3 reports the plot of the total process time t_{tot} required to obtain an SM with a certain accuracy ($\text{NRMSE}_{\text{CV}} < 0.05$) as

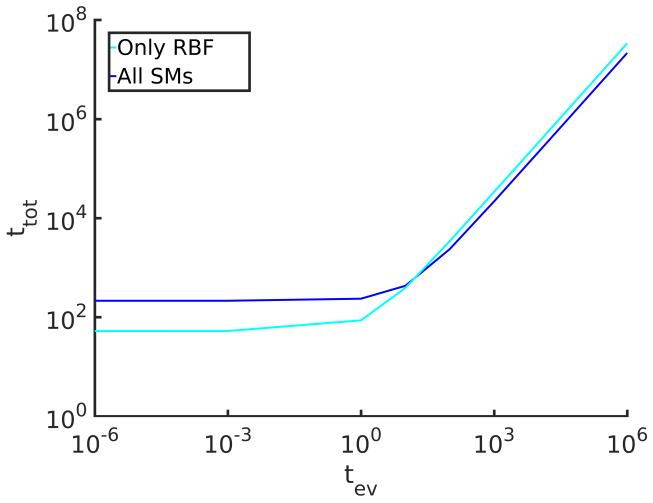


Figure 1.3: Example of impact of non-linear training time on total SM creation process time

a function of the evaluation time t_{ev}. As reported in the original study [30], the SM creation process requires an average of 22 samples when the OKRG is included in the set of available SM functional forms (All SMs), compared to the 34 required when only RBFs (Only RBF) are considered. Figure 1.3 shows how the All SMs configuration is worthwhile only for t_{ev} higher than around 10^1 sec; for smaller t_{ev}, the additional computational time required by the OKRG non-linear training at the beginning of each sampling iteration strongly impacts t_{tot} (Equation (1.2)). For this reason, in this example, the All SMs configuration is "time-inefficient" for $t_{\text{ev}} < 10^1$ even though it completes the process with fewer n_{tot} than Only RBF.

1.4.2 SM Functional Form Selection: the No-Free-Lunch Theorem

It is clear how the conspicuous number of available SM functional forms confronts the user with an elementary question: which one should I use?

A typical way to choose the SM functional form is to rely on historical data or experience about previous studies that attempted to model a similar function. For example, it is well known in the aerospace community that there is a quadratic relation between the lift and drag aerodynamics coefficients, and therefore it would be unwise and wasteful to use

an artificial neural network to model the drag polar of a NACA 4-digit airfoil. But what if previous literature is limited or even absent? The user could begin to speculate about the existence of a powerful and "general" SM formulation that is able to accurately model all possible functions; if it existed, the user would then be relieved from the burden of selecting the SM functional form. First of all, such an SM formulation would likely be tremendously complex and involve a very large number of training coefficients θ making the computational complexity prohibitive for engineering applications. Secondly, a "general" SM does not exist as indicated by the *No-Free-Lunch* theorem (NFL) [107, 108, 109].

Avoiding the mathematical and formal proof of the theorem available in [107, 108, 109], the essence of the NFL is contained in a sentence written by Wolpert in [107]:

> ...for any two learning algorithms, there are just as many situations (appropriately weighted) in which algorithm one is superior to algorithm two as vice versa, according to any of the measures of superiority.

where "learning algorithm" can be considered as synonymous to SM functional forms. In addition to the formal proof, several studies [16, 30, 31, 35, 71, 86, 105] have elucidated this limitation by testing different SM techniques on a wide range of test functions and comparing their modeling performance. For example, Garbo [30] tests four different RBF SMs and an OKRG SM with Sum Squared Exponential covariance function (OKRK-SSE) on five different test functions with dimensionality from 2 to 8. The modeling performance is assessed considering the average number of samples – generated by a maximin Latin hypercube sampling (LHS) – required to obtain a given SM quality level ($\mathrm{NRMSE_{CV}}$ and R^2) out of fifteen repetitions. The results reported in tables 1.4 and 1.5 show the percentage of cases when a specific SM reaches the quality requirement with fewer samples than all the others. As it is possible to notice, the consequences of the NFL theorem are clearly demonstrated: for both the quality metrics and for all the accuracy levels, it is impossible to identify one SM functional form that outperforms all others in all the situations. Even though the OKRG-SSE has better performance in most of the cases (70.2% for

$\mathrm{NRMSE_{CV}} = 0.05$ and 48.3% for $R^2 = 0.99$), there is still a notable portion of cases in which other SM techniques have better predictive behavior.

Table 1.4: Percentage of cases where a SM reaches specific levels of $\mathrm{NRMSE_{CV}}$ with the fewest number of samples [30]

$\mathbf{NRMSE_{CV}}$	0.10	0.08	0.07	0.05
RBF C	4.5%	3.7%	4.5%	7.0%
RBF G	0.0%	0.0%	0.9%	0.9%
RBF MQ	24.5%	24.1%	22.7%	20.2%
RBF invMQ	5.5%	3.7%	2.7%	1.8%
OKRG-SSE	65.5%	68.5%	69.1%	70.2%

Table 1.5: Percentage of cases where a SM reaches specific levels of $R^2$with the fewest number of samples [30]

$\mathbf{R^2}$	0.90	0.95	0.97	0.99
RBF C	3.5%	6.8%	12.9%	12.2%
RBF G	0.0%	0.9%	4.3%	8.7%
RBF MQ	29.8%	22.2%	20.9%	19.2%
RBF invMQ	4.4%	6.8%	9.4%	11.6%
OKRG-SSE	62.3%	63.2%	52.5%	48.3%

Similar conclusions are obtained by Garbo and German in [31] where the authors investigate the effect of NFL when a model dependent sampling strategy (i.e. it uses an SM to identify the design space region in need of sample refinement) is used to sample the design space of eleven test functions. The results summarized in the pie chart in figure 1.4 clearly indicate how it is impossible to determine an SM functional form able to perform better than others irrespective to the test function. More details about the effect of NFL on model dependent sampling performance are discussed and analyzed in chapter 4.

This set of results leads to the conclusion that an SM functional form selection strategy is needed every time an SM has to be used, and no previous studies have been conducted on

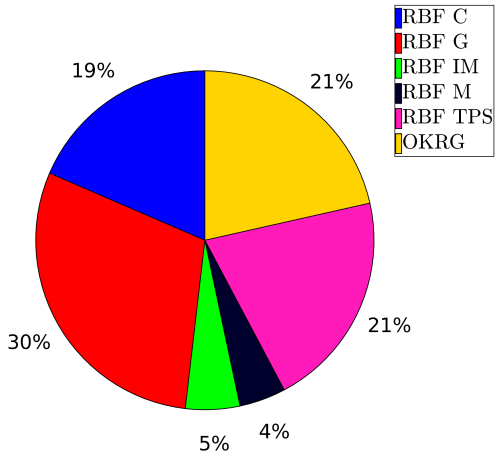
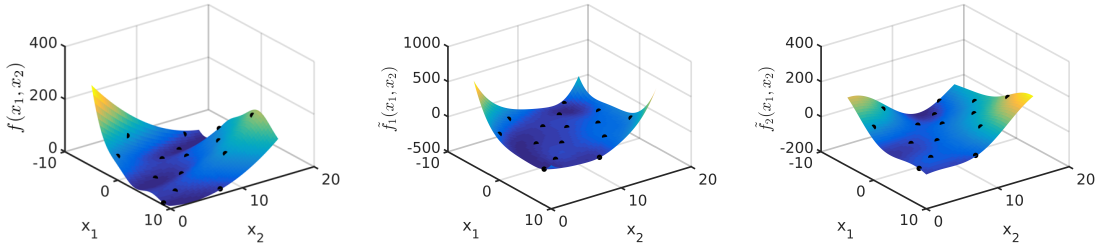


Figure 1.4: Percentage of sampling repetitions when a specific SM reaches $\mathrm{NRMSE_V} < 10^2$ with fewer samples than the other SMs considering all the test functions



(a) Branin function surface

(b) Branin SM obtained with cubic RBF

(c) Branin SM obtained with gaussian RBF

Figure 1.5: Example of two different SM functional forms applied to the same training dataset

similar problems. Even though the training dataset D_T accurately samples the design space, the use of an inappropriate SM technique can lead to a complete misinterpretation of the collected information. Figure 1.5 shows a test function response (figure 1.5a), and two SMs obtained using two different RBFs (figures 1.5b and 1.5c) and the same training dataset; the importance of an accurate selection of the functional form formulation is evident by noticing the considerable difference between the two SM surfaces.

If no historical data or experience about modeling similar responses are available, global error estimators (GEEs) are commonly used as metrics to assist in the selection of the SM technique by observing the following procedure:

1. define a set of possible candidate SM functional forms;

2. train all the candidate SM functional forms;

3. choose a GEE and evaluate it for all the trained SMs;

4. select the functional form with the lowest GEE.

Unfortunately, the "best" resulting SM can be strongly dependent on the adopted GEE, and this dependency is evidenced by the discrepancy between the results in tables 1.4 and 1.5. It may therefore seem that the difficulty of choosing the functional form has been transfered to the GEE selection. However, the choice of the GEE is usually guided by the quality requirement for the final SM: it is much easier to decide the GEE to use as an SM selection criterion than to directly choose the $\tilde{f}$ form. For example, the maximum absolute error (MAE) metric could be considered if the final SM must have the MAE under a give threshold, or normalized root mean squared error (NRMSE) could be used if an average modeling performance over the entire design space is desired. Unfortunately, the main drawback of the SM selection process is the resulting additional computational cost required by the training of all the considered SMs, in particular if non-linear SMs are included. The increase in computational time caused by the necessity of implementing an SM selection process every time an SM has to be created is magnified when model dependent sampling strategies are used [30], as described in section 2.1.3 and chapter 4.

The impossibility of finding a general SM technique implied by the No-Free-Lunch theorem, and the consequential additional computational time required by an SM selection process based on a GEE are the main motivations behind the objective in this thesis research of developing a model independent sampling technique, as described in chapter 2.

1.5 Surrogate Model Training Data

The last element of the surrogate modeling process to introduce is the *training dataset* D_T. At this point a user of SM should ask the following question: if I have the opportunity to

run a given number of experiments, how should I determine the elements of the matrix $\mathbf{X}_T$ that represent the locations of these experiments in the design space? In other words, what is a good strategy to decide the set of design points where the experiments should be run to appropriately sample the design space? The main hope behind this question is that there exists a strategy that is able to select the experiments so as to maximize the information obtained about the response. The application of such a strategy would make it possible to either reduce the number of samples required to achieve a certain level of error or to increase the amount of knowledge obtained using a fixed number of samples.

The techniques and algorithms used to define the design space locations of the experiments are generally called *sampling strategies*, and the related research field for engineering applications is usually termed *Design of Experiments*. The set of points defined by $\mathbf{X}_T$ is generally referred to as *training points*, *sample points* or simply *samples*, but different terminology is sometimes used in the literature. Some clarification about the term "experiment" in this context is required. In this document, the term "experiment" is used synonymously with "simulation", whereas in a large part of the literature, it directly refers to $\mathbf{X}_T$. This is the reason why the field of research connected with sampling strategies is sometimes indicated as Design of Experiment and not Design of Experiments ([52]).

The importance of the sampling phase has been emphasized by various studies [30, 60, 103] that have shown how the SM accuracy heavily depends on the sample point locations. Indeed, a poor sampling could have the same consequences of a poor choice of SM functional form: the resulting SM might be completely inadequate to approximate the "true" function f. Even though the selection of both SM functional form and sampling strategy is equally important for the accuracy of the final SM, there is a substantial difference: training data are available at the time of choosing the SM technique and they can be used to select the most appropriate SM, whereas no information is available at the time to decide the sampling strategy to use.

As described in chapter 2, several different sampling strategies have been proposed in literature, and they can be classified according to their architecture and behavior [30, 60]. The architecture can be either one-shot or sequential, and the behavior can be either space-filling or adaptive. Regarding the sampling architecture, a *one-shot* approach defines all the sample locations in a single step, whereas a *sequential* strategy sequentially populates $\mathbf{X}_T$ during the sampling process. Considering the sampling behavior, *space filling* approaches completely neglect response values $\mathbf{y}_T$ available in D_T and they focus only on evenly spreading the samples across the design space; conversely, *adaptive sampling* techniques use the response information $\mathbf{y}_T$ contained in D_T to concentrate the samples in regions of the domain that need refinement based on considerations of the local behavior of the function being approximated. As described in chapter 2 and widely shown in the literature [30, 36, 39, 47, 53, 61, 65], the adaptive-sequential is the most suitable class of sampling strategies when the number of simulation required to obtain an accurate SM has to be reduced as much as possible (e.g. in case a single experiment run is expensive in terms of time and/or money). Indeed, an adaptive-sequential technique not only uses all the information contained in D_T – both $\mathbf{X}_T$ and $\mathbf{y}_T$ –, but also gives the opportunity to monitor the process and to stop it when a certain level of accuracy is reached.

The importance of the sampling strategy in the overall SM creation process performance and the simultaneous increase of SM popularity led to the continuous development of new sampling strategies [60]. The main goal in this research area has been to develop sampling techniques that are applicable to a wider range of problems and that reduce as much as possible the number of choices that the end user has to make. Unfortunately, many of the sampling techniques that have been developed for this purpose have substantive shortcomings, and – as described in chapter 2 – some common characteristics of state-of-the-art adaptive-sampling techniques degrade their usability for model-based engineering applications [30]:

high computational complexity Some sampling techniques like integrated mean squared error (IMSE) [39, 98] or the cross-validation error based techniques [30, 47, 53] suffer from high computational complexity that makes them unscalable to high-dimensional applications.

SM dependency Some strategies – like cross-validation sampling and IMSE – use an SM to synthesize the information in D_T and to identify the region of the design space where the next sample should be located. Therefore an SM functional form has to be chosen at the beginning of the sampling process, meaning that such techniques are affected by the consequences of the No-Free-Lunch (NFL) theorem (section 1.4.2). In other words, the SM selected by the user at the beginning of the process with no available problem-specific prior information could be inappropriate for that problem, and therefore, it can negatively influence the performance of the sampling strategy and of the overall surrogate modeling process [30, 31].

high conceptual complexity Usually, a wider applicability is obtained by introducing advanced features and additional degrees of freedom in the process formulation, leading to sampling strategies that are highly conceptually complex. For an inexperienced user, these strategies are very challenging to implement and to use properly without a deep understanding of the mathematical formulation, and this is one of the reasons why advanced techniques are not widely considered for real-world applications. The gap between user knowledge and the knowledge required to properly use the strategies is so high that well-known and simpler strategies are preferable.

lack of robustness Some of the advanced sampling strategies suffer a lack of robustness. As with many sophisticated numerical formulations, the risk of numerical issues is high and can strongly affect the robustness of the overall process; examples include the sampling techniques that require the use of Kriging-like SM. Additionally, a sam-

pling strategy for engineering application must be able to handle situations when the solver used for the f evaluation (e.g. CFD) does not converge or returns an error.

It is apparent how these negative aspects of most of the state-of-the-art adaptive-sequential sampling techniques make their application very challenging in model-based engineering. Consequently, classical designs of experiment approaches are usually used in practice due to their ease of implementation and robustness even if they probably require a higher number of simulations to achieve a particular representation accuracy.

1.6 Motivations, Objectives, and Contributions

The scope of this section is to provide a high level discussion about the motivations and the fundamental objectives of this dissertation listed in chapter 3.

The previous section (section 1.5) identifies four main concerns about current adaptive-sequential sampling formulations: high computational complexity, model dependency, high conceptual complexity, and deficient algorithm robustness. Most of the current sampling techniques for engineering design [60] embody elegant mathematical formulations to efficiently use the collected response information with the goal of reducing the number of required samples, but usually they are limited by their assumptions and conceptual complexity, and, in the experience of the author, they are typically not used by practicing engineers.

Considering the introduction to sampling techniques provided in section 1.5 and the detailed literature review available in chapter 2, it is reasonable to expect that an effective and practical sampling strategy for surrogate modeling during the early phases of engineering design should:

1. be *adaptive* to use response information initially available or collected during the process to guide the sampling phase, thereby reducing the number of samples required to obtain a good representation of the response over the entire design space ([29, 30, 53, 96]);

2. be *sequential* to permit in-the-loop SM quality checks and the suspension of the sampling process whenever a certain accuracy level is reached ([29, 30, 53]);

3. be *flexible* to be applicable to a wide range of problems ([30]);

4. be *simple and robust* to be implementable by users with limited know-how in technical statistics and numerical methods ([4]);

5. have *low computationally* complexity and good scaling behavior with respect to the problem dimensionality to make the method usable in high-dimensional engineering design problems ([4]);

6. be *model independent* to reduce the impact of the particular SM choice on the surrogate modeling process efficiency. [30]).

The review of currently available sampling techniques presented in chapter 2 indicates that there is no present formulation able to achieve all of these characteristics. This gap in the capabilities of present methods is the main motivation behind this dissertation, which focuses on the development of a sampling strategy that attempts to address all six characteristics above. Additionally, literature lacked a thorough study about the influence of sampling strategy model dependency on the resulting representation accuracy. For this reason, a secondary objective of this dissertation is to complete a research which highlights the relevant impact that model dependency may have on the quality of the resulting sample distribution. The complete study, which also proposes a sampling architecture that mitigates the identified drawbacks of model dependence, is reported in chapter 4 and it has also been published in [31].

Liu et al. in [4, 60] identifies two additional characteristics that a sampling strategy should have:

- capability to work in either single or batch selection mode. Most current adaptive-sequential strategies have been developed to work only in single selection mode in

which a single sample is added at every iteration. On the other hand, a batch selection approach would give the opportunity to add more than one point per iteration, a property particularly important for an effective use of high performance computing (HPC) resources;

- capability to handle multiple responses. Indeed, it is common in engineering applications to deal with several responses at the same time; examples are the lift and the drag of a wing, or thrust and noise of an engine. In these cases, the sampling technique should be able to combine the information about the behavior of all the responses throughout the design space and to identify sample locations that are good in the sense of balanced effect in reducing the error of all of the responses.

Considering the complexity of state-of-the-art simulation codes and the development that they had during the last decade, the author believes that a sampling algorithm for engineering applications should also include two additional fundamental features:

- capability to use response derivatives. The development of adjoint methods ([32, 79]) led to simulation codes able to compute both the value and the gradient of the response with a limited impact in the computational time. A sampling strategy should embody the capability of using the gradient information to guide the sampling process.

- capability to handle critical errors in the simulation code. As previously described, the sampling algorithm usually samples the design space of an unknown response, and therefore it could happen that some designs are infeasible or their performance cannot be assessed using standard settings in the solver. In these cases, critical errors may occur in the solver, and consequentially no response y_T is returned to the sampling algorithm. Consider for example a CFD simulation of a rotor at different rotational speeds which may need distinct solver settings in presence of transonic or supersonic conditions. A robust sampling algorithm should be able to continue the

sampling process without being affected by solver critical errors, and also it should give to the user the opportunity of adjusting the simulation and refreshing the training dataset without stopping the sampling process.

The inclusion of these four additional characteristics in the proposed sampling technique is a secondary objective of this dissertation.

As described in chapter 2, all the adaptive-sequential strategies couple a refinement metric for the identification of design space regions in need of a sample refinement with an exploration metric to spread the points across the entire design space. A more detailed analysis of these algorithms (section 2.3.4) show two characteristics in common of all adaptive sequential techniques. First, all the formulations attempt to balance refinement and exploration by converting the multi-objective optimization problem (simultaneous maximization of exploration and refinement metrics) to a single-objective problem via penalty function or weighted sum approaches. Second, all the proposed model independent refinement metrics identify regions of the design space where more samples are required, without providing any indication about a direction along which next samples should be placed. The development of a technique able to achieve a refinement-exploration balance by keeping the multi-objective flavor of the problem, and to leverage directional information to guide the sampling process are other two main objectives of this dissertation.

1.7 Proposed Technique: Nearest Neighbor Adaptive Sampling

The proposed sampling strategy is developed with the objective of creating an effective and usable approach for engineering design problems. This objective is pursued by developing an adaptive-sequential sampling technique based on a local linear model, which is named Nearest Neighbors Adaptive Sampling (NNAS). Even though the details of the technique are discussed in chapter 5, it is important to understand how an adaptive-sequential sampling strategy based on a local linear model can potentially meet the requirements discussed above for a good sampling strategy:

- adaptive-sequential. Uses the available information about the response to guide the sampling process and makes it possible to perform in-the-loop operations such as SM accuracy evaluation;

- based on local model. This characteristic reduces the algorithm complexity and makes the NNAS model independent. Indeed, the adoption of a local model does not require the selection of a global SM functional form (as commonly required in most of the existing adaptive sampling strategies), and the effect of a sample addition is localized to its neighborhood, leading to a meaningful reduction in computational complexity (chapter 5);

- based on a linear model. The training process of a linear model is simple, fast, and robust; practicing engineers have sufficient knowledge about linear models and how to build them. Furthermore, linear model formulations can be easily adapted to use any type of available derivative information (e.g. from the adjoint method).

NNAS achieves the balance between the exploration and refinement by a stochastic selection criterion based on a Pareto-ranking procedure applied on the exploration-refinement metric domain. Additionally, the advanced NNAS algorithm formulation is able to handle solver critical errors and it includes batch selection, multi-response, and directional sampling capabilities.

Table 1.6 summarizes the solutions adopted to meet the sampling strategy requirements and indicates how these solutions are related to the characteristics of the early phases of the engineering design process.

1.8 Document Structure

The structure of this dissertation document is illustrated in figure 1.6.

In particular, chapter 2 provides a detailed literature review and classification of the currently available sampling strategies which is useful to highlight some of their key char-

Table 1.6: How NNAS can meet our sampling strategy requirements

Engineering design requirements	**Sampling strategy requirements**	**NNAS**
Use all the information available to define the sample locations and to reduce the total number of required simulation runs	Adaptive	Adaptive
Permit in-the-loop SM quality checks	Sequential	Sequential
Be implementable by users with a wide spectrum of knowledge	Simple and robust	Based on linear models
Usable in typical engineering design problems where the dimensionality is usually high	Low computational complexity with respect to problem dimensionality	Based on local models
Reduce the impact of the SM type chosen by the user	Model independent	Based on local models

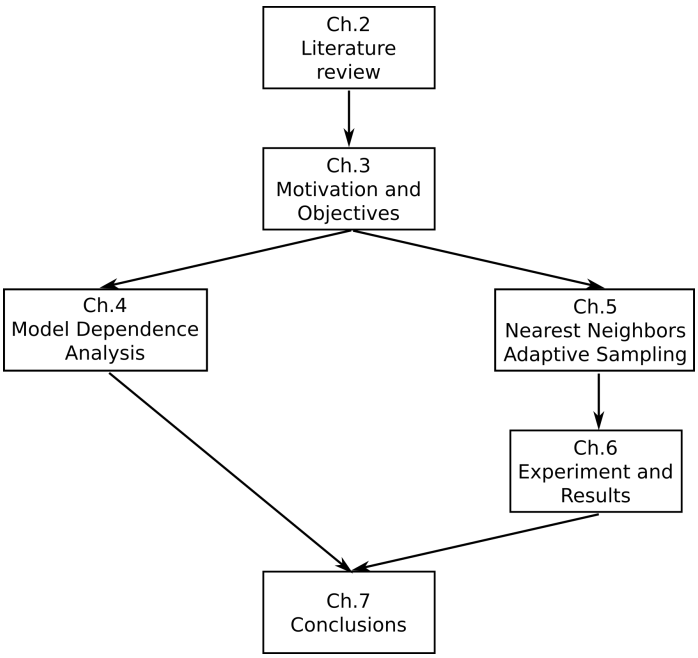


Figure 1.6: Document structure

acteristics that motivate the development of the proposed technique. Chapter 3 references to the outcome of the literature review in chapter 2 to formulate the objectives, motivations, and hypothesis of this dissertation. Chapter 4 includes an exhaustive analysis about the effect of model dependence on sampling efficiency, which is the first objective of this dissertation. Chapter 5 describes in details the proposed Nearest Neighbors Adaptive Sampling technique, including the formulations for batch-mode, multiple responses and infeasible region applications. Chapter 6 includes a description of the experiment plan and a complete set of results used to stress several aspects of the proposed sampling technique. This analysis is designed to substantiate the hypotheses introduced in chapter 3. Finally, chapter 7 summarizes the key outcome of these dissertation.

CHAPTER 2

SAMPLING STRATEGIES

This chapter presents a more detailed description of the *sampling* phase in the context of the surrogate modeling process that is fundamental to properly understand the motivations behind the proposed technique and the reasons for particular choices involving its formulation. After a brief description of the importance of the sampling phase in the surrogate modeling process, the chapter continues with an exhaustive survey of existing sampling strategies, how they are classified, and the analysis of their positive and negative aspects with respect to the overall process efficiency.

At this point, it is important to remember that this dissertation is focused on the process of creating global SMs for computer experiment applications. The fact that only computer experiments are considered removes the need for characteristics in sampling procedures necessary to evaluate aleatory uncertainties – such as experiment repetitions – that usually are necessary to estimate the output value y obtained by real experiments. Computer simulations are typically deterministic, which means that the unknown function $f(\mathbf{x})$ has a one-to-one correspondence between the input vector $\mathbf{x}$ and the output y, and aleatory uncertainties are not present [60, 87, 103].

Several studies [18, 20, 36, 39, 60, 102] show how the sampling phase is crucial for the creation of an accurate SM, because it is during this phase that information about the phenomenon modeled by the computer simulation is gathered and collected in the training dataset $D_T = \{\mathbf{X}_T, \mathbf{y}_y\}$. The objective of the sampling algorithm is to populate the matrix $\mathbf{X}_T$ that defines the design space locations (samples) where the response f will be evaluated by the computer simulation. This process is schematically represented in figure 2.1.

In principle, an infinite number of samples, uniformly distributed across the domain, would make it possible to create an SM that perfectly matches the unknown f (in the case

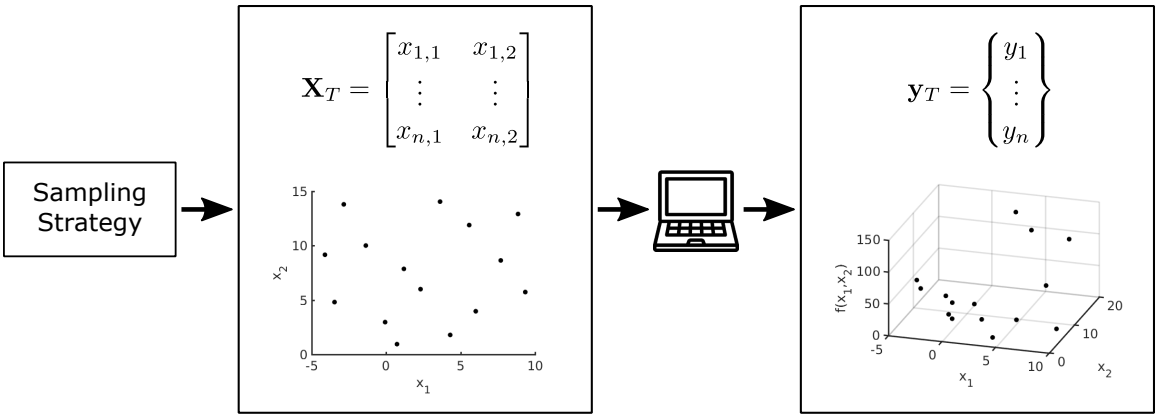


Figure 2.1: Schematic representation of a sampling strategy

of an interpolating SMs), but unfortunately the allowable number of samples is bounded in practice by the limited amount of time available to build the SM and by the simulation runtime t_{ev} (Equation (1.1)). Therefore, the goal of a sampling technique is to efficiently gather as much information as possible about f by limiting the total process time t_{tot} (or the total process cost $\$_{\text{tot}}$). As described in section 1.3 and studied by Garbo in [30], time efficiency depends on a trade-off between the total number of samples n_{tot} and the ratio between the additional computational time and the evaluation time (Equation (1.7)).

The question at this point is: what does "efficiently gather information" mean? The example illustrated in figure 2.2 should help to qualitatively answer this question. Figures 2.2a and 2.2b report two different sets of sample locations, while figures 2.2c and 2.2d represent the contour and surface plots of the real function f that has to be modeled. Imagine that you have to select one of the two sample distributions shown in figures 2.2a and 2.2b to define the set of experiments used to collect information about the unknown function f, and assume that you do not have access to figures 2.2c and 2.2d. Most people would likely select the option in figure 2.2a because the samples are evenly spread over the entire design space. This logical choice would be based on the consideration that, for a given number of samples, obtaining an even distribution of samples across the domain increases the probability of capturing all the interesting characteristics of f. Now, imagine repeating the decision considering also the information contained in figures 2.2c and 2.2d: do you

still choose the sample locations of figure 2.2a? In this case, most of people would likely select the sample distribution of figure 2.2b. The information about f contained in the two lower figures leads one to choose the second set of samples because their distribution is more suitable to model the region of the design space where f has a high variability. It is also clear that the first set of samples is not appropriate to efficiently gather information about f, and it would probably cause a severe misrepresentation of the highly nonlinear portion of the response. Therefore, "efficiently gather information," in the context of adaptive sampling, must be understood to mean to distribute the samples such that the entire design space is explored, and the regions where the response has high variability are accurately described. The test function used for this example is not fictitious and expressly created for this exercise, but rather is the input noise current of a low noise amplifier [20, 38] (section D.7).

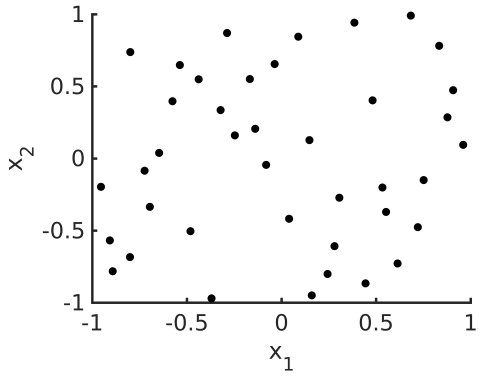
This simple example helps to introduce the two desired behaviors that should guide the sampling process: exploration and refinement (section 2.1.1).

2.1 Exploration vs Refinement: Space Filling vs Adaptive Sampling Strategies

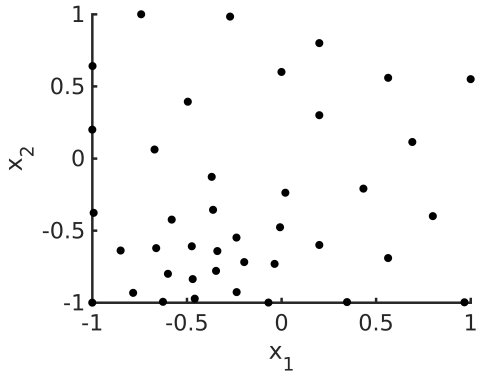
2.1.1 Exploration vs Refinement

As introduced in the previous example, a sampling strategy should both explore the design space to detect as many response features as possible, and refine the sample distribution in portions of the design space where particular features are detected. These two behaviors are commonly termed *exploration* and *refinement* (or exploitation) in sampling and optimization research areas [36, 49, 60].

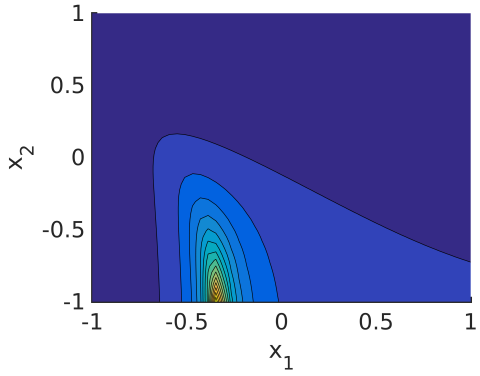
In the optimization field, a good algorithm has to balance exploration and exploitation actions to increase the probability of finding the global optimum: exploitation refines the search around possible optimal regions identified by the exploration of the design space. The algorithm cannot rely only on exploitation because it will probably converge to the first detected local optimum, and therefore exploration is required to identify other possible op-



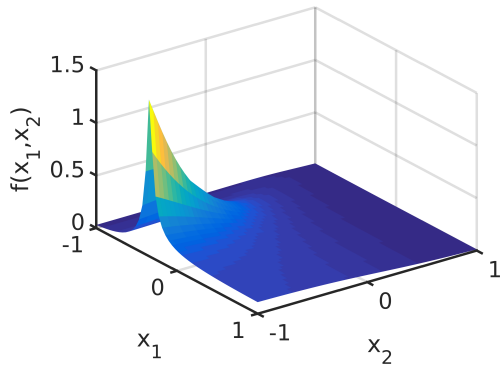
(a) Samples generated using a one-shot space-filling algorithm



(b) Samples generated using a sequential adaptive algorithm



(c) Response contour



(d) Response surface

Figure 2.2: Importance of choosing the correct sampling strategy

timal regions, thereby increasing the probability of converging on a global optimum. Several optimization algorithms try to embody these two search characteristics, and a famous example is the EGO algorithm proposed by Jones [49] based on the expected improvement formulation.

A similar behavior is desired in the sampling strategies where discrete information about f has to be collected to properly model the response by an SM. This point-wise information must be spread over the entire design space to identify as many features as possible (*exploration*), but also samples have to be concentrated where the response shows a high variability (*refinement*). These two opposite behaviors must coexist in the same strategy to efficiently sample the design space, and therefore a logic to make the two behaviors cooperate is required. Ideally, a sampling strategy should determine the location of the samples ($\mathbf{X}_T$) by simultaneously maximizing both exploration and refinement metric leading to a two-objective optimization problem:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{Maximize}} \quad [R(\mathbf{x}, \boldsymbol{\iota}), E(\mathbf{x})] \\ &\text{subject to} \quad \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \end{aligned} \tag{2.1}$$

where the vector $\boldsymbol{\iota}$ contains the available knowledge about the response f used in the evaluation of the refinement metric R, the exploration metric E is usually based on euclidean distance, and $\mathbf{x}_L$ and $\mathbf{x}_U$ are the lower and upper bounds of the design space, respectively. How the optimization problem in (2.1) is solved is a characteristic of each sampling strategy. For example, some techniques evaluate the refinement metric at candidate samples generated by a preliminary exploration phase [53], others perform the exploration of a design space region previously identified by the maximization of the refinement metric [20], yet others combine the two behaviors in a single sampling metric $S(\mathbf{x}) = S(E(\mathbf{x}, \boldsymbol{\iota}), R(\mathbf{x}, \boldsymbol{\iota}))$ [16]. As described in section 5.1, the technique proposed in this dissertation achieves the exploration-refinement balance by a more rigorous Pareto-ranking approach due to the

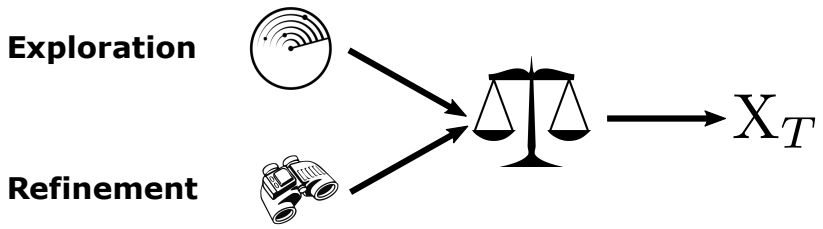


Figure 2.3: Schematic representation of a sampling metric

multi-objective characteristic of the problem in (2.1). A schematic representation of the metrics involved in the sampling process is illustrated in figure 2.3.

2.1.2 Space-filling vs Adaptive

The absence or the presence of refinement in the sampling behavior determines the first sampling strategy classification: space filling or adaptive sampling techniques, respectively.

Space-filling Space-filling sampling strategies have as a unique goal to evenly spread the samples across the design space without taking any refinement action; examples of these techniques are the factorial and Latin hypercube designs [4, 48, 98]. The additional computational time needed by the sampling phase t_{sampl} is almost negligible for this class of techniques since the algorithms are fast, robust, and well developed. Because of these properties, space-filling designs are usually adopted at the beginning of the surrogate modeling process when no information about f is available, and some samples are required to ignite more advanced techniques like sequential adaptive strategies. Other common applications are problems where the function evaluation time t_{ev} is very low (e.g. in wind tunnel experiments [22, 74, 76]), and a low t_{sampl} is desired to mitigate the impact of additional computational time $\bar{t}_{\mathrm{add}}$ on the total process time t_{tot} (equation (1.2)).

Adaptive Adaptive techniques synthesize the available knowledge $\boldsymbol{\iota}$ into a refinement metric to detect the design space regions where more samples are required ("regions of interest") to enhance the response representation. The specific formulation of the

metric R, and the type of required information $\boldsymbol{\iota}$ are unique characteristics of each adaptive sampling strategy, as it is illustrated in the brief description of state-of-the-art techniques in section 2.3. Unfortunately, relying only on R to guide the sampling is not possible, since it may lead to a sample clustering around the first detected region of interest. To avoid this side effect, adaptive techniques always combine R with an exploration metric E, and the approach used to couple these two metrics is specific to each particular sampling strategy formulation.

2.1.3 Model Dependency of Adaptive Sampling Strategies

Particular attention must be paid to a specific trait of some adaptive sampling techniques: **model dependency**. As previously stated, adaptive sampling strategies use the information vector $\boldsymbol{\iota}$ to evaluate the refinement metric R that helps in the identification of design space regions where additional samples are needed to enhance the response representation. A subset of adaptive formulations uses a global SM to synthesize the information $\boldsymbol{\iota}$, thereby making the resulting sampling behavior dependent on the chosen SM formulation. Additionally, the challenge of selecting a priori a suitable SM functional form to sample and model the unknown response (No-Free-Lunch (NFL) section 1.4.2) may lead to an unexpected and poor design space sampling in case of an inappropriate choice. As studied in [30, 31] and reported in chapter 4, *model dependence* consequences can be mitigated by including an active SM selection process within the sampling algorithm, however causing an increase in computational complexity as side effect. Some examples of model dependent adaptive strategies (described later in section 2.3) are the Kriging-based strategies (such as maximum variance and IMSE) that depend on the chosen covariance function, or the CV-based strategies that are influenced by the functional form used in the cross-validation process.

As shown by Garbo and German [30, 31], the selection of the most appropriate SM is crucial to efficiently sample the design space with a model dependent sampling strategy.

The expected reduction in n_{tot} due to the usage of an adaptive sampling procedure may completely vanish if an inappropriate SM functional form is adopted. In addition, the required SM selection process may lead to a considerable increase in computational time – particularly if SMs with nonlinear training procedures are involved – and consequentially on the total SM creation process time t_{tot} (Equation (1.1))[30, 31]. These two side effects of model dependent adaptive sampling strategies are the main motivations behind the objective of developing a model independent adaptive sampling technique in this work. Such a strategy will remove the risks connected to the selection of an inappropriate SM or the need for performing an active SM functional form selection, with direct benefits to the overall surrogate modeling process efficiency.

More details about model dependence in sampling strategies are available in chapter 4 which is entirely devoted to present a study that underlines the need of coupling model dependent sampling strategies with active SM selection.

2.2 Sampling Strategy Architectures: One-shot vs Sequential

The algorithm architecture is the characteristic considered in another classification of sampling strategies [30, 39, 60]: *one-shot* (open-loop) vs. *sequential* (closed-loop). As is illustrated in figures 2.4 and 2.5, the terms open-loop and closed-loop are respectively related to the absence or the existence of a feedback connection in the algorithm implementation scheme.

one-shot (open-loop) A one-shot sampling strategy creates the entire set of sample locations $\mathbf{X}_T$ in a single algorithm run before the beginning of the evaluation phase. Once $\mathbf{X}_T$ is generated, there is no remaining sampling budget to refine particular design space regions that a post processing phase may reveal as inappropriately sampled. Additionally, all the sample locations in $\mathbf{X}_T$ must be evaluated before performing post processing actions such as checking SM quality, because most one-shot algo-

rithms do not follow any particular order when they populate $\mathbf{X}_T$, and therefore a partial $\mathbf{X}_T$ evaluation may result in an uneven design space sampling.

The *a priori* availability of the fully populated $\mathbf{X}_T$ makes the evaluation process embarrassingly parallelizable; nowadays, this property is important due to the wide availability of HPC resources. Examples of one-shot strategies are the fractional factorial designs [4, 98], Latin Hypercube [4, 49], and the one-shot versions of the adaptive techniques based on Kriging SM [29, 39, 97, 98].

sequential (closed-loop) A sequential sampling strategy sequentially adds batches of sample locations to $\mathbf{X}_T$, and it waits until the end of the last sample batch evaluation before proceeding with the generation of the next batch. It is important to clarify that "sequential" is not meant to label the way in which the matrix $\mathbf{X}_T$ is generated, but instead, it describes the alternating execution of the sampling and the evaluation phases. If a sampling algorithm sequentially creates the full $\mathbf{X}_T$ before the evaluation phase, it is not intended as a sequential strategy in our terminology. Indeed, several one-shot techniques use this approach to populate $\mathbf{X}_T$ because the solution of the optimization problem required to simultaneously identify all n_{tot} sample locations is computationally prohibitive, especially when n_{tot} is high or when the sampling metric evaluation is computationally expensive.

A sequential technique consumes the sampling budget as the process proceeds and does not expend the entire budget at the beginning as in a one-shot strategy. This characteristic of the algorithm architecture allows to analyze the available data incrementally during the process, and to guide the subsequent sampling behavior using the information already collected if the strategy is adaptive. For example, if the available training dataset D_T reveals a highly nonlinear behavior of the response f a specific region of the design space, part of the remaining sampling budget can be used to perform a better investigation of that region.

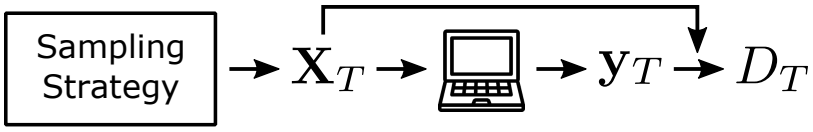


Figure 2.4: One-shot sampling architecture scheme

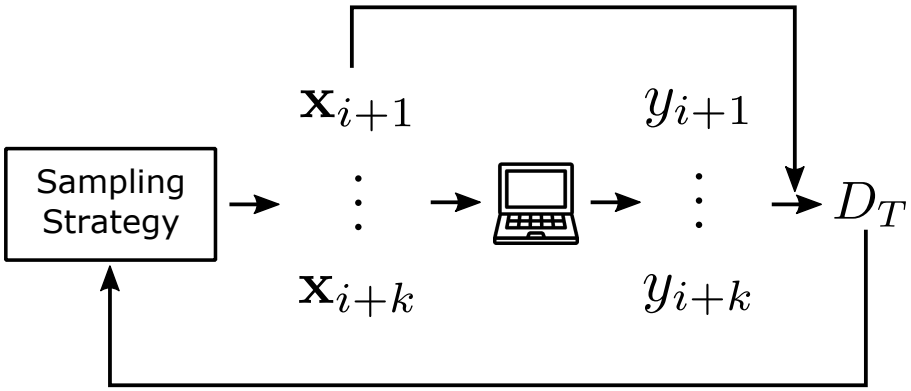


Figure 2.5: Sequential sampling architecture scheme ($i = k\,(n_{\text{it}} - 1)$)

A schematic representation of these two classes of sampling strategies is reported in figure 2.4 and 2.5, and examples of several state-of-the-art sampling techniques are provided in section 2.3.

2.3 Sampling Technique Configurations: Analysis and Survey of Existing Techniques

Sections 2.1 and 2.2 introduced two different sampling strategy classification paradigms: behavior (space-filling or adaptive), and architecture (one-shot or sequential). The two classifications are not mutually exclusive, and therefore all the four behavior-architecture combinations are possible. The objective of this section is to describe and characterize these four configurations to identify their positive and negative aspects.

All the behavior-architecture configurations are schematically represented in table 2.1 where the positive and negative characteristics – described later in this section – are listed for each combination.

Table 2.1: Summary of positive and negative characteristics of the four possible sampling configurations

Behavior	**Architecture**: One-shot	**Architecture**: Sequential
Space-filling	👍 Easy to use 👍 Fast and robust 👍 Wide software availability 👍 Easy to use with HPC resources 👎 No refinement 👎 All function evaluations must be run before proceeding with the SM creation process	👍 Easy to use 👍 Fast and robust 👍 Implementation from one-shot space-filling technique 👍 Opportunity to run quality check during the surrogate modeling process since it is not required that all the function evaluations be completed to have an evenly sampled distribution 👎 No refinement even if an SM has been created to check the quality of the process
Adaptive	👍 Refinement is achieved using function information available before the beginning of the sampling process ($\boldsymbol{\iota}_0$) 👍 Easy to use with HPC resources 👎 Some computational complexity is introduced to include $\boldsymbol{\iota}_0$ in the algorithm 👎 All function evaluations must be completed before proceeding with the SM creation process	👍 Refinement is achieved using all the available collected data 👍 Opportunity to run quality check during the surrogate modeling process since it is not required that all the function evaluations be completed to have an evenly sampled distribution 👎 Difficulty to use it with HPC resources 👎 High computational complexity 👎 Some formulations are model dependent

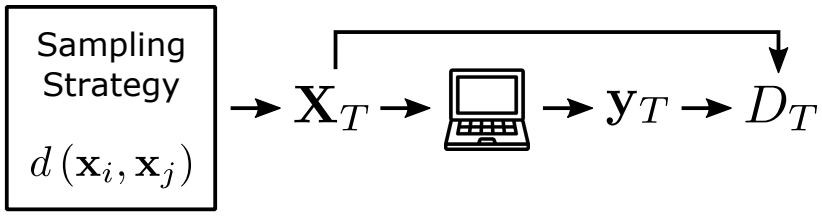


Figure 2.6: One-shot space-filling sampling configuration scheme

2.3.1 One-shot Space-filling Sampling

A one-shot space filling sampling technique defines all the sample locations $\mathbf{X}_T$ in a single stage using only their reciprocal distance in the design space $d(\mathbf{x}_i, \mathbf{x}_j)$ as a sampling metric; therefore, no refinement is involved in the process, and the sampling metric can be written as $S(\mathbf{x}) = E(\mathbf{x})$. Once $\mathbf{X}_T$ is populated with the n_{tot} samples, the samples are evaluated to obtain the response vector $\mathbf{y}_T$ required to create the training dataset D_T, as illustrated in figure 2.6.

This configuration inherits from the one-shot architecture the lack of suitability for performing a SM quality check during the evaluation phase because the order of samples in $\mathbf{X}_T$ is usually random and therefore the already-obtained samples are not assured to be evenly distributed across the design space (In other words, a large number of randomly sampled points is needed before we can have confidence of an even distribution of points in an $n-$dimensional design space. By stopping short with only a few points, one should not expect an even distribution).

One-shot space-filling sampling techniques are the easiest to use and to implement. The algorithms have been throughly studied, and they are robust and efficient [4, 68, 98], requiring a negligible additional computational time during the sampling phase (t_{sampl}). On the other hand, the fact that no refinement metric is considered during the sampling process makes this approach inefficient in terms of n_{tot}. Indeed, the only goal of a one-shot space-filling technique is to evenly spread all the samples across the design space, with the assumption that this will lead to a good sampling of the response. Unfortunately, this as-

Table 2.2: List of some existing one-shot space-filling sampling techniques

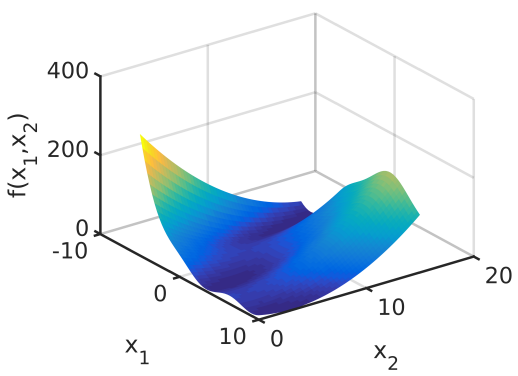
Name	**References**
Full Factorial	[4]
Fractional Factorial	[4, 77, 98]
Latin Hypercube	[4, 49, 77]

sumption is not always correct, in particular when the response function f is non stationary and anisotropic, as it is in many practical applications. For example, a highly nonlinear region may be concentrated in a limited portion of the overall design space, leading to an inefficient use of the sampling budget.

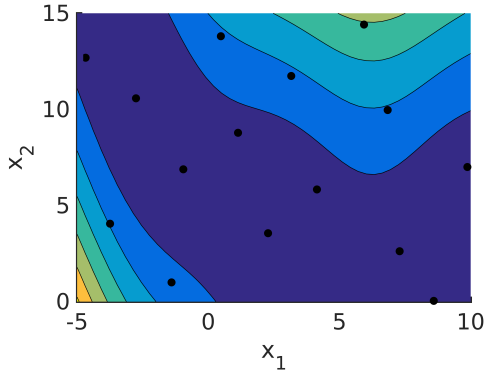
Consider the examples illustrated in figure 2.7, where the same $\mathbf{X}_T$ of 15 points is used to sample two different functions. In the first case (figures 2.7a and 2.7b), the function has a qualitatively isotropic variability across the entire design space, and the samples are able to capture the overall trend of the function; some misrepresentation errors may appear on the lower left corner of the design space where a steep function change is present. In the second case (figures 2.7c and 2.7d), the sample distribution completely misses the important peak in the function in the lower middle part of the design space; most of the samples are located on an almost flat region, and only few are used to sample the zone with high variability in the function. This is caused by the fact that a one-shot space-filling design has no access to any piece of information about f, either available at the beginning of the SM creation process or collected during the evaluation phase.

The fact that the entire $\mathbf{X}_T$ is completely populated before the evaluation phase makes this sampling approach attractive for applications where HPC resources are available. These parallel computing architectures gives the opportunity to evaluate several samples at the same time and consequentially to reduce the time required during the evaluation phase.

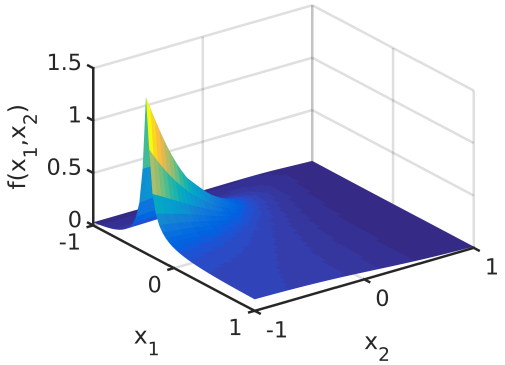
Examples of existing one-shot space-filling techniques with relative references are listed in table 2.2.



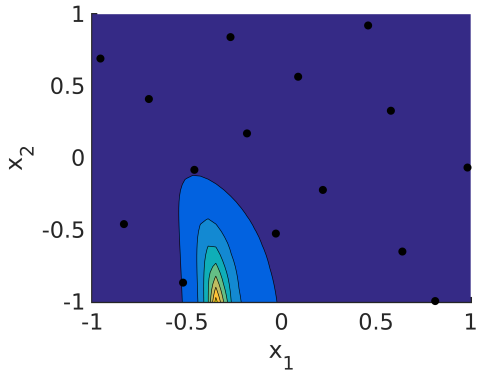
(a) Branin function surface



(b) Branin contour plot with samples



(c) LNA function surface



(d) LNA contour plot with samples

Figure 2.7: Examples of samples resulting from a one-shot space-filling method for two different test functions

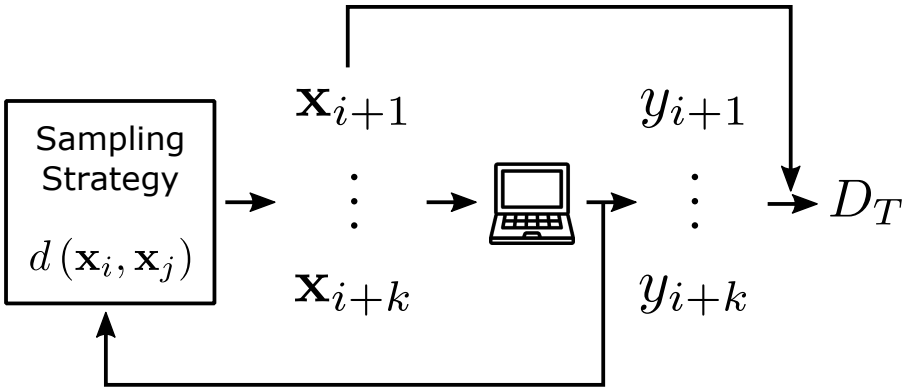


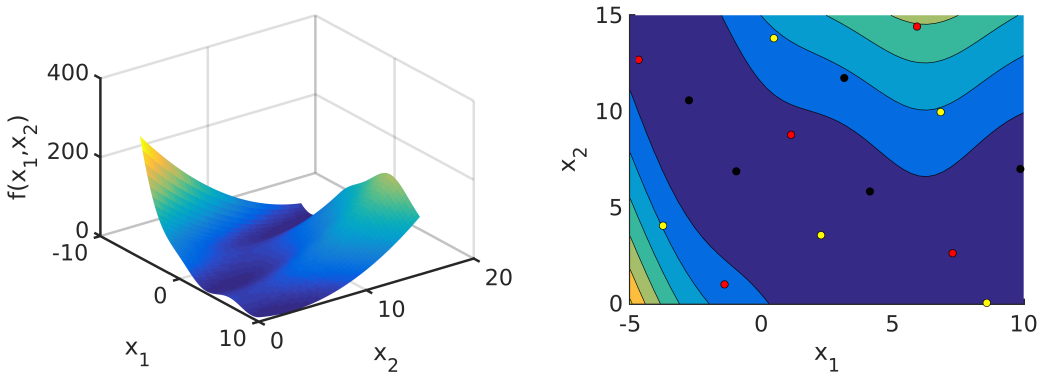
Figure 2.8: Sequential space-filling sampling configuration scheme adding k samples at iteration

2.3.2 Sequential Space-filling Sampling

A sequential space-filling sampling strategy sequentially adds sets of samples to the matrix $\mathbf{X}_T$ using only their reciprocal distance in the design space $d(\mathbf{x}_i, \mathbf{x}_j)$ as the sampling metric. Similarly to one-shot space-filling techniques, no refinement is involved in the process, and the sampling metric can be written as $S(\mathbf{x}) = E(\mathbf{x})$.

The sequential architecture implies that a new set of samples is added to $\mathbf{X}_T$ only when the evaluation of the previously generated one is completed, as illustrated in figure 2.8. A direct consequence of this "sampling-evaluation schedule" is the opportunity to run quality checks during the sampling phase. For example, it is possible to create the SM, compute a GEE, and add other samples if the quality requirement is not satisfied. On the other hand, no information about the function f can be returned to the sampling algorithm in a space-filling technique, even though it is available. The sampling approaches which allow information feedback belong to the sequential adaptive category that is described in section 2.3.4.

The implication of the absence of refinement previously described for one-shot space-filling techniques is valid also for the sequential ones: the sampling algorithm is not able to refine the sample distribution using function information, and that may cause a misrep-



(a) Branin function surface

(b) Branin contour plot with samples

Figure 2.9: Examples of samples resulting from a sequential space-filling method

resentation of the function f (figure 2.7). The inability of using the collected data to guide the sampling process makes the use of sequential space-filling techniques very rare.

Similarly to the one-shot space-filling techniques, the sequential ones can be efficiently used when HPC resources are available if the number of samples generated at every iteration is proportional to the number of simulations that can be run in parallel. Usually, the sequential space-filling algorithms are simply the one-shot versions used in a sequential fashion (table 2.2).

An example of a sequential space-filling technique is reported in figure 2.9, where three sets of five samples (different colors) are sequentially added to $\mathbf{X}_T$.

2.3.3 One-shot Adaptive Sampling

A one-shot adaptive sampling technique defines all the sample locations in a single stage using both an exploration metric E (usually based on reciprocal distance $d(\mathbf{x}_i, \mathbf{x}_j)$) and a refinement metric R computed from the function information vector $\boldsymbol{\iota}$ as prescribed by the specific formulation. The sampling metric can therefore be written as $S(\mathbf{x}) = S\left(E(\mathbf{x}), R(\mathbf{x}, \boldsymbol{\iota})\right)$.

Since the entire $\mathbf{X}_T$ is created in a single stage (one-shot) using only the preliminary information $\boldsymbol{\iota}_0$ available at the beginning of the process (adaptive), the refinement accuracy

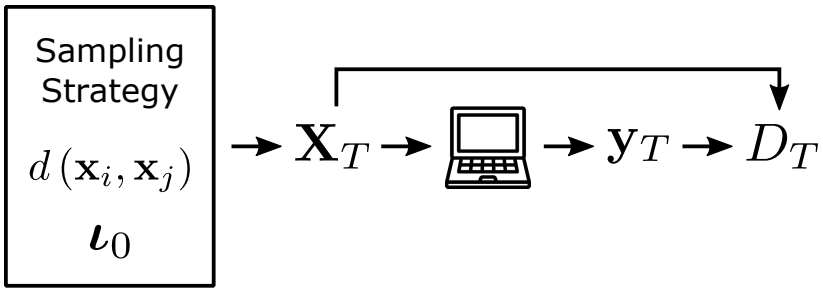
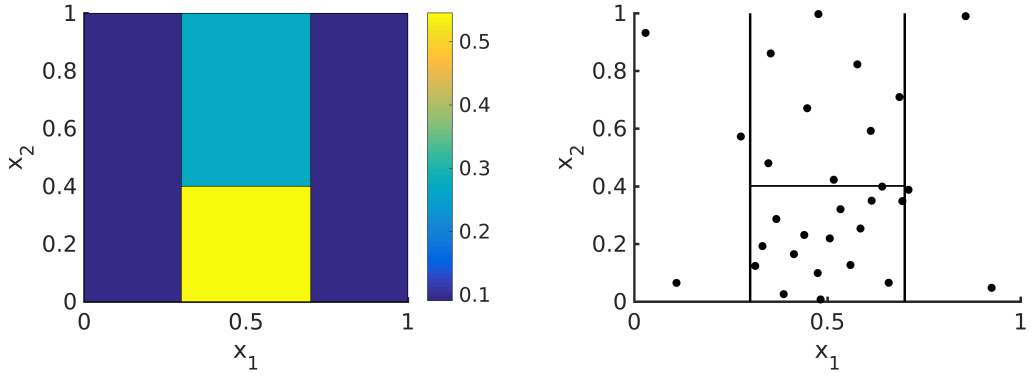


Figure 2.10: One-shot adaptive sampling configuration scheme

strongly depends on the quality of $\boldsymbol{\iota}_0$. If $\boldsymbol{\iota}_0$ were incorrect, no refinement can be performed after the end of the function evaluations because no remaining sampling budget is available (one-shot architecture). Therefore, the resulting sample locations can be completely inaccurate in representing the response f. As previously described (section 2.1.2), an adaptive technique also needs an exploration metric E to avoid excessive sample clustering in particular regions of the design space. The one-shot adaptive sampling strategy configuration is illustrated in figure 2.10.

The use of information $\boldsymbol{\iota}$ in the sampling process usually implies a more advanced algorithm formulation with a resulting increase in the computational complexity and sampling time t_{sampl}. On the other hand, if $\boldsymbol{\iota}_0$ is accurate, the adaptive behavior of the algorithm usually reduces the number of samples required to achieve a given quality in the final SM [39, 97]. As already discussed, the one-shot architecture does not allow intermediate quality checks because all the samples in $\mathbf{X}_T$ must be evaluated before any analysis can be performed on D_T.

The type of information required by one-shot adaptive techniques depends on the specific strategy formulation. The two most commonly used techniques are the covariance-based and the region-of-interest-based ones. In a *covariance-based* technique, $\boldsymbol{\iota}_0$ is used to fully define the covariance function $K(\mathbf{x}_i, \mathbf{x}_j, \mathbf{h})$ – where $\mathbf{h}$ is the vector of covariance coefficients – required by the algorithm to evaluate the refinement metric R. Covariance-based one-shot adaptive techniques are "model dependent" since the resulting sample locations strongly depend on the selected covariance "model" K and the estimated $\mathbf{h}$.



(a) Region of interest values

(b) Resulting samples distribution

Figure 2.11: Example of region-of-interest-based one-shot adaptive technique

Table 2.3: List of some existing one-shot adaptive sampling techniques

Name	**References**
Covariance-based	[39, 75, 97, 98]
Region-of-interest-based	[4, 98]

In a *region-of-interest-based* technique, the preliminary information $\boldsymbol{\iota}_0$ is used to subdivide the design space into regions with different "interest levels" that will guide the sampling algorithm to unevenly distribute the samples across the domain. Usually, these algorithms are derived from the one-shot space-filling algorithms: a one-shot space-filling technique is used to generate in each design space subregion a number of samples proportional to its "interest level". This approach inherits the simplicity and robustness of the space-filling approach, but it is extremely dependent on the quality of information $\boldsymbol{\iota}_0$ used to determine the regions of interest. An example of region-of-interest-based sampling technique is illustrated in figure 2.11

Similarly to the one-shot space-filling strategies, these techniques generate $\mathbf{X}_T$ in a single stage, and therefore they are suitable for applications where HPC resources are available.

A list of existing one-shot adaptive techniques is reported in table 2.3

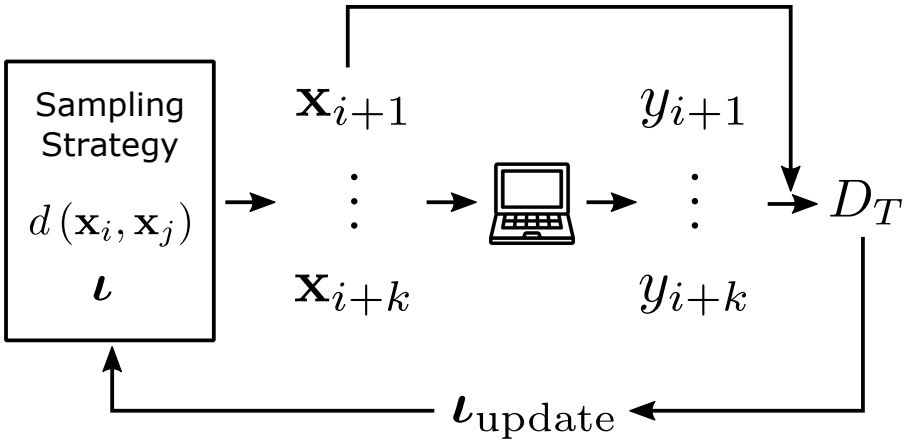


Figure 2.12: Sequential adaptive sampling configuration scheme adding k samples at iteration

2.3.4 Sequential Adaptive Sampling

A sequential adaptive sampling technique sequentially adds sample locations to the matrix $\mathbf{X}_T$ by using both an exploration metric E and a refinement metric R computed from the response information ($\boldsymbol{\iota}$). Therefore, the sampling metric can be written as $S(\mathbf{x}) = S\left(E(\mathbf{x}), R(\mathbf{x}, \boldsymbol{\iota})\right)$ in which the approach followed to combine E and R depends on the specific formulation.

Thanks to the adaptive behavior, the information vector $\boldsymbol{\iota}$ is used identify the regions of the design space where a sample refinement is needed, but in contrast to the one-shot techniques, the sequential architecture permits the update of $\boldsymbol{\iota}$ during the sampling process, softening the dependence of the resulting sample distribution on the initial information vector $\boldsymbol{\iota}_0$. Similarly to other adaptive techniques, an exploration metric E is usually included in the sampling metric to avoid sample clustering and to enforce a certain degree of exploration. A schematic representation of a sequential adaptive sampling strategy is illustrated in figure 2.12.

Since $\boldsymbol{\iota}$ is updated every time new information about f is collected, the overall computational complexity of this class of sampling strategies is the highest of all four possible sampling approaches. Obviously, the degree of complexity mainly depends on how the

specific algorithm formulation implements the collection, update, and use of $\boldsymbol{\iota}$. The additional computational complexity directly affects the sampling time t_{sampl}, but the efficient use of information about f makes it possible to drastically reduce the number of required samples compared to the other classes of sampling techniques [30, 47, 53, 61]. A preliminary study of the interaction between n_{tot} and t_{sampl} for a particular sequential adaptive sampling strategy is presented by Garbo an German in [30]. In this article, the authors show that sequential adaptive techniques are generally worthwhile when the evaluation time t_{ev} is high enough to mitigate the impact of the additional sampling time t_{sampl} on the total process time t_{tot} (Equation (1.4)).

Likewise the sequential space-filling strategy, this sampling configuration permits quality check analyses during the process. The matrix $\mathbf{X}_T$ is sequentially populated by the addition of new samples only when the evaluation of the old ones is completed, thereby making possible to stop the sampling process once a certain level of representation quality is reached, even if the sampling budget n_{tot} is not completely used.

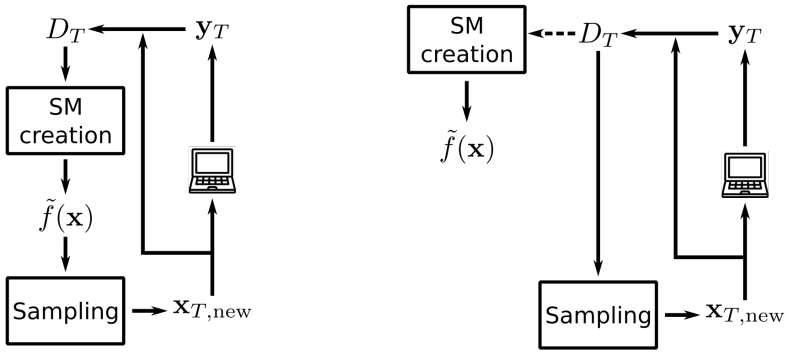
Unfortunately, most of the available sequential adaptive sampling techniques work in "single selection mode," making their use inefficient when HPC resources are available. This characteristic is mainly due to the computational complexity of the adopted sampling metric S that causes the solution of the sampling optimization problem to be practically feasible only for the single selection mode.

Since the technique proposed in this dissertation is a sequential adaptive sampling strategy, the chapter concludes with a more detailed description of three representative classes of state-of-the-art techniques: covariance, cross validation, and Voronoi based strategies. Before that, an analysis of two categories of sequential adaptive technique is proposed, namely the model dependent sequential adaptive strategies (MDSASs) and model independent sequential adaptive strategies (MISASs). As can be foreseen from the names, the model dependence/independence is the characteristic that determines the classification

within the two groups. This analysis highlights the benefits of MISAS which is the class of the sampling technique proposed in this dissertation.

Model Dependent vs Model Independent Sequential Adaptive Strategies As previously described, a sequential adaptive strategy populates the training dataset D_T by sequentially adding batches of samples in design space regions where a refinement is needed to increase the response representation accuracy (*sequential*). The identification of these regions is usually supervised by a refinement metric which is synthesized from the most updated D_T (*adaptive*). In the majority of the sequential adaptive strategies, this synthesis process is conducted by means of a global SM, leading to a sampling behavior that is dependent on the chosen SM functional form. Such techniques are usually called model dependent sequential adaptive strategies (MDSASs) [60](figure 2.13a). The implication of model dependence of MDSASs is investigated by Garbo and German in [30, 31] and described in chapter 4, where the authors underline the importance of carefully selecting the SM functional form when such techniques are used. The results clearly indicate how the reduction in number of samples expected by the use of an MDSAS may vanish if an inappropriate SM formulation is chosen to supervise the sampling process. This MDSAS side effect can be mitigated by adopting a sampling architecture with active SM selection [31] which reduces the risk of selecting an inaccurate supervising SM at the price of an increased computational complexity (in particular when complex and advanced SMs are considered). Examples of MDSASs are the Kriging-based integrated mean squared error (IMSE) and maximum entropy (ME) methods ([39, 98]), or the strategies based on local error estimators ([30, 31, 47, 53, 61, 111]).

The need to reduce the dependency of the sampling behavior on the selected SM led to the development of model independent sequential adaptive strategies (MISASs) [60]. By introducing refinement metrics that are not computed from a global SM, MISASs decouple the sampling process from SM training (figure 2.13b), thereby making it possible to



(a) MDSAS (b) MISAS

Figure 2.13: Adaptive sampling architectures

test all the desired SM classes without affecting the sampling process, which continuously and independently updates D_T. This flexibility is not available in applications of MDSASs in which an updated global SM is required at the beginning of each sampling iteration to compute the model dependent refinement metric used by the algorithm (figure 2.13a). Examples of MISASs are the Voronoi-based [18, 20, 44] and Lipschitz-based [64] strategies, and the technique presented in this dissertation. Some techniques such CV-Voronoi ([111]), that are occasionally referred to as model independent [45], are considered as model dependent in this context because the evaluation of CV metrics always requires the specification of an SM.

Covariance Based Sequential Adaptive Sampling This class of sequential adaptive sampling techniques includes all the strategies that use a covariance function $K(\mathbf{x}_i, \mathbf{x}_j, \mathbf{h})$ to compute the sampling metric. These strategies are usually closely related to Kriging SM where the covariance function is the principal "parameter" of the SM functional form $\tilde{f}$ (however, Kriging is usually described as "non-parametric"). These strategies share the same core formulation with the covariance-based one-shot adaptive techniques, with the difference being that the sequential architecture implies a continuous update of the covariance parameter vector $\mathbf{h}$ every time a new entry is added to the training dataset D_T. The

Table 2.4: List of some existing covariance-based sequential adaptive techniques

Name	References
Maximum variance	[29, 75, 98]
Integrated Mean Squared Error	[29, 39, 98]
Mutual Information	[98]
Interquartile Range	[115]

adopted algorithms to estimate $\mathbf{h}$ from $\boldsymbol{\iota}$ are usually computationally expensive due to their nonlinear formulation, and sometimes they suffer robustness issues [89].

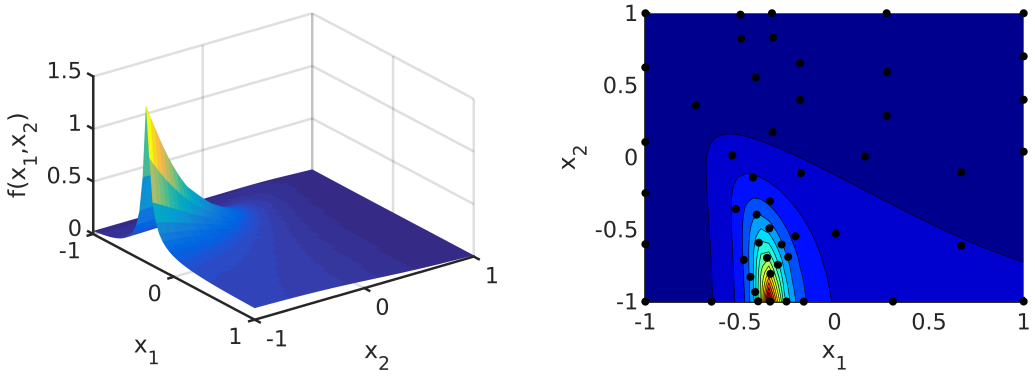
The dependency on the covariance function choice is another drawback of this class of sampling techniques, making them "covariance model dependent" and therefore MDSASs. Some authors suggest to actively select the covariance function within a set of possible alternatives using the available training dataset D_T [24], but to the best knowledge of the author, the active covariance function selection has never been applied to sequential adaptive sampling applications due to the extremely high computational cost required by the process.

A list of covariance-based sequential adaptive techniques with relevant references is reported in table 2.4.

Cross Validation Based Sequential Adaptive Sampling This class includes all the adaptive sampling strategies that synthesize the information vector $\boldsymbol{\iota}$ into a cross validation (CV) local error estimator (LEE) which is then used to evaluate the refinement metric R.

Unfortunately, these approaches are highly computationally expensive due to the linear increase of the number of training processes required by CV as samples are added to the training dataset D_T (section A.2). This effect on sampling time t_{sampl} is magnified when nonlinear SMs are used, as highlighted by Garbo in [30].

In addition, the necessity of using an SM to compute the sampling metric makes this class of techniques "model dependent" (MDSAS). The use of an inappropriate SM can



(a) LNA test function contour plot

(b) Resulting sample distribution

Figure 2.14: Example of sample distribution obtained with a CV-based sequential adaptive sampling technique

lead to an inefficient sample distribution as described in [30, 31], requiring an active SM selection process to increase the robustness of this class of sampling techniques [30, 31].

An example of a sample distribution obtained using a CV-based technique is shown in figure 2.14, where the balance between exploration and refinement is clearly visible: more samples are placed in the middle lower region of the design space (figure 2.14b) where the test function has a higher variability (figure 2.14a).

A list of CV-based sequential adaptive sampling strategies with their relative references is given in table 2.5. Unfortunately, to the best of author's knowledge, all the CV-based sampling techniques work in a single-selection mode, and therefore they are not efficient when HPC resources are available.

The following paragraph presents the cross validation variance adaptive sampling (CVVAS) strategy which is the MDSAS representative in both the study about model dependence consequences on sampling behavior (chapter 4) and final performance result comparison (chapter 6).

cross validation variance adaptive sampling (CVVAS) cross validation variance adaptive sampling (CVVAS) is a CV-based technique developed by Lam ([16]), but similar approaches have been proposed by Jin ([47]) and Kleijnen ([53]). The identification of the

Table 2.5: List of some existing CV-based sequential adaptive techniques

Name	**References**
Variance based	[16, 47, 53]
Jackknife based	[53]

design space regions in need of a sample refinement is guided by the leave-one-out cross validation variance ($\sigma^2_{\mathrm{LOO-CV}}$):

$$\sigma^2_{\mathrm{LOO-CV}}(\mathbf{x}) = \frac{1}{n_T}\sum_{i=1}^{n_T}(\tilde{f}(\mathbf{x})^{(-i)} - \tilde{f}(\mathbf{x}))^2 \tag{2.2}$$

where $\tilde{f}(\mathbf{x})^{(-i)}$ is the LOO-CV SM trained without considering the i-th training data of D_T (Figures 2.15a and 2.15b). The new sample to include in D_T is the design space location that maximizes the modified leave-one-out cross validation variance ($\tilde{\sigma}^2_{\mathrm{LOO-CV}}$):

$$\mathbf{x}_{\mathrm{new}} = \mathrm{argmax}(\tilde{\sigma}^2_{\mathrm{LOO-CV}}) \tag{2.3}$$

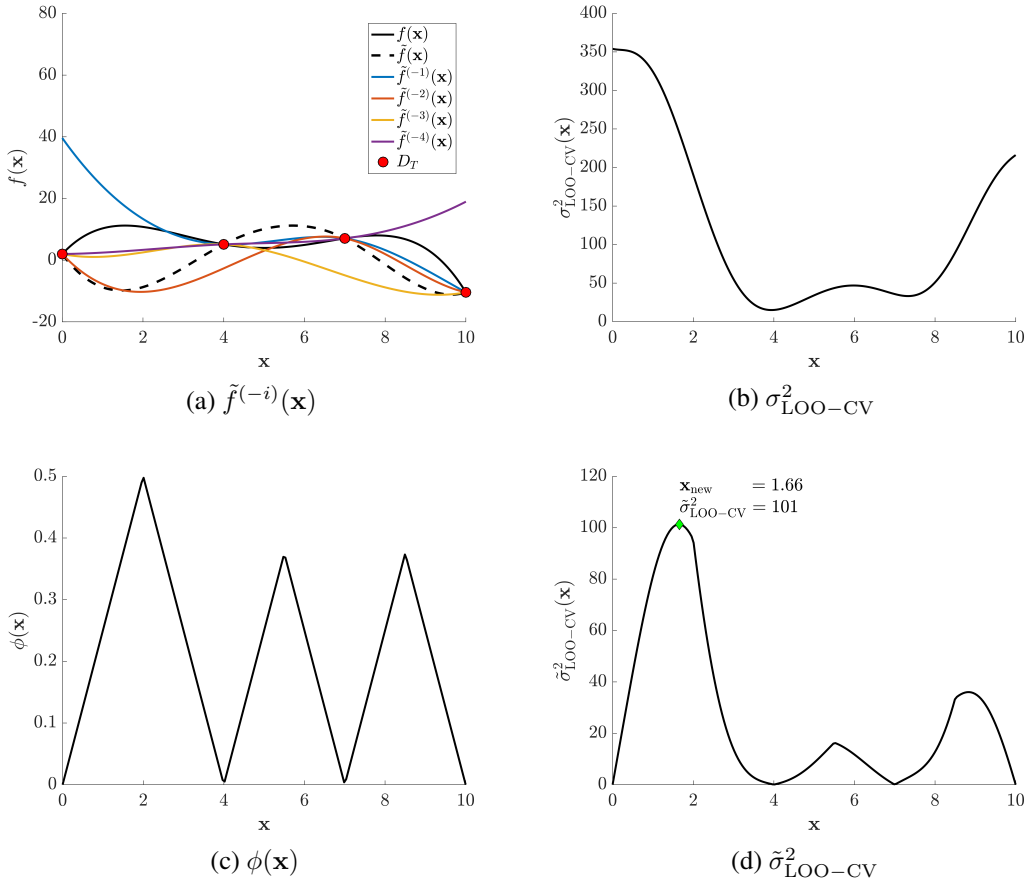
where

$$\tilde{\sigma}^2_{\mathrm{LOO-CV}}(\mathbf{x}) = \sigma^2_{\mathrm{LOO-CV}}(\mathbf{x}) \cdot \phi(\mathbf{x}) \tag{2.4}$$

and $\phi(\mathbf{x})$ is the penalty function which enforces the spread of the points by penalizing the locations that are too close to existing samples. Indeed, it often happens that the point with the highest $\sigma^2_{\mathrm{LOO-CV}}$ coincides with a design already included in D_T, as shown in figure 2.15b. The penalty function $\phi(\mathbf{x})$ (2.5) is specified as the distance between the candidate point $\mathbf{x}$ and the closest existing sample, normalized by the maximin distance between all existing training points:

$$\phi(\mathbf{x}) = \frac{\bar{d}_{\min}(\mathbf{x})}{\bar{d}_{\mathrm{MaxMin}}} \tag{2.5}$$

Once the location of the new training point $\mathbf{x}_{T,i+1}$ has been identified, the corresponding response $f(\mathbf{x}_{T,i+1})$ is evaluated, and this information is appended to D_T. The process is



(a) $\tilde{f}^{(-i)}(\mathbf{x})$ (b) $\sigma^2_{\mathrm{LOO-CV}}$

(c) $\phi(\mathbf{x})$ (d) $\tilde{\sigma}^2_{\mathrm{LOO-CV}}$

Figure 2.15: Example of the $\tilde{\sigma}^2_{\mathrm{LOO-CV}}$ sampling strategy adopted in this study

repeated by adding one sample at the time using the most updated D_T until a convergence criterion is reached.

An example of the $\tilde{\sigma}^2_{\mathrm{LOO-CV}}$ sampling strategy is shown in figure 2.15. figure 2.15a shows the true function $f(\mathbf{x})$ and the LOO-CV SMs $f^{(-i)}(\mathbf{x})$, figure 2.15b shows the value of $\sigma^2_{\mathrm{LOO-CV}}$ evaluated using eq.(A.13), figure 2.15c shows the distance penalty function $\phi(\mathbf{x})$, and figure 2.15d plots the sampling function $\tilde{\sigma}^2_{\mathrm{LOO-CV}}$. Considering figure 2.15d, the next training point should be located around $x = 1.66$. Clearly, a different location would have been found if a different SM formulation had been used: the choice of an SM affects the shape of $\tilde{f}^{(-i)}$ and consequently the sampling metric $\tilde{\sigma}^2_{\mathrm{LOO-CV}}$, making this technique model dependent. This characteristic is confirmed by the results described in section 4.3.2.

To implement this strategy, the initial training dataset $D_{T,\text{in}}$ required to "ignite" the sampling process is generated by the MATLAB routine `lhsdesign` with 20 iterations, and a sample size set equal to two times the problem dimensionality D, which corresponds to 20% of the typically-suggested final sample size of $10D$ [63]. Due to the multi-modal characteristic of the objective function $\tilde{\sigma}^2_{\text{LOO}-\text{CV}}$ (figure 2.15d), the maximization problem is solved using a genetic algorithm (MATLAB `ga`) coupled with a gradient based strategy (MATLAB `fmincon`): the optimal $\mathbf{x}$ returned by the genetic algorithm is used as initial point condition for the gradient based search.

Voronoi-based Sequential Adaptive Sampling This class of sequential adaptive sampling techniques sequentially adds samples to $\mathbf{X}_T$ by using local gradient approximations to evaluate R and an estimation of the Voronoi tessellation volumes to enforce exploration. This class of sampling strategies is the inspiration basis for the technique proposed in this dissertation. The main positive aspect of these approaches is their "model independence" meaning that no global SM is used to guide the sampling process, making this technique a model independent sequential adaptive strategy (MISAS). This fundamental characteristic both relieves the user from the burden of SM selection and removes the need of an active SM selection in the sampling process like in CV-based techniques. The Voronoi-based formulations are usually easy to understand and to implement, and the algorithms seem to have an high degree of robustness.

A specific Voronoi-based strategy called LOLA-Voronoi developed by Crombecq [20] is considered to briefly analyze some crucial aspects of this type of technique. LOLA-Voronoi formulation assigns a score H to each sample $\mathbf{x}_T$ based on both its Voronoi tessellation volume and a metric that assesses the nonlinearity of its neighborhood:

$$H(\mathbf{x}_{T,i}) = V(\mathbf{x}_{T,i}) + \bar{E}(\mathbf{x}_{T,i}) \tag{2.6}$$

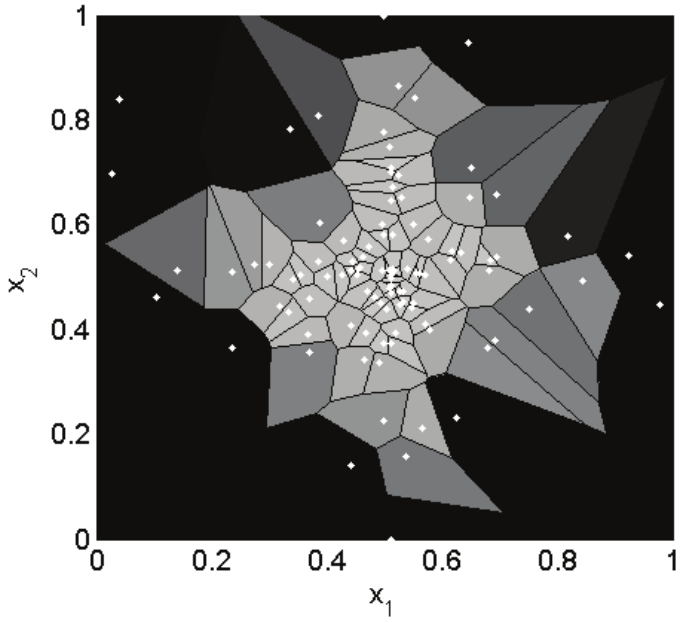
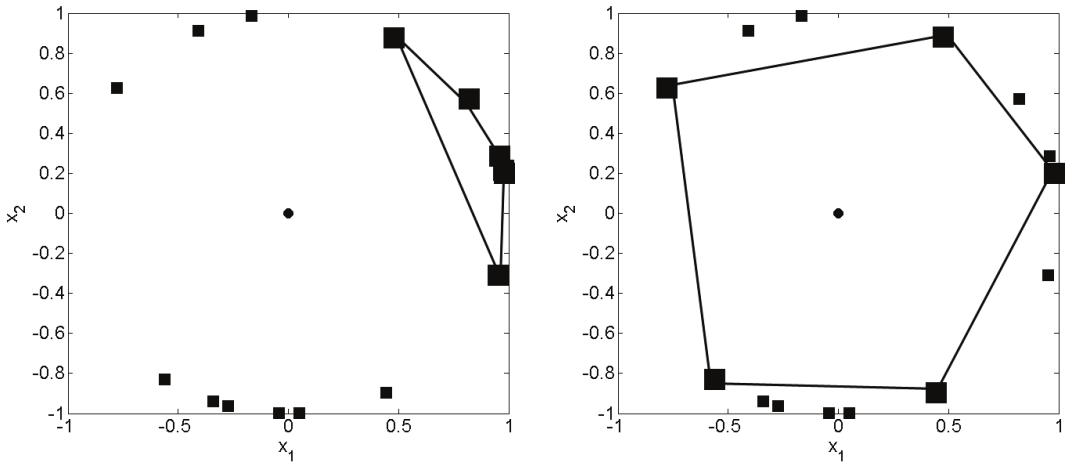


Figure 2.16: A set of data points and their Voronoi cells in a two-dimensional (2D) design space. Larger Voronoi cells have a darker color. The data points are drawn as white dots. Unbounded Voronoi cells are black [20]

where $V(\mathbf{x}_{T,i})$ is the volume of the Voronoi cell relative to $\mathbf{x}_{T,i}$, and $\bar{E}(\mathbf{x}_{T,i})$ is the metric that estimates the non linearity of $\mathbf{x}_{T,i}$ neighborhood. New samples are sequentially added by applying a space-filling approach on the Voronoi cell with the highest score.

The Voronoi cell volume V (an example of 2D Voronoi tessellation is reported in figure 2.16) is proven to be a good exploration metric, because cells in regions with a more coarse sample distribution have a higher value of V compared to the ones in a more finely sampled region. Unfortunately, performing the Voronoi tessellation is prohibitive for high dimensional problems (at worst $\Omega(n^{D/2})$ [51]), and therefore the LOLA-Voronoi technique estimates the cell volume by a Monte-Carlo approach.

The term $\bar{E}(\mathbf{x}_{T,i})$ in equation (2.6) is responsible for the detection of Voronoi cells where the nonlinearity of the response requires a sample refinement. Even though a detailed description of the $\bar{E}$ formulation is provided in [20], it is important to discuss aspects of the formulation to identify certain negative characteristics that the technique proposed in this dissertation is intended to mitigate. The evaluation of $\bar{E}$ requires the identification of the



(a) Bad neighborhood (b) Ideal neighborhood

Figure 2.17: Example of bad and ideal neighborhood [20]

"ideal neighborhood" $N^{(i)}$ for each sample point $\mathbf{x}_{T,i}$ in $\mathbf{X}_T$ [20]. The samples in such a neighborhood must lie relatively close to the reference sample to be meaningful (*cohesion*), but they must also lie far away from each other in order to cover each direction as equally as possible (*adhesion*) [20]. A complete description of the metrics required to accomplish the identification of the ideal neighborhood is beyond the scope of this section, but a discussion is available in the original article [20]. Figure 2.17 clearly illustrates examples of a bad (figure 2.17a) and an ideal (figure 2.17b) neighborhood containing $m = 5$ points [20].

Once the ideal neighbors have been identified for all the samples in $\mathbf{X}_T$, the gradient is estimated at each sample $\mathbf{x}_{T,i}$ by fitting a hyperplane through $\mathbf{x}_{T,i}$ and its ideal neighborhood $N^{(i)}$. The original formulation suggests using $m = 2D$ (where D is the problem dimensionality) points to define each neighborhood $N^{(i)}$, meaning that the hyperplane fitting problem is overdetermined: there are $D + 1$ unknowns (hyperplane equation in D dimensions) and $2D + 1$ equations (number of ideal neighbors and the considered reference point). The approach followed by Crombecq [20] keeps the overdetermined system, but he forces the hyperplane to pass through $\mathbf{x}_{T,i}$. The coefficients computed by the solution of the hyperplane regression are then used to estimate the gradient $\mathbf{g}$ at $\mathbf{x}_{T,i}$.

Finally, the local nonlinearity $E\left(\mathbf{x}_{T,i}\right)$ is estimated at each sample point using the equation (2.7) which "computes how much the response at the neighbors differs from the local linear approximation" [20]:

$$E\left(\mathbf{x}_{T,i}\right) = \sum_{k=1}^{m} \| y_{N_i,k} - \left(y_{T,i} + \mathbf{g}_i \cdot \left(\mathbf{x}_{N^{(i)},k} - \mathbf{x}_{T,i}\right)\right) \| \tag{2.7}$$

where the subscript $N^{(i)}$ is relative to the ideal neighborhood of sample point $\mathbf{x}_{T,i}$. The normalized version $\bar{E}$ is computed as:

$$\bar{E}(\mathbf{x}_{T,i}) = \frac{E\left(\mathbf{x}_{T,i}\right)}{\sum_{j=1}^{n_T} E\left(\mathbf{x}_{T,i}\right)} \tag{2.8}$$

Even though the LOLA-Voronoi technique may seem complex from this brief description, a careful reading of the original article [20] reveals the fundamental simplicity of the strategy. The attractive characteristics of the LOLA-Voronoi are:

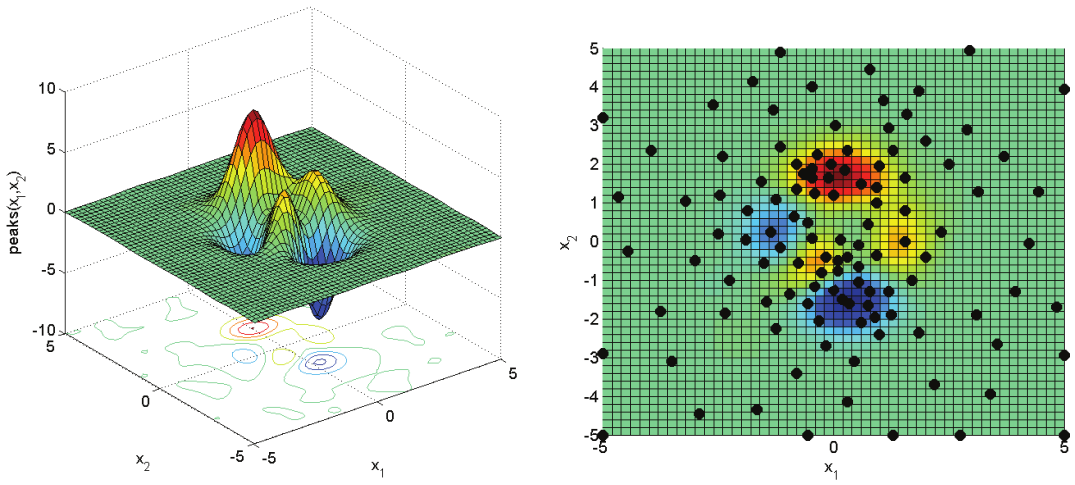
Model independent This technique requires the fitting of local hyperplanes, and it does not need the selection of a global SM functional form.

Parallelizable The local hyperplane fitting problem can be easily parallelized using HPC resources.

Conceptually simple The basis of the LOLA-Voronoi sampling is conceptually simple and well described in the original paper [20].

Robust The mathematical operations involved in the process are simple, and their relative algorithms are well developed and robust.

The description of LOLA-Voronoi strategy includes an extended algorithm capable of dealing with multi-response problems and to sample in a batch mode, but to the best of author's knowledge it has never been tested in such applications. Additionally, some pitfalls affect the LOLA-Voronoi formulation. First, the statement that "the ideal neighborhood provides



(a) Surface representation of the test function

(b) Resulting sample distribution using LOLA-Voronoi technique

Figure 2.18: Example of LOLA-Voronoi technique application [20]

Table 2.6: List of some existing Voronoi-based sequential adaptive techniques

Name	**References**
Voronoi Sampling	[18]
LOLA-Voronoi	[20]
Fuzzy logic LOLA-Voronoi	[44]

more information about the behavior of the system around the reference sample than other neighborhoods" [20] is not proved or empirically shown. Secondly, the computational complexity of the ideal neighborhood identification algorithm grows with $\mathcal{O}(n_T^2)$ [20], and it can severely slow down the sampling process when n_T is large.

An example of the resulting sample distribution obtained using a LOLA-Voronoi algorithm is shown in figure 2.18, and a list of Voronoi-based sequential sampling techniques with relative references is reported in table 2.6.

2.4 Motivations for a New Sampling Technique

Section 1.6 lists six characteristics of a good sampling strategy for engineering design applications: adaptive, sequential, flexible, simple and robust, with low computationally complexity, and model independent. Additionally, Liu in [60] identifies other two desired

Table 2.7: Summary of the characteristics of existing classes of sampling strategies (O-S *one-shot*, Seq. *sequential*, S-F *space filling*, Ad. *adaptive*)

Sampling Class	**Adap.**	**Seq.**	**Flex.**	**Simp. & Rob.**	**Low Comp. Compl.**	**Mod. Indep.**	**Batch Sel.**	**Multi-res.**
O-S S-F	no	no	no	yes	yes	yes	yes	no
Seq. S-F	no	yes	no	yes	yes	yes	yes	no
O-S Ad.	yes	no	yes	some	some	some	yes	no
Seq. Ad. CV-based	yes	yes	yes	no	no	no	no	no
Seq. Ad. Cov-based	yes	yes	yes	no	no	no	no	no
Seq. Ad. Vor-based	yes	yes	yes	yes	no	yes	N/A	N/A

features for a sampling technique: the capability to work in a batch selection mode, and the ability to handle multi-response problems. These eight requirements are compared in table 2.7 for all the classes of sampling strategy described in the first part of the chapter: as apparent in the table, no sampling class is able to meet all the recognized requirements.

In particular, the space filling approaches are really powerful when a simple and computationally inexpensive technique is desired, but the absence of the adaptive formulation implies an increase in the number of evaluations required to accurately sample the design space. The one-shot adaptive techniques include the refinement metric in their formulations, but their sampling efficiency is extremely dependent on the quality of the initial response knowledge. In addition, the more advanced techniques (Covariance-based) are model dependent, a characteristic that can affect both the sampling efficiency and the total process time.

Considering only the quality and the efficiency of the sample distribution, the sequential adaptive techniques are the most effective ones. They embody a refinement metric, and they update the knowledge about the response every time a new sample is added to the training data set. Unfortunately, these sampling strategies are usually model dependent and they require complex and computationally expensive algorithm for the refinement metric evaluation. Most of the requirements are met by the Voronoi-based sequential adaptive class, which is able to synthesize a refinement metric without the need for a global SM by using a local approximation of the response gradient. Unfortunately, the Voronoi-based

formulations are still affected by excessive computational complexity, even though it is significantly reduced when compared to the other sequential adaptive techniques. Moreover, none of the sequential adaptive strategies has shown the ability to work in batch selection mode, making these techniques probably inefficient when HPC resources are available.

After the analysis of the characteristics of existing formulations, it is apparent that a technique that satisfies all the requirements for a good sampling strategy for engineering design is missing. A strategy that embodies all these characteristics will be an extremely powerful tool for early phases of the design process, and its development is the main objective of this dissertation.

In addition, the review of state-of-the-art sequential adaptive techniques (section 2.3) showed how all refinement metrics identify the regions where additional samples are required without providing any information about particular directions in the design space along which the samples should be concentrated. Making an analogy with the optimization field, these techniques are similar to sequential grid-search optimization algorithms, where a refinement is made around the optimal point of a coarser grid. Considering the performance improvements that can be achieved when gradient information is included in optimization algorithms, we speculate that the introduction of directional information will also improve the efficiency of the sampling algorithm for SM creation. To the best of our knowledge, this field is completely unexplored, and therefore the development of this sampling feature is one of the main objectives of this dissertation.

Furthermore, no sequential adaptive formulation has been developed and tested to be usable in multi-response applications. A literature review shows that strategies have been defined to treat either general [6, 19, 58, 62, 64, 90, 94], or symmetric [56, 78, 92, 93], or asymmetric [7, 42, 85] multi-response problems by single-response algorithms, but no sampling techniques have been developed to directly handle multi-response problems [60]. Additionally, there are not available results about sampling processes conducted in batch mode or in situations with critical errors in the solver. For this reason, the final version

of the proposed sampling technique includes the capability of dealing with multi-response and critical error situations, and it can be used in either single-selection or batch mode.

CHAPTER 3

DISSERTATION OBJECTIVES

The present chapter summarizes the hypotheses, the motivations, and the objectives of this dissertation; a quick overview is reported in table 3.1.

Table 3.1: Dissertation objectives, motivations, and hypotheses

Objectives	**Motivations**	**Hypotheses**
Study model dependence effect on adaptive sampling	No comprehensive study available about consequences of model dependence in adaptive sampling strategies	Sampling efficiency of model dependent sequential adaptive techniques reduces if an inappropriate SM is used
NNAS development	Create a sampling technique suitable for early phases of engineering design	NNAS meets all the requirements for a good sampling technique
Directional sampling	No existing directional sampling technique	Increase in sampling performance
Batch selection mode	HPC resources	
Multi-response formulation	Multi-response problems are common in engineering design	
Formulation unaffected by solver critical errors	Critical errors commonly occur in engineering simulation codes	

3.1 Objective 1

Study the effect of model dependence in the sample distributions obtained by model dependent sequential adaptive strategies.

3.1.1 Motivation

As seen in the previous chapter, model dependent sequential adaptive strategies (MDSASs) use an SM to synthesize the available information in D_T and to compute the refinement metric R, thereby causing the resulting sample distribution to be influenced by the particular SM choice. Literature about MDSASs lacks a complete analysis of model dependence effects on the representation efficiency of resulting sample distributions. In other words, there is not a study able to address the following question:"What happens if the SM selected a priori to supervise the sampling process is inaccurate in modeling the unknown response?"

3.1.2 Hypotheses

1. The representation efficiency of sample distributions obtained by an MDSAS is substantially reduced if the supervising SM functional form is inappropriate to model the unknown response.

2. A sampling architecture which couples an MDSAS with an active SM selection mitigates the reduction in sampling efficiency caused by model dependence.

3.1.3 Tasks

1. use an MDSAS to repeatedly sample the design space of several responses using different SM functional forms;

2. compare the sampling performance;

3. repeat the sampling processes by coupling the MDSAS with an active SM selection process.

3.2 Objective 2

Develop an adaptive-sequential sampling technique based on a local linear model: Nearest Neighbors Adaptive Sampling.

3.2.1 Motivation

A technique that is adaptive, sequential, simple, robust, computationally efficient, and model independent is missing in the state-of-the-art sampling strategies. All these properties are fundamental for engineering design applications.

3.2.2 Hypothesis

An adaptive-sequential sampling strategy based on local linear models which are created from information about nearest neighbors will meet the requirements to be suitable for engineering design applications, leading to a reduction of both the required number of samples and computational time.

3.2.3 Tasks

1. define the exploration and refinement metrics;
2. define the stochastic selection approach based on Pareto-ranking to achieve a refinement-exploration balance;
3. assess the NNAS sampling performance on several example functions from low to high dimensionality, and from low to high complexity;
4. compare the NNAS performance with the results obtained using other adaptive-sequential techniques, in terms of both number of samples and computational time.

3.3 Objective 3

Introduce *directional sampling* into the NNAS technique

3.3.1 Motivation

Current model-independent sequential adaptive sampling strategies add samples in the region of interest identified by a refinement metric, but they do not guide the sampling along a specific direction.

3.3.2 Hypothesis

Prediction errors in the neighborhood of each sample can be used to guide the sampling behavior, leading to a reduction in the required number of samples at the price of a limited increase in computational cost.

3.3.3 Tasks

1. expand the NNAS algorithm to use the prediction error in the neighborhood of each training point as a local refinement metric;
2. define a criterion to balance local refinement and exploration;
3. assess the performance of the directional NNAS implementation on several example functions from low to high dimensionality, and from low to high complexity;
4. compare the NNAS performance with the results obtained using other adaptive-sequential techniques and the standard NNAS, in terms of both number of samples and computational time.

3.4 Objective 4

Extend NNAS to be used in batch selection mode

3.4.1 Motivation

The wide availability of high performance computing (HPC) resources makes it possible to simultaneously run several simulations making the batch selection mode a requirement for modern and future engineering applications.

3.4.2 Tasks

1. define the strategy to identify a batch of sample locations;
2. assess the performance of the batch mode NNAS on several example functions from low to high dimensionality, and from low to high complexity;
3. compare the batch mode NNAS performance with the results obtained using single selection mode NNAS, in terms of both number of samples and computational time.

3.5 Objective 5

Extend NNAS to work with multiple responses

3.5.1 Motivation

A common engineering design has to deal with several outputs at the same time (i.e. lift and drag of a wing). The ability to simultaneously handle multiple responses will extend the class of applications for NNAS.

3.5.2 Tasks

1. define a refinement metric able to handle multiple responses;

2. assess the sampling performance of multi-response NNAS implementations on several example functions from low to high dimensionality, and from low to high complexity;

3.6 Objective 6

Modify the NNAS formulation to be unaffected by critical errors in the computer simulation solvers

3.6.1 Motivation

Sampling strategies usually sample the design space of an unknown response, and therefore it may happen that a particular design is infeasible or the standard solver settings are not suitable for that particular condition. In these cases, the solver used to assess the response value at that particular design point is likely to return a critical error, meaning that no y_T is returned to the sampling algorithm. A robust sampling algorithm should be able to continue the sampling process without being affected by solver critical errors, and also it should give to the user the opportunity to adjust adjusting the simulation and refresh the training dataset without stopping the sampling process.

3.6.2 Tasks

1. reformulate the refinement metric such that it is not affected by missing response training values;

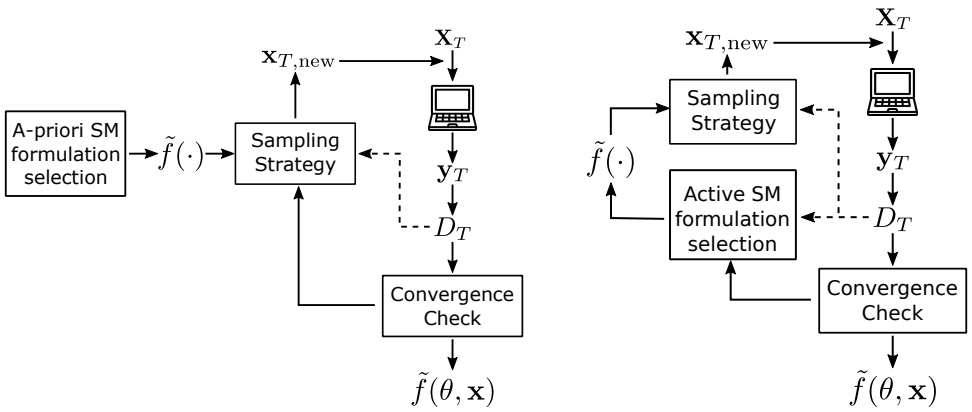
2. assess the sampling performance on several example functions in which a portion of the design space is infeasible, i.e. the evaluation of f returns a critical error.

CHAPTER 4
SAMPLING STRATEGY MODEL DEPENDENCE

This chapter presents a study that has been conducted to investigate the effects of model dependence on the sampling efficiency and behavior of model dependent sequential adaptive strategies (MDSASs). As described in section 2.1.3, these techniques use refinement metrics which are computed from an SM that is trained using the available D_T, making the sampling behavior dependent on the considered SM functional form. Some examples of state-of-the-art MDSASs are the Kriging-based Integrated Mean Squared Error (IMSE) and Maximum Entropy (ME) ([39, 98]), or the ones based on local error estimators ([47, 53, 61]). To the best of the author knowledge, no studies have been conducted to assess the importance of a careful SM selection when such a technique is used for the sampling phase of a surrogate modeling process.

The goal of the first part of this chapter is to investigate to what extent model dependency can affect the performance of a sequential-adaptive sampling technique. In other words, what happens if the chosen SM functional form which is selected *a priori* is not the most appropriate to model the initially unknown response f? To answer this question, different a priori SM formulations (figure 4.1a) are used with a model dependent sequential adaptive strategy, and the number of samples required to obtain a certain level of approximation accuracy is compared. The results (section 4.3.2) clearly show how the choice of the SM significantly affects the sampling performance: the expected reduction in the number of required samples may vanish if an inappropriate SM formulation is adopted in the process.

A possible solution to mitigate the consequences of model dependency is presented and tested in the second part of this study. The same MDSAS is coupled with an active SM selection technique (figure 4.1b) which chooses the SM formulation that best models



(a) A-priori SM selection (b) Active SM selection

Figure 4.1: Comparison of different surrogate modeling architectures

the existing and most updated D_T (with respect to a chosen accuracy metric). The results (section 4.3.3) confirm how this architecture enhances the "robustness" of the surrogate modeling process performance in terms of the number of samples required to obtain an accurate SM. This architecture inherits the advantages of both constituent elements: the sequential adaptive sampling strategy efficiently uses all the available information to effectively sample the design space, and the active SM selection identifies the "best" formulation to represent the training dataset, D_T, relieving the user from the burden of choosing a good SM a priori.

The MDSAS considered in this study is the CVVAS presented in section 2.3.4. This technique is tested with and without active SM selection for the sampling phase of nine two-dimensional and two five-dimensional examples (table 4.1), representing both simple analytic functions and engineering models (e.g. an equation describing the admittance of a low noise amplifier). The suite of tested SMs includes five RBFs, each with a different kernel function, and an OKRG model (appendix B); this test suite is guided by the intention of considering models with different levels of formulation complexity and modeling capability. For an actual engineering application of the technique, the selection of the formulations to include in the suite of SMs should be guided by historical data or experience in modeling similar responses. Indeed, the choice of SM formulations for particular problems is still a

challenging open question in the literature about SMs for engineering applications. The influence of other parameters – specifically the SM selection criterion and frequency – on the number of samples required to complete the modeling process is also investigated to define useful guidelines for the application of the method.

At this point, it is important to underline one aspect of this research. The leave-one-out cross validation (LOO-CV) is the only cross-validation formulation tested in this study because it was the approach adopted in sampling techniques similar to the one considered in this research ([16, 47, 53]). However, the sampling strategy can be easily extended to use other CV formulations that are available in literature, i.e. k-fold or leave-k-out ([69]), which are particularly useful when there is a large number of training samples and the LOO-CV approach may become extremely computationally expensive.

This chapter is organized as follows: section 4.1 presents the proposed architecture with the active SM selection process, section 4.2 lists the settings used to complete the analysis, and finally the results are presented in section 4.3.

4.1 Sampling with an Active Surrogate Model Selection Architecture

The SM dependence of the sampling behavior leaves an open question: how should the SM technique for the sampling phase be selected? It is widely agreed ([33, 71, 84, 87, 105, 107]) that it is impossible to devise a SM technique that outperforms all others in terms of approximation quality for all possible applications. The natural approach is therefore to select the SM formulation that better represents the training dataset D_T by minimizing a global error estimator (GEE) [87]. Therefore, the problem of SM selection can be stated as follows: given a set of possible SMs and a GEE, find $i_{\min}$ such that $\operatorname{argmin} \mathrm{GEE}(\mathrm{SM}_i) = \mathrm{SM}_{i_{\min}}$.

The proposed sampling architecture (figure 4.1b) couples a model dependent sequential adaptive strategy with the previously described SM selection approach. The goal is to mitigate the effect of the model dependence by actively selecting the "best" SM (relative to

a GEE) to use in the sampling phase. The general algorithm of such a sampling architecture can be summarized as follow (figure 4.1b):

1. Define a suite of SM formulation options.
2. Select the metric that will be used to rank the SM accuracy.
3. Create the initial $\mathbf{X}_T$ using a classical DoE strategy such as maximin Latin hypercube.
4. Evaluate the function at $\mathbf{X}_T$ ($\mathbf{y}_T = f(\mathbf{X}_T)$) to create the training dataset D_T.
5. At the beginning of the sampling process and after ϕ_{upd} samples have been added since the last SM selection, train all candidate SMs using D_T and select the best SM based on the accuracy metric chosen in step 2.
6. Run the selected sequential adaptive strategy to identify the next sample point, and update $\mathbf{X}_T$.
7. Repeat from 4 until a threshold level of SM quality is reached.

The value ϕ_{upd} is the "frequency of SM selection", or in other words the number of samples added to D_T between each SM selection. For example, if $\phi_{\text{upd}} = 10$ and the initial D_T has 4 samples, the selection of the SM formulation used to compute $\tilde{\sigma}^2_{\text{LOO-CV}}$ is done at the beginning of the process and when there are $14, 24, 34, \ldots$ samples in D_T.

Among all the possible GEEs ([46, 71, 87]), the ones considered in this study are two versions of Root Mean Squared Error (RMSE), one based on LOO-CV (eq.(4.1)) and one based on additional validation points (eq.(4.2)). To obtain comparable results, the RMSEs are normalized as indicated in eq.(4.3).

$$\text{RMSE}_{\text{CV}} = \sqrt{\frac{1}{n_T}\sum_{i=1}^{n_T}\left(y_{T,i} - \tilde{y}_i^{(-i)}\right)^2} \tag{4.1}$$

$$\mathrm{RMSE_V} = \sqrt{\frac{1}{n_V}\sum_{i=1}^{n_V}\left(y_{V,i} - \tilde{y}(\mathbf{x}_{V,i})\right)^2} \tag{4.2}$$

$$\mathrm{NRMSE} = \frac{\mathrm{RMSE}}{\max(\mathbf{y}_T) - \min(\mathbf{y}_T)} \tag{4.3}$$

4.2 Analysis Plan

The CVVAS sampling process is tested on 11 test functions (table 4.1, appendix D) first considering a single SM at a time (therefore without the SM selection process, figure 4.1a), and afterward using the proposed architecture with active SM selection (figure 4.1b) and the 6 SM options (5 RBFs, 1 OKRG). As previously introduced, this study investigates the influence of other parameters on the sampling performance, in particular:

- 2 ways of obtaining the leave-one-out (LOO) SMs when OKRG is considered. As described in section 2.3.4, CVVAS requires training the LOO SMs to compute the sampling metric (2.4), and this process can be highly computationally expensive in the case of a SM that requires non-linear training such as OKRG. For example, if D_T contains 100 points, it is necessary to train the 100 LOO OKRGs to compute the $\sigma^2_{\mathrm{LOO-CV}}$ required by eq.(2.4). A way to reduce this computational cost is to create a OKRG using the entire D_T, and reuse its hyperparameters to define the LOO OKRGs. In this way, only one OKRG has to be trained at each sampling iteration leading to a sensible reduction in the computational time.

- 5 different values of ϕ_{upd} (SM selection frequency, section 4.1): 1, D, $2D$, $5D$, and $10D$ where D is the dimensionality of the problem;

- 2 different GEEs as SM selection criterion (section 4.1): $\mathrm{NRMSE_{CV}}$ and $\mathrm{NRMSE_V}$. The latter requires a set of validation points which may lead to additional computational time if the evaluation of f is computationally expensive.

- 6 different numbers of validation points for the $\mathrm{NRMSE_V}$ selection criterion: $2D$, $4D$, $6D$, $8D$, $10D$, and $20D$.

Each sampling setting is repeated 15 times, each time with a different set of $2D$ starting points, and it is stopped when the training dataset D_T contains $50D$ points. The quality of the final SM is assessed using $\mathrm{NRMSE_V}$ based on $100D$ validation points and normalized respect to the real response range within the design space. The locations on the unit cube of the 15 sets of starting points and the sets of validation points used either for SM selection or final quality assessment are kept the same for problems with the same dimensionality. This avoids the introduction of additional aleatory effects which can be caused by the randomization of the validation points at each repetition. Additionally, the analysis about the LOO OKRG training approach and the number of validation points for the $\mathrm{NRMSE_V}$ selection criterion have been independently conducted to limit the number of sampling process runs required to complete the study. The summary of all the tested sampling configurations is reported in table 4.2.

Table 4.1: List of test functions with relative dimensionality and references

Function name	Dimensionality (D)	References
f2DBranin	2	[3, 31, 116]
f2DPeaks3	2	[20, 31]
f2DPeaks5	2	[20, 31]
f2DPeaks8	2	[20, 31]
f2DExponential	2	[16, 31]
f2DNonPolySurf	2	[16, 31, 70]
f2DSixHumpCamelBack	2	[16, 31, 70]
f2DLNA	2	[31, 38, 55]
f2DWitchHat	2	[31]
f5DCantilever	5	[31, 33, 110]
f5DLNA	5	[31, 38, 55]

Table 4.2: Setting summary

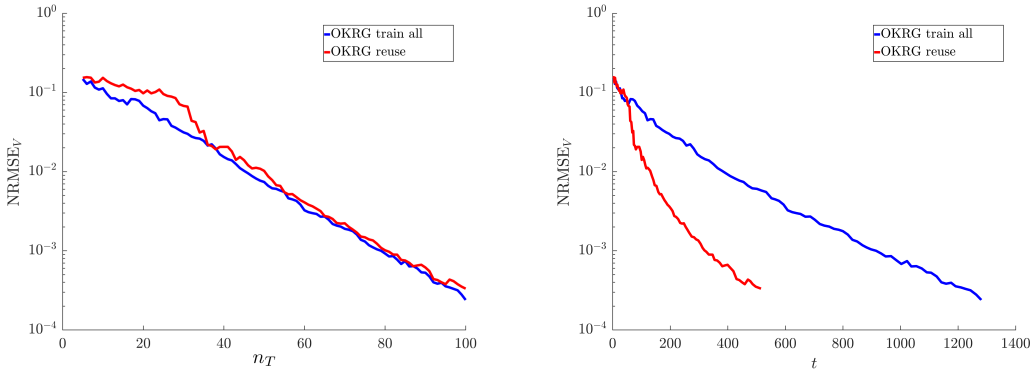
SMs		RBF M	RBF InvM	RBF G	RBF C	RBF TPS	GP retrain	GP reuse	All SMs													
SM sel. criterion									$\mathrm{NRMSE_{CV}}$					$\mathrm{NRMSE_V}$								
SM sel. frequency									1	D	2D	5D	10D	1						D	2D	5D	10D
number of val. points														2D	4D	6D	8D	10D	20D	4D	4D	4D	4D
Test functions	f2DBranin	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DPeaks3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DPeaks5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DPeaks8	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DExponential	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DNonPolySurf	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DSixHumpC.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DLNA	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f2DWitchHat	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	f5DCantilever	X	X	X	X	X		X	X	X	X	X	X		X					X	X	X	X
	f5DLNA	X	X	X	X	X		X	X	X	X	X	X		X					X	X	X	X

4.3 Results

4.3.1 Ordinary Kriging Hyperparameter Reuse

The results presented in this section compare the CVVAS sampling performance for the two approaches to obtain the LOO OKRGs: specifically with and without reuse of full model hyperparameters. This analysis is conducted only on two-dimensional functions because the results give a clear indication about the more efficient approach which is therefore used for higher dimensional problems.

The median $\mathrm{NRMSE_V}$ computed using the results of the 15 repetitions is reported in figure 4.2 as a function of both number of samples and computational time for the f2Branin test function (similar trends are obtained for the other two-dimensional example functions). As expected, the $\mathrm{NRMSE_V}$ rate of decrease with computational time is higher when LOO OKRGs are obtained by reusing the hyperparameters of the full OKRG. Indeed, the reuse of the hyperparameters prevent the independent training of all the LOO OKRGs leading to a relevant time saving. Additionally, the two approaches do not show significant differences when the median $\mathrm{NRMSE_V}$ is plotted versus the number of samples, suggesting that the independent training of the LOO OKRGs does not lead to any meaningful advantage in terms of the number of samples required to reach a prescribed accuracy level. A simi-



(a) $\mathrm{NRMSE_V}$ vs n_T (b) $\mathrm{NRMSE_V}$ vs time

Figure 4.2: $\mathrm{NRMSE_V}$ history for f2DBranin test function

lar behavior is observable in figure 4.3 which shows the box plots of both the number of samples and the computational time required to reach $\mathrm{NRMSE_V} < 10^{-2}$ for four different two-dimensional test functions: the performance in terms of number of samples is similar (first row of figures), while the difference in terms of computational time is significant (second row of figures).

Looking at these results, it is possible to conclude that computing the $\sigma^2_{\mathrm{LOO-CV}}$ by reusing the hyperparameters of the OKRG created using the entire D_T is much faster and does not consistently nor substantively degrade the sampling performance compared to re-training the hyperparameters for each LOO evaluation. For this reason, it has been decided to consider only the OKRG configuration with the reuse of the model hyperparameters for the subsequent analyses in this chapter. This decision helped to significantly reduce the computational time required to complete the study. Therefore, all the labels displaying OKRG henceforth refer to OKRG with the reuse of the hyperparameters.

4.3.2 Single SM performance

Before analyzing the results obtained coupling a model dependent adaptive sequential sampling strategy (CVVAS) with an active SM selection technique, it is important to see how much the model dependence of CVVAS is affecting the sampling performance when a sin-

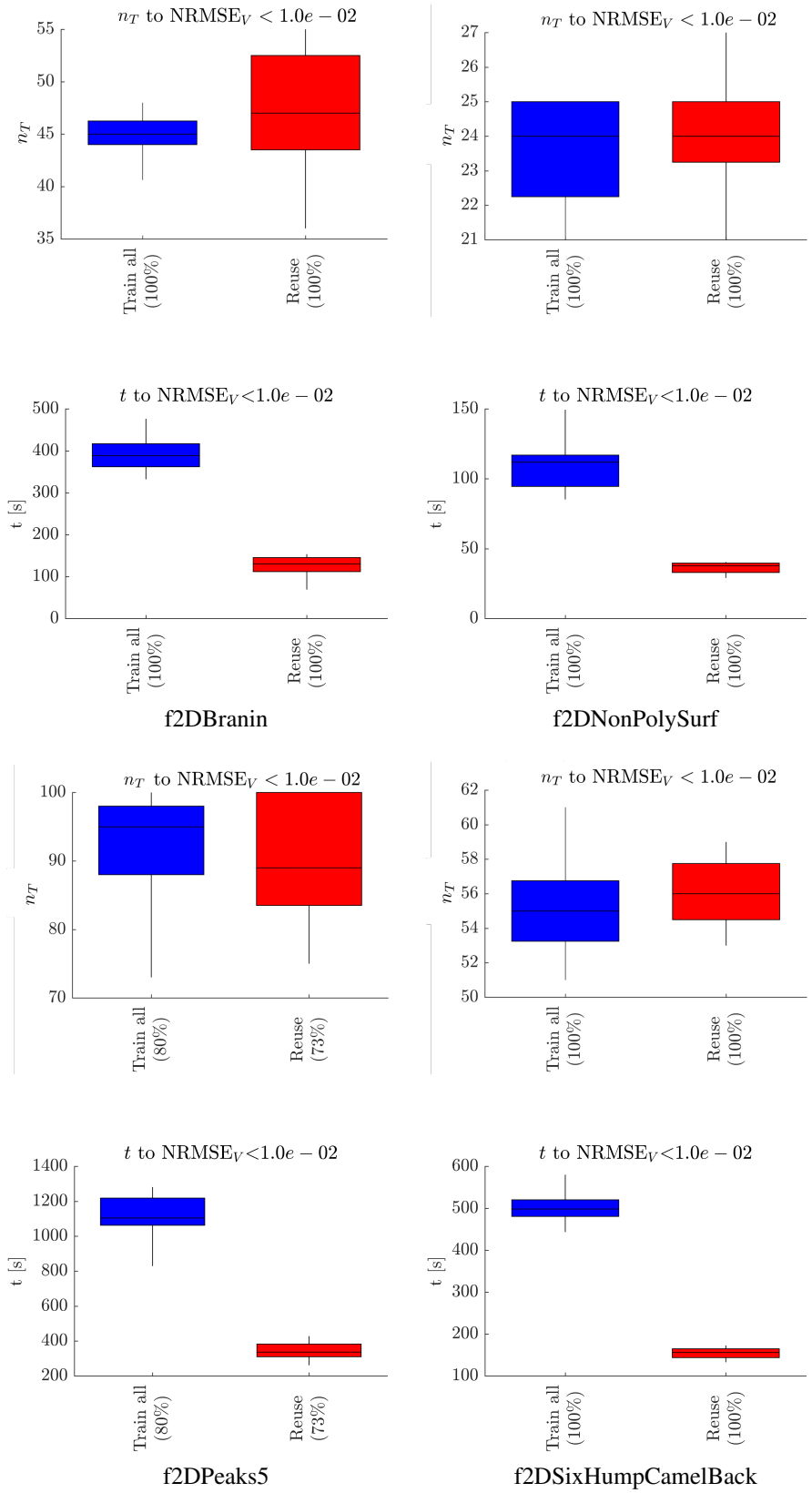
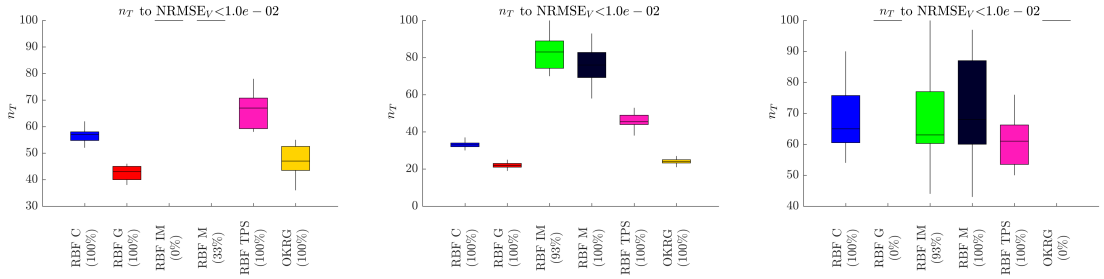


Figure 4.3: Boxplots of number of samples and time to reach $\mathrm{NRMSE_V} = 10^{-2}$ for four different test functions. The percentage of repetitions able to reach the $\mathrm{NRMSE_V}$ before the maximum number of samples is reported in parenthesis.



(a) f2DBranin (b) f2DNonPolySurf (c) f2DLNA

Figure 4.4: Number of samples to reach $\mathrm{NRMSE_V} < 10^{-2}$ considering a single a-priori chosen SM

gle SM is used in the process. In other words, what happens if CVVAS with the same SM technique is used to sample the design space of different responses? Is the performance consistent or does it have a high variability across the different test functions?

figure 4.4 shows the boxplots of the number of samples required to reach $\mathrm{NRMSE_V} < 10^{-2}$ when CVVAS is used without active SM selection for three different test functions. The bar labels indicate the SM formulation and the percentage of the 15 repetitions able to achieve the $\mathrm{NRMSE_V} < 10^{-2}$ accuracy level before the maximum number of samples convergence criterion ($50D$ samples) is reached. Figures 4.4a and 4.4b suggest that the Gaussian RBF and the OKRG are suitable models to use for these test functions; however, if one of these SMs is used to sample the response of the third test function which describes the behavior of a low noise amplifier, the process is not able to reach the required accuracy level within the $50D = 100$ training samples limit. This is a first clue of how risky it is to use a model dependent sampling strategy without carefully selecting the SM technique.

The need to create a single plot able to express the sampling performance variability across all the test functions leads to the definition of the $\bar{\Delta}_{\mathrm{samp}}$ metric. This metric is based on the fact that all the 15 sampling process repetitions are repeated using the same 15 sets of starting points. For each test function, we can therefore compare the performance of all the CVVAS-SM configurations when they start the process from the same initial condition. $\bar{\Delta}_{\mathrm{samp}}$ is intended to measure how many additional training samples each configuration

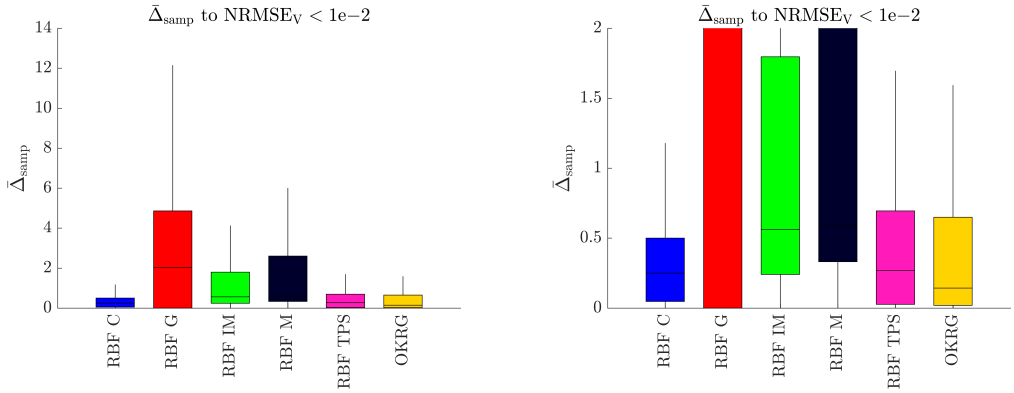
requires with respect to the best performance obtained with that initial condition. The mathematical formulation of the $\bar{\Delta}_{\text{samp}}$ metric for the i-th configuration is:

$$\bar{\Delta}_{\text{samp},i} = \frac{n_i - n_{\text{SM,best}}}{n_{\text{SM,best}}} \tag{4.4}$$

where n_i and $n_{\text{SM,best}}$ are the numbers of samples required to reach the accuracy level for the i-th configuration and for the best configuration respectively. The denominator is used to normalize the metric and to remove the dependency of $\bar{\Delta}_{\text{samp}}$ on the specific test function.

figure 4.5 reports the $\bar{\Delta}_{\text{samp}}$ boxplots when all the single SM tests are considered. These plots clearly reveal the significant variability in the number of training points required to achieve a $\text{NRMSE}_{\text{V}} < 10^{-2}$ accuracy if a single "a priori chosen" SM is used with CVVAS to sample all the test functions. For example, if CVVAS is used with RBF-TPS to sample the design space of all the test functions, 75% of the 15 repetitions (top of box in boxplots) requires up to 60% ($\bar{\Delta}_{\text{samp}} = 0.6$) more training points than the number required by the SM formulation which achieves the best performance on that particular test function and with that particular set of initial points (figure 4.5b). Similarly, if RBF-G is considered, the third $\bar{\Delta}_{\text{samp}}$ quartile is around 5, meaning that in 75% of the cases the sampling process can take up to 5 times more samples than the most appropriate SM technique. Limited to the SM options and test function considered in this study, these results indicate that RBF-C seems the option which gives the most "robust" sampling performance if CVVAS is used without active SM selection.

The pie charts in figure 4.6 are another way to view the performance variability caused by the CVVAS model dependence. They show the percentage of repetitions when a specific CVVAS-SM configuration reaches an accuracy level of $\text{NRMSE}_{\text{V}} < 10^{-2}$ with fewer samples than the other options for three test functions. (Again, this comparison is possible because all the configurations are tested with the same 15 sets of starting points). Even



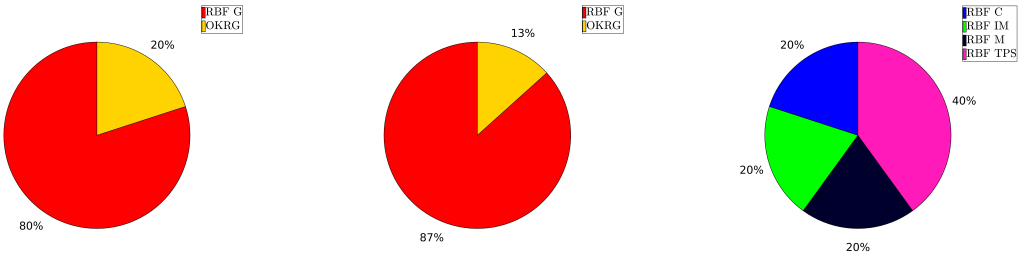
(a) No zoom (b) Zoomed

Figure 4.5: $\bar{\Delta}_{\text{samp}}$ to reach $\text{NRMSE}_{\text{V}} < 10^{-2}$ considering all the test functions

though RBF-G clearly prevails in two test functions (figures 4.6a and 4.6b), it is dominated by other CVVAS-SM configurations in the f2DLNA case (figure 4.6c). The pie chart obtained considering all the test functions is shown in figure 4.7, and – as expected – there is not a prevailing CVVAS-SM configuration. This plot seems to indicate that RBF-G could be a good initial choice if no information is available about the response to model, and this fact is in conflict with the results in figure 4.5 which indicate RBF-G as one of the worst options. This disagreement is caused by the fact that figure 4.7 is not showing how RBF-G performs in the 70% of the cases when it is not the best option, information that is instead included in the boxplot in figure 4.5 which clearly indicates how RBF-G would lead to poor sampling performance in most of the cases when it is not the most appropriate SM.

This modeling performance variability is mainly caused by the intrinsic capability of a SM to accurately model some specific class of functions. For example, OKRG built on a stationary covariance is expected to perform well in representing stationary responses such as f2DBranin and f2DNonPolySurf, and to perform pourly in modeling highly non stationary responses such as f2DLNA (figure 4.4).

The results shown in Figures 4.5 and 4.7 support the hypothesis that when a model dependent sampling strategy is used, the SM must be carefully selected because an inaccurate choice can negate the performance improvement expected by the adaptive sampling



(a) f2DBranin (b) f2DNonPolySurf (c) f2DLNA

Figure 4.6: Percentage of the 15 sampling repetitions when a specific SM reaches $\mathrm{NRMSE_V} < 10^{-2}$ with fewer samples than the other SMs for three different functions

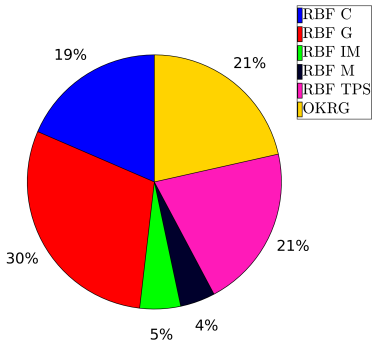


Figure 4.7: Percentage of sampling repetitions when a specific SM reaches $\mathrm{NRMSE_V} < 10^{-2}$ with fewer samples than the other SMs considering all the test functions

technique. Unfortunately, most of the time the shape of the response is unknown at the beginning of the process, and therefore it is challenging to select a priori the model to use. As shown in the next sections, this problem can be mitigated by including active SM selection in the sampling process.

4.3.3 Cross Validation Variance Adaptive Sampling with Active Surrogate Model Selection

The results presented in section 4.3.2 lead to the conclusion that no CVVAS-SM configuration is able to outperform the others across all the test functions, and this is the main motivation for including an active SM selection step in the sampling process (section 4.1, figure 4.1b).

As indicated in section 4.2, different numbers of validation points are tested when $\mathrm{NRMSE_V}$ is used as the SM selection criterion: in particular $2D$, $4D$, $6D$, $8D$, $10D$, and $20D$ points (where D is the problem dimensionality). The analysis of this parameter is conducted only on two-dimensional functions and with an update frequency ϕ_{upd} of 1 to limit the overall number of sampling process runs. As already stated in section 4.2, the sets of validation points are kept fixed to avoid any other aleatory effects. $\bar{n}_T$ is defined as:

$$\bar{n}_T = \frac{n_{T,m\cdot D}}{n_{T,2D}} \tag{4.5}$$

where $n_{T,m\cdot D}$ is the number of samples required to reach $\mathrm{NRMSE_V} < 10^{-2}$ (computed using a different set of $100D$ validation points (section 4.2)) by CVVAS with $\mathrm{NRMSE_V}$ as the SM selection criterion and $m \cdot D$ validation points. figure 4.8 shows the boxplots of $\bar{n}_T$ obtained considering all 15 sampling process repetitions for all the two-dimensional test functions. Surprisingly, there is a limited performance improvement (decrease of $\bar{n}_T$) as the number of validation points used for the SM selection increases. A more evident performance enhancement was expected due to the fact that a higher number of validation points should presumably help to better identify the most appropriate SM during the SM

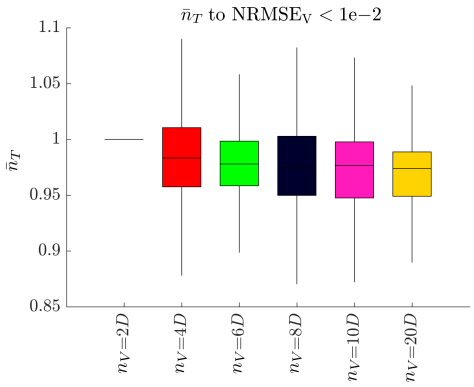


Figure 4.8: Normalized number of samples considering all the test function and 6 different values of number of validation points for $\mathrm{NRMSE_V}$ as SM selection criteria

selection phase. A thorough understanding of this outcome needs a deeper investigation of the dynamics between the sampling algorithm and the SM selection, an analysis that is not included here because out of the scope of this dissertation. Based on the median values in figure 4.8, the remaining sampling processes involving the $\mathrm{NRMSE_V}$ selection criterion have been conducted considering only $4D$ validation points.

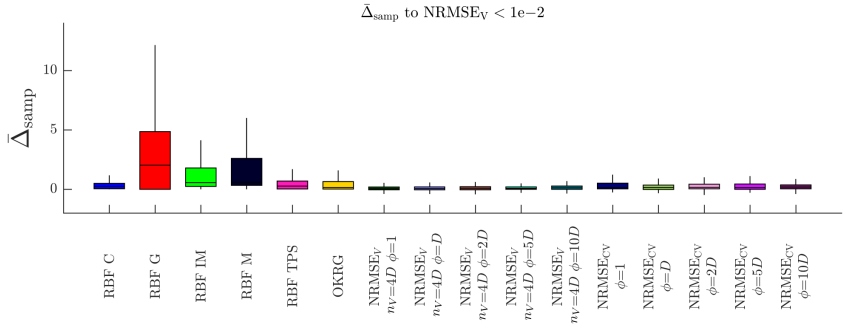
The boxplot of $\bar{\Delta}_{\mathrm{samp}}$ required to reach a $\mathrm{NRMSE_V} < 10^{-2}$ level of accuracy considering all the test functions and all the tested CVVAS-SM configurations (single SM, active SM selection with both $\mathrm{NRMSE_V}$ and $\mathrm{NRMSE_{CV}}$ criteria, and different ϕ_{upd}) is presented in figure 4.9. The first aspect that is immediately apparent is the sampling performance enhancement in terms of both robustness and total number of samples when the sampling phase is conducted with the active SM selection architecture. The use of this architecture for sampling different test functions makes it possible to not only reduce the required number of samples to reach the specified accuracy level (as indicated by the median being closer to zero) but also to reduce the performance variability (as indicated by the reduced box height representing the distance between the 25% and 75% quantiles). The reason for this improvement in the sampling performance has to be attributed to the capability of the active SM selection architecture to "learn" the most appropriate SM formulation to model the specific response. Surprisingly, in some cases, the number of samples required to reach the desired model accuracy is lower than the number needed by the most appropriate

SM technique for the particular test function and set of initial points, as indicated by the whiskers extending below the zero level.

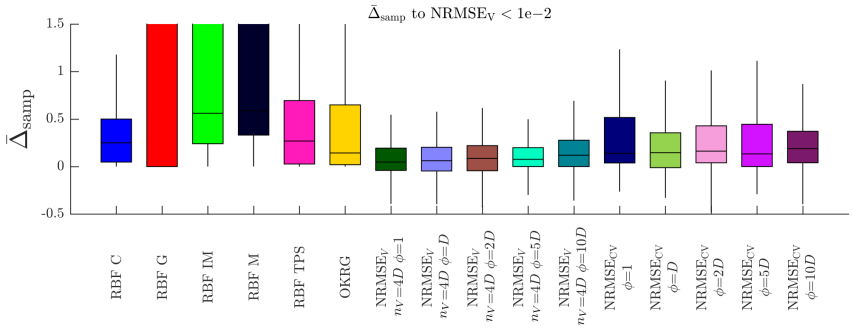
The increase in robustness due to active SM selection is also evident in figure 4.10 where the average evolution of $\mathrm{NRMSE_V}$ as samples are added to the training set is reported for three different test functions. As expected, the $\mathrm{NRMSE_V}$ has a decreasing trend which is consistent with prior observations of CVVAS convergence ([16, 47]); the different rates of convergence between sampling configurations are due to the differences in the inherent appropriateness of the functional form of each SM in modeling the particular test function. Additionally, the value of the $\mathrm{NRMSE_V}$ at a particular n_T obtained when CVVAS is coupled with an active SM selection (dashed line in figure 4.10) is always very close to the best single SM performance, irrespective of the test function and of which SM is the most suitable to model it. This behavior provides additional evidence of the convergence and improved robustness of the proposed sampling architecture which couples an adaptive sequential algorithm with active SM selection.

The second observation is about the performance of the two SM selection criteria adopted in this study: $\mathrm{NRMSE_V}$ and $\mathrm{NRMSE_{CV}}$. Considering figure 4.9b, it is clearly visible how using $\mathrm{NRMSE_V}$ (and $4D$ validation points as previously discussed) as selection criterion have slightly better performance than $\mathrm{NRMSE_{CV}}$. As preliminarily noticed in [30], this is another indication that better sampling performance is obtained when the metric used as the SM selection criterion is the same as the one used to assess the quality of the final SM. In other words, if a final SM with a low $\mathrm{NRMSE_V}$ is desired, $\mathrm{NRMSE_V}$ should be preferred over other accuracy metrics as SM selection criterion.

The last modeling parameter investigated in this study is the SM selection update frequency ϕ_{upd}. Focusing on figure 4.11 (which is an enlargement of the right part of figure 4.9), it is possible to notice how this configuration parameter generally seems to have a negligible effect on $\bar{\Delta}_{\mathrm{samp}}$. A minimally better performance may be obtained if $\phi_{\mathrm{upd}} = 1$ is used with $\mathrm{NRMSE_V}$ as the selection criterion, but it is difficult to identify a clear and

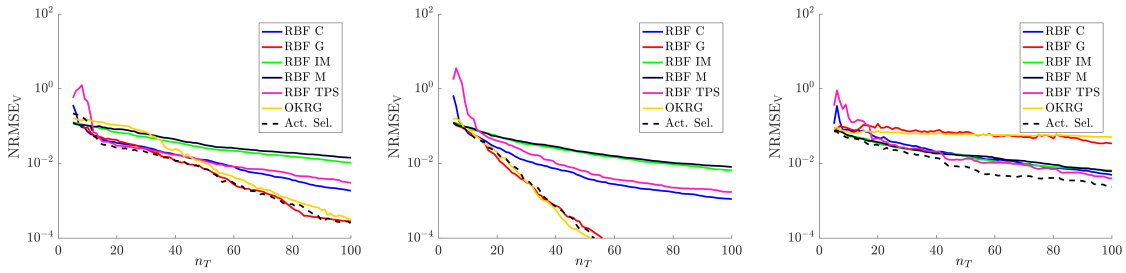


(a) No zoom



(b) Zoomed

Figure 4.9: $\bar{\Delta}_{\text{samp}}$ to reach $\text{NRMSE}_\text{V} < 10^{-2}$ considering all the test functions and all the sampling settings



(a) f2DBranin (b) f2DNonPolySurf (c) f2DLNA

Figure 4.10: Evolution of NRMSE_V as samples are sequentially added to the training set by CVVAS with and without active SM selection

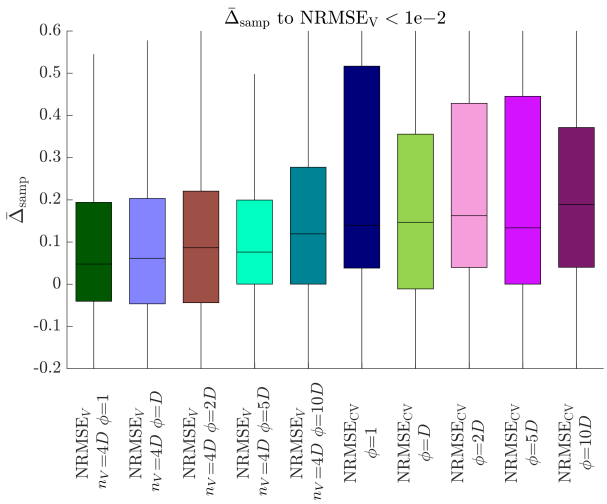
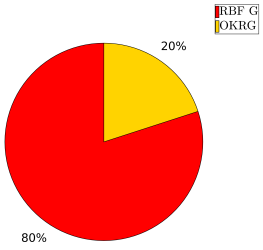


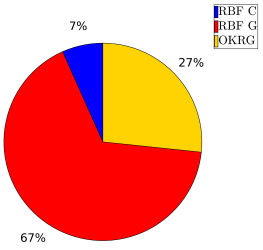
Figure 4.11: Effect of SM selection frequency on $\bar{\Delta}_{\text{samp}}$ to reach $\text{NRMSE}_{\text{V}} < 10^{-2}$ considering all the test functions

definite trend that would make possible to formalize a guideline. Similarly to the effect of the number of validation points, these results indicate the need for a deeper study of the dynamics between the sampling strategy and the SM selection process to define guidelines for a finer tuning of these parameters. Such analysis is out of the scope of this dissertation but is recommended for a future study.

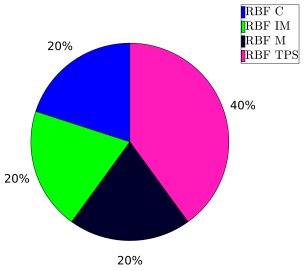
A question that may arise at this point is: does the last SM selected in the active SM selection process correspond to the best SM identified by running all the CVVAS-SM configurations independently? This question is addressed in figure 4.12 which shows two sets of pie charts for 4 different test functions. The charts on the left indicate the percentage of repetitions for which a SM formulation reaches $\text{NRMSE}_{\text{V}} < 10^{-2}$ with the fewest number of samples when no active SM selection is used, while the charts on the right represent the percentage of times a particular SM formulation is the final one selected by the active SM selection when the accuracy reaches $\text{NRMSE}_{\text{V}} < 10^{-2}$. As evident in the figure, the left and right charts look very similar in terms of both the selected SM formulations and the percentages, meaning that the final SM formulation chosen by the active selection is –with high probability– the one which can be identified by running all the CVVAS-SM configurations independently and selecting the one with the best performance.



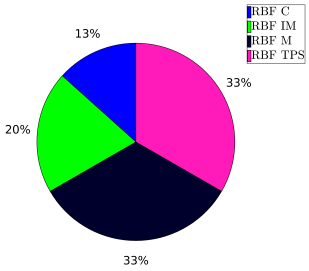
(a) f2DBranin no selection



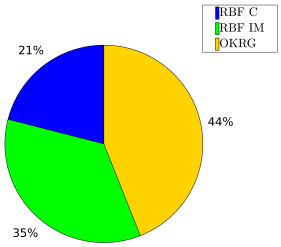
(b) f2DBranin with selection



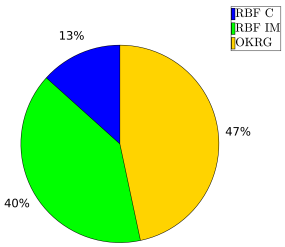
(c) f2DLNA no selection



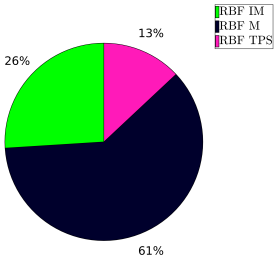
(d) f2DLNA with selection



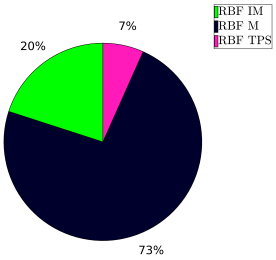
(e) f5DCantilever no selection



(f) f5DCantilever with selection



(g) f5DLNA no selection



(h) f5DLNA with selection

Figure 4.12: Percentage of the 15 sampling repetitions for four test functions. (a,c,e,g) without active SM selection when a specific SM reaches $\mathrm{NRMSE_V} < 10^{-2}$ with fewer samples than the other SMs. (b,d,f,h) with active SM selection ($\mathrm{NRMSE_V}$ as selection criterion and $4D$ validation points) when a specific SM is the chosen one at the moment that $\mathrm{NRMSE_V} < 10^{-2}$

4.4 Key Outcomes of Model Dependence Study

The first objective of this study was to investigate to what extent the appropriateness of the SM formulation selection can affect the performance of a surrogate modeling process when a model dependent sampling technique is adopted. Next, the performance of the same sampling technique was tested within an architecture which includes an active SM selection. Specifically, a strategy based on modified leave-one-out cross validation variance ($\tilde{\sigma}^2_{\mathrm{LOO-CV}}$) was used to sample the design space of 11 example functions, and the sampling performance was assessed in terms of number of samples required to obtain a SM with a certain accuracy level. The first set of results (section 4.3.2) clearly indicates how the sampling performance enhancement expected by the use of an adaptive sequential technique may vanish if an inappropriate SM is chosen at the beginning of the process. Additionally, even the SM leading to the most robust sampling performance shows a high variability in the number of required samples when used on different test functions.

These results motivate the second part of this study where an active SM selection is coupled with the CVVAS sampling strategy. This architecture actively chooses the SM formulation to use in the sampling phase as the one that better represents the available training dataset. Results (section 4.3.3) show how the addition of the active SM selection phase in the surrogate modeling architecture drastically reduces the performance variability observed when the SM formulation is selected a priori. This aspect relieves the user from selecting the SM formulation at the beginning of the process when there is usually limited knowledge about the function to model, resulting in direct benefit to the performance robustness of the overall surrogate modeling process.

The influence of both the number of validation points used in the SM selection and the SM update frequency has also been investigated. The results of this preliminary analysis showed how these two parameters seem to have a negligible effect on the sampling performance. Nevertheless, the author acknowledges that additional research is necessary to

thoroughly analyze the dynamics and the coupling between the sampling algorithm and the active SM selection. Such a study will improve understanding of the influence of these two parameters and others that were fixed in this work (i.e. the suite of candidate SMs, the strategy and the size of the initial training set, and the GEEs used in the SM selection) on the overall process performance. In particular, the choice of the initial training set and of candidate SMs are well known challenges in surrogate modeling for engineering applications; future research is needed to assess their impact on the sampling performance and to more completely characterize the proposed architecture.

CHAPTER 5
SEQUENTIAL ADAPTIVE SAMPLING BASED ON LOCAL LINEAR MODELS: NEAREST NEIGHBORS ADAPTIVE SAMPLING

The present chapter describes the formulation of the proposed model independent sequential adaptive strategy (MISAS) technique named Nearest Neighbors Adaptive Sampling (NNAS). NNAS has been conceived and developed with the intention of fulfilling all the sampling strategy requirements for engineering applications described in section 1.6.

Like all MISASs, the NNAS algorithm decouples the sampling phase from the SM selection and training (figure 2.13b). As described in section 2.3.4, this architecture is advantageous compared to model dependent algorithms (MDSASs) because it reduces computational time and removes bias which may be introduced by the user's a priori choice of the SM functional form. Similarly to the majority of sequential adaptive algorithms (of both MDSAS and MISAS types), the NNAS sampling phase is guided by a refinement (R) and an exploration (E) metric: the former helps to identify the regions in which more samples are needed to achieve a better response representation, and the latter is responsible for spreading the samples across the entire design space.

The basic NNAS (NNAS-B) formulation (section 5.2) introduces the Non Linearity Index (NLI) as refinement metric (section 5.2.1) that is then coupled with a measure of the euclidean distance between samples as exploration metric (section 5.2.2). As described in algorithm 1 and shown in section 5.7, the basic R and E formulations lead to a sampling algorithm that is simple to implement and has a quasilinear complexity ($\mathcal{O}(n_T \log(n_T))$) which is significantly lower than complexity of other state-of-the-art sequential adaptive strategies, e.g. $\mathcal{O}(n_T^4)$ and $\mathcal{O}(n_T^2)$ for CVVAS and LOLA Voronoi (LOLA-V) algorithms, respectively. The drawback of using these simple R and E formulations is a decrease in sample distribution representation quality which however results to be negligible in most

of the tested functions (chapter 6). The objective of introducing directional sampling to enhance the representation quality of the sample distributions obtained via NNAS-B leads a second version of NNAS, named directional NNAS (NNAS-D). Despite new refinement and exploration metric formulations, NNAS-D retains the quasilinear complexity of the overall algorithm, but it requires some additional computational time for the approximate Monte-Carlo Voronoi tessellation involved in the exploration and directional sampling. Because the algorithm computational time is directly related with t_{sampl}, the choice between NNAS-B and NNAS-D should be guided case-by-case by the effect of t_{sampl} on the overall surrogate modeling time (τ effect in eq.(1.5))

Similarly to optimization algorithms ([21, 49]), a good sampling technique must balance the influence of the exploration and refinement metrics to avoid excessive sample clustering and to assure that samples are sufficiently spread across the entire design space. Such a balance should lead to a finer sampling in highly non-linear regions while capturing response features throughout the entire domain. To the best of author's knowledge, all current adaptive sampling techniques achieve this exploration-refinement balance either by forming a convex sum of R and E (as in LOLA-Voronoi ([20])) or by combining R with a distance penalty function (as in CV ([31, 47]) or Lipschitz ([64]) techniques). The NNAS technique presented in this dissertation instead approaches the exploration-refinement balance by adopting a different form of multiobjective optimization approach to maximize both metrics, and therefore to solve the problem in (2.1). First, a Pareto-ranking procedure is used to identify the design space regions that are equally "optimal" for the maximization of both the exploration and refinement metrics. Then, a target exploration-refinement balance is obtained by choosing the region around which the next sample will be placed by means of a stochastic selection criterion (section 5.1).

The first part of this chapter describes the Pareto-ranking based stochastic selection criterion that is used by both NNAS-B and NNAS-D to achieve a target exploration-refinement balance. Sections 5.2 and 5.3 present the NNAS-B and NNAS-D formulations, respec-

tively, and they are followed by the description of the algorithm modifications required to avoid the influence of critical error in the solver, and to use NNAS in multi-response and batch-mode applications. Finally, section 5.7 estimates the computational complexity of both NNAS formulations.

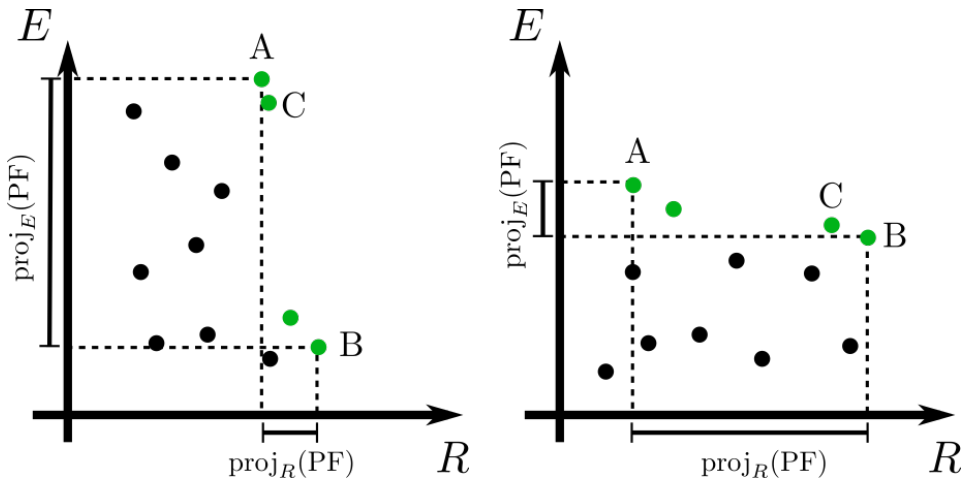
5.1 Refinement-Exploration Balance: Pareto-ranking Based Selection

The refinement and exploration metrics are used in both NNAS formulations to select a sample in D_T ($\mathbf{x}_T^*$) as identifier of the region in which the next sample ($\mathbf{x}_{\text{next}}$) should be placed. Ideally, $\mathbf{x}_{\text{next}}$ should be located in a region around an $\mathbf{x}_T$ which maximizes both R and E, meaning that $\mathbf{x}_T^*$ is one solution of the following multiobjective optimization problem (previously introduced in section 2.1.1):

$$\begin{aligned} &\underset{i}{\text{Maximize}} \quad [R_{T,i}, E_{T,i}] \\ &\text{subject to} \quad 1 \leq i \leq n_T \end{aligned} \tag{5.1}$$

where $R_{T,i}$ and $E_{T,i}$ are the refinement and exploration metrics evaluated at $\mathbf{x}_{T,i}$, respectively. The Pareto frontier (PF) representing the solution of eq.(5.1) (PF^*) is obtained by the application of a Pareto ranking procedure on the R and E values for all the samples $\mathbf{x}_{T,i}$ in the design space.

PFs usually contain more than one point, and therefore a criterion is needed to select a single $\mathbf{x}_T^*$ from PF^*. Based on the definition of Pareto dominance ([14]), the PF^* points are equally "optimal" solutions of the optimization problem in eq.(5.1), and therefore a possible legitimate approach is to randomly draw $\mathbf{x}_T^*$ from the PF^* points. However, this randomly selected $\mathbf{x}_T^*$ may not be "optimal", as can be seen in figure 5.1. Consider for example the situation in figure 5.1a, where PF^* spreads over a much smaller range in R than E. If R and E are properly normalized to have comparable metrics, the PF^* points in figure 5.1a are almost equivalent with respect to the maximization of R. Assuming



(a) High E (b) High R

Figure 5.1: Example of $\max(R)$-$\max(E)$ Pareto frontiers

that R and E are equally important, point A in figure 5.1a is one of the best solution of the optimization problem in eq.(5.1). A similar conclusion can be drawn for the case illustrated in figure 5.1b, where instead point B should be chosen as solution of eq.(5.1). Additionally, it is possible to realize from the high level definition of exploration and refinement metrics that PF^* naturally evolves toward the center of the R-E axes as more points are added to D_T. Indeed, anytime a new sample $\mathbf{x}_{\text{next}}$ is added to D_T, both E and R around $\mathbf{x}_{\text{next}}$ are expected to decrease because of the reduced distance between samples and the improved response representation in the surrounding region (assuming that a new highly nonlinear region is not detected at $\mathbf{x}_{\text{next}}$).

Hence, it is reasonable to presume that a good refinement-exploration balance is achieved when PF^* evolves toward the center of the R-E axes (which naturally happens) keeping the lengths of its projections on the R and E axes as equal as possible. More formally

$$\rho = \frac{\mathrm{proj}_E(\mathrm{PF}^*)}{\mathrm{proj}_R(\mathrm{PF}^*)} \approx \hat{\rho} = 1 \tag{5.2}$$

where $\mathrm{proj}_E(\mathrm{PF}^*)$ and $\mathrm{proj}_R(\mathrm{PF}^*)$ are the lengths of PF^* projections on the E and R axes (figure 5.1), respectively, and $\hat{\rho}$ is the target value for the ratio ρ. Such an evolution of PF^*

is achieved in NNAS by a stochastic selection of $\mathbf{x}_T^*$ based on a probability distribution function (ϕ_{PF^*}) which is continuously adapted to guide the sampling process toward $\rho \approx \hat{\rho} = 1$: different target values for $\hat{\rho}$ can be adopted to obtain behavior more heavily weighted towards exploration or refinement.

The chosen $\phi_{\mathrm{PF}^*}(t)$ is a trapezoidal distribution with bases proportional to $\mathrm{proj}_E(\mathrm{PF}^*)$ and $\mathrm{proj}_R(\mathrm{PF}^*)$; the variable t is the linear coordinate along the line connecting the two extreme points of the PF^*, as illustrated in figure 5.2. The mathematical definitions of $\phi_{\mathrm{PF}^*}(t)$ (eq.(5.3)) and its cumulative density function $\Phi_{\mathrm{PF}^*}(t)$ (eq.(5.4)) are:

$$\phi_{\mathrm{PF}^*}(t) = mt + q \tag{5.3}$$

$$\Phi_{\mathrm{PF}^*}(t) = \frac{1}{2}mt^2 + qt \tag{5.4}$$

where,

$$m = \frac{2}{t_{\max}}\left(\frac{1-\rho/\hat{\rho}}{\rho/\hat{\rho}+1}\right) \tag{5.5}$$

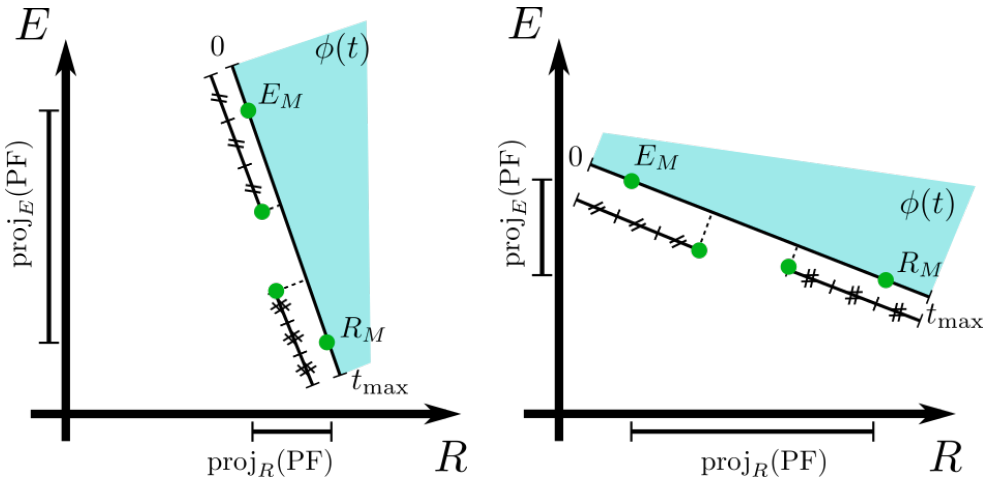
$$q = \frac{2\rho/\hat{\rho}}{(\rho/\hat{\rho}+1)t_{\max}} \tag{5.6}$$

Φ_{PF^*} (eq.(5.4)) is invertible, and therefore the inverse cumulative density function method ([104]) can be used to draw a random value t_r from the ϕ_{PF^*}:

$$t_r(s) = \Phi^{-1}_{\mathrm{PF}^*}(s) \tag{5.7}$$

$$= \frac{-q + \sqrt{q^2 + 2sm}}{m} \tag{5.8}$$

where s is a random value between 0 and 1 drawn from a uniform distribution. Finally, $\mathbf{x}_T^*$ is chosen as the PF^* point with the t coordinate closer to the random value t_r drawn using



(a) High E (b) High R

Figure 5.2: Example of trapezoidal probability distributions for the stochastic Pareto-ranking-based selection criterion

eq.(5.8):

$$\mathbf{x}_T^* = \mathbf{x}_{T,j^*} \tag{5.9}$$

where,

$$j^* = \underset{j}{\text{argmin}}(|t_j - t_r|) \quad \text{with} \quad j = 1 \ldots n_{\text{PF}^*} \tag{5.10}$$

and n_{PF^*} is the number of points in PF^*.

As it is possible to see from figure 5.2, the extreme values of the coordinate t ($t = 0$ and $t = t_{\max}$) are located outside the limits defined by the projection of PF^* on the t-axis. Using the notation E_M and R_M for the points on PF^* which maximize E and R respectively, $t = 0$ is located at t equal to $t(E_M)$ minus half the distance from E_M and its neighbor in the t-axis. Similarly, $t = t_{\max}$ is located at t equal to $t(R_M)$ plus half the distance from R_M and its neighbor in the t-axis. This is done to assign the PF^* extreme points a fair chance of being selected as $\mathbf{x}_T^*$. An example of PF^* evolution as training points are added to D_T is illustrated in figure 5.3.

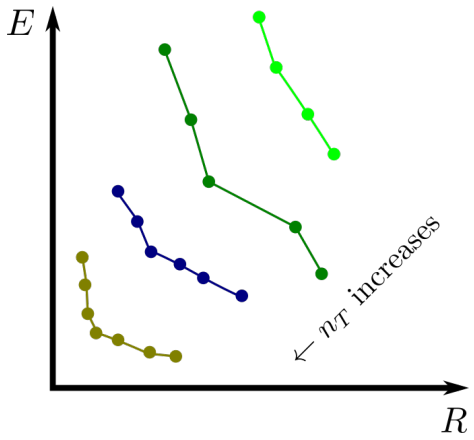


Figure 5.3: Example of Pareto frontier evolution as samples are added to D_T

The stochastic selection of $\mathbf{x}_T^*$ described above has been introduced in NNAS to limit inductive bias by assigning to every PF^* point a probability – even if very small – to be selected. Additionally, the approach gives the PF^* points which are closely spaced on the t-axis a similar likelihood to be drawn as $\mathbf{x}_T^*$ (i.e. points A and C in figure 5.1a and points B and C in figure 5.1b). This behavior is appropriate because R is based on an approximate estimation of the response nonlinearity and therefore there is not strong motivation to prefer one point compared to another if they are close on the PF^*.

5.2 Basic Nearest Neighbors Adaptive Sampling

The basic NNAS (NNAS-B) is the simplest implementation of NNAS algorithm that uses local linear model to estimate the response non-linearity around each existing training sample. The following sections provide the details about the NNAS-B refinement and exploration metrics (Sections 5.2.1 and 5.2.2), the approach used to select the next sample $\mathbf{x}_{\mathrm{next}}$ once $\mathbf{x}_T^*$ has been selected by the Pareto-ranking based approach (section 5.2.3), and finally a detailed description of the overall NNAS-B algorithm and implementation (section 5.2.4).

5.2.1 Refinement Metric: Non Linearity Index

The proposed Non Linearity Index (NLI) refinement metric has been devised to produce a finer sample distribution in design space regions where the response shows a higher departure from linearity. NLI estimates the level of nonlinearity in the neighborhood of a training sample without need for a global SM – a key characteristic that makes NNAS-B a *model independent* sampling strategy.

NLI at the ith training sample $\mathbf{x}_{T,i}$ (NLI_i) is defined as the difference between the true response $y_{T,i}$ and an approximation of f in the neighborhood of $\mathbf{x}_{T,i}$. This approximation is represented by the D-dimensional hyperplane ($\Pi^{(i)}$) obtained by a weighted least squares regression (WLSR) on the $D + k$ nearest neighbors of $\mathbf{x}_{T,i}$ which constitute its neighborhood $N^{(i)}$ (figure 5.4). More formally,

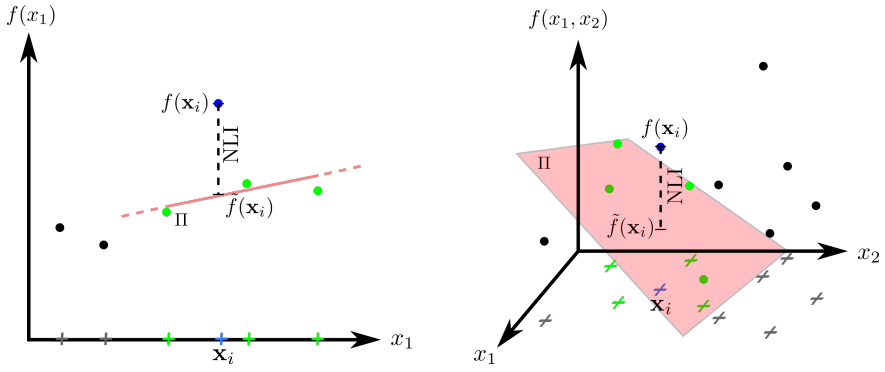
$$\mathrm{NLI}_i = \left| y_{T,i} - \Pi^{(i)}(\mathbf{x}_{T,i}) \right| \tag{5.11}$$

where $\Pi^{(i)}$ is

$$\Pi^{(i)}(\mathbf{x}) = c_0^{(i)} + \sum_{q=1}^{q=D} c_q^{(i)} x_q \tag{5.12}$$

and $c_q^{(i)}$ are the coefficients returned by the WLSR. Based on eq.(5.12), the WLSR linear system is

$$\begin{bmatrix} w^{(i,1)} & & \\ & \ddots & \\ & & w^{(i,D+k)} \end{bmatrix} \begin{bmatrix} 1 & x_1^{(i,1)} & \cdots & x_D^{(i,1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(i,D+k)} & \cdots & x_D^{(i,D+k)} \end{bmatrix} \begin{bmatrix} c_0^{(i)} \\ \vdots \\ c_D^{(i)} \end{bmatrix} = \begin{bmatrix} w^{(i,1)} & & \\ & \ddots & \\ & & w^{(i,D+k)} \end{bmatrix} \begin{bmatrix} y_T^{(i,1)} \\ \vdots \\ y_T^{(i,D+k)} \end{bmatrix} \tag{5.13}$$



(a) 1D (b) 2D

Figure 5.4: Graphical representation of NLI evaluation. Crosses denote the $\mathbf{X}_T$ location in the design space, while dots represent the $[\mathbf{X}_T, \mathbf{y}_T]$ pairs. The $\mathbf{x}_{T,i}$ sample where NLI has to be computed is displayed in blue, while the points in its neighborhood $N^{(i)}$ are in green.

or more compactly

$$\mathbf{W}^{(i)} \left[\mathbf{1}, \mathbf{X}_T^{(i)}\right] \mathbf{c}^{(i)} = \mathbf{W}^{(i)} \mathbf{y}_T^{(i)} \tag{5.14}$$

where superscript (i, j) indicates "the jth neighbor of the ith training point", and w are the weights of WLSR. Therefore, the notation $x_p^{(i,j)}$ indicates the pth coordinate of the jth neighbor of the ith training point. The weights $w^{(i,j)}$ (eq.(5.15)) are defined as the inverse of the euclidean distance between the jth neighbor and the ith training points around which the nonlinearity of the response is intended to be estimated by solving eq.(5.14):

$$w^{(i,j)} = \frac{1}{|\mathbf{x}_{T,i} - \mathbf{x}_{T,j}|} \tag{5.15}$$

This choice is based on the assumption that closer neighbors better represent the region around $\mathbf{x}_{T,i}$.

A $D + k$ dimensional regression (eq.(5.14)) is used instead of a linear interpolation of the $D + 1$ nearest neighbors because preliminary tests have shown a frequent incidence of singular matrices due to point alignment. A value of $k = 2$ has led to a complete removal

of the singularity issues on all the test cases that we have examined. Nevertheless, the value of k can be increased case-by-case if the matrix inversion returns a singularity warning, as described in the algorithm implementation section (section 5.2.4).

Finally, the refinement metric at $\mathbf{x}_{T,i}$ ($R_{T,i}$) is defined as the NLI_i normalized by the range of the collected responses:

$$R_{T,i} = \frac{\mathrm{NLI}_i}{\max(\mathbf{y}_T) - \min(\mathbf{y}_T)} \tag{5.16}$$

5.2.2 Exploration Metric: Mean Neighborhood Distance

Similarly to other sampling techniques ([47, 53, 64]), the exploration metric E used in NNAS-B is based on the euclidean distance between samples. In particular, E at $\mathbf{x}_{T,i}$ ($E_{T,i}$) is defined as half of the mean distance between $\mathbf{x}_{T,i}$ and its neighbors in $N^{(i)}$ normalized by the diagonal of the design space:

$$E_{T,i} = \frac{\frac{0.5}{D+k}\sum_{j=1}^{D+k}\left|\mathbf{x}_{T,i} - \mathbf{x}^{(i,j)}\right|}{\sqrt{\sum_{d=1}^{D}(\mathrm{UB}_d - \mathrm{LB}_d)^2}} \tag{5.17}$$

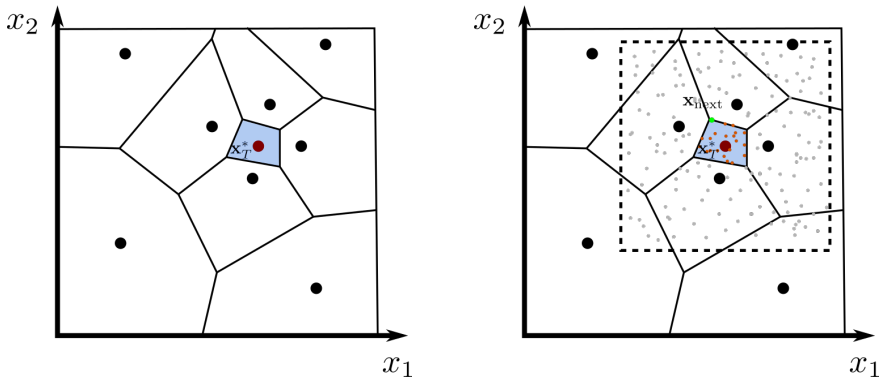
where UB_d and LB_d represent the upper and lower bounds of the dth dimension (if all the design variables are normalized within a 0-1 range, the denominator of (5.17) is obviously $\sqrt{D}$). $E_{T,i}$ can be considered as the normalized radius of the hypersphere representing – with high level of approximation – the locus of points in the design space that are closer to $\mathbf{x}_{T,i}$ than other samples (this is also the motivation for the 0.5 factor in (5.17)). A more rigorous approach would require a design space Voronoi tessellation ([5]) based on $\mathbf{X}_T$, but such an approach scales poorly with increasing dimension, D (at worst $\Omega(n^{\frac{D}{2}})$ [51]). As shown in the results (chapter 6), the E formulation of eq.(5.17) is able to achieve effective design space exploration with a negligible impact in algorithm complexity.

5.2.3 Random Search: Identification of the Next Sample

Once R and E have been computed for every sample in D_T, the Pareto-ranking based procedure described in section 5.1 selects $\mathbf{x}_T^*$ which is the indicator of the region where $\mathbf{x}_{\text{next}}$ should be placed to improve the response representation by keeping a target exploration-refinement balance. At this point, two questions are still unanswered: how is the region surrounding $\mathbf{x}_T^*$ defined? And how is $\mathbf{x}_{\text{next}}$ selected within it?

Assume for now that the design space can be partitioned by the Voronoi tessellation ([5]), where every location $\mathbf{x}$ in the domain is "assigned" to the closest $\mathbf{x}_T$: the Voronoi cell (VC) of $\mathbf{x}_{T,i}$ (VC_i) is the locus of domain locations that have $\mathbf{x}_{T,i}$ as closest point. VC^* is therefore a reasonable representation of $\mathbf{x}_T^*$ surrounding as illustrated in the 2-dimensional Voronoi tessellation example in figure 5.5a. Ideally, we would like to select $\mathbf{x}_{\text{next}}$ to maximize a "local" version of the exploration and refinement metrics – $\tilde{E}$ and $\tilde{R}$ respectively – within the limits of VC^*. Since NLI is a point-based metric computed at training points and there is not a simple and elegant approach to extrapolate NLI at every design space location without the use of an SM, we assume that – at any point within VC^* – $\tilde{E}$ is equal to the distance from $\mathbf{x}_T^*$, and $\tilde{R}$ is equal to R at $\mathbf{x}_T^*$ (R_{T,i^*}). With this assumption, the multiobjective optimization of maximizing both $\tilde{E}$ and $\tilde{R}$ turns into a single objective problem where the only objective is the maximization of $\tilde{E}$, and the solution is the VC^* corner located at the highest distance from $\mathbf{x}_T^*$ (figure 5.5a).

Even though it is possible to exactly compute the Voronoi corners, the Voronoi tessellation algorithm is extremely computationally expensive (at worst $\Omega(n^{\frac{D}{2}})$ [51]) making this approach to find $\mathbf{x}_{\text{next}}$ infeasible for most high dimensional engineering applications. Therefore, NNAS-B adopts an approximate strategy that creates a large number (proportional to problem dimensionality ($n_R \propto D$)) of random points $\mathbf{x}_R$ in an extended surrounding of $\mathbf{x}_T^*$, and then it selects $\mathbf{x}_{\text{next}}$ as the random point that is within VC^* (i.e. among the random points that have $\mathbf{x}_T^*$ as the closest sample) and with the greatest distance from $\mathbf{x}_T^*$. The extended surrounding is defined as the hypercube centered at $\mathbf{x}_T^*$ with an edge equal



(a) (b)

Figure 5.5: Example of $2D$ Voronoi tessellation (5.5a), and random search of $\mathbf{x}_{\text{next}}$ (green dot) in the surrounding (blue Voronoi cell) of $\mathbf{x}_T^*$ (red dot) (5.5b). The region limited by the dashed line represents the hypercube, and the gray dots the random candidates points

to double the max distance from $\mathbf{x}_T^*$ to any of its neighbors in $N^{(i^*)}$ (figure 5.5b). More formally, the location of $\mathbf{x}_{\text{next}}$ is the solution of the following maximization problem:

$$\mathbf{x}_{T,\text{next}} = \mathbf{x}_{R,i_{\text{next}}} \tag{5.18}$$

where,

$$i_{\text{next}} = \underset{i}{\text{argmax}}(|\mathbf{x}_{R,i} - \mathbf{x}_T^*|) \quad \text{with} \quad \mathbf{x}_R \in \text{VC}^* \tag{5.19}$$

5.2.4 Basic NNAS Algorithm

In summary, the overall NNAS-B algorithm can be qualitatively described as follows:

1. Initialize D_T with a set of initial training points ($n_{T,\text{in}} > D + k$) using standard space-filling algorithms.

2. Compute NLI for each training point in D_T (eq.(5.11)).

3. Compute E (eq.(5.17)) and R (eq.(5.16)) at each training point.

4. Identify the Pareto frontier PF^* for the $\max(E)$-$\max(R)$ problem .

5. Randomly select $\mathbf{x}_T^*$ in PF^* based on the probability distribution function ϕ_{PF^*} (Eq.5.5) using the inverse cumulative distribution function method (eq.(5.8)).

6. Create n_R random points spread over a hypercube centered in $\mathbf{x}_T^*$ with edge length equal to double the maximum distance between $\mathbf{x}_T^*$ and its neighbors in $N^{(i^*)}$.

7. Select the next sample location $\mathbf{x}_{\mathrm{next}}$ as the random point within the Voronoi cell of $\mathbf{x}_T^*$ that has the maximum distance from $\mathbf{x}_T^*$.

8. Evaluate $f(\mathbf{x}_{\mathrm{next}})$.

9. Add $\{\mathbf{x}_{\mathrm{next}}, f(\mathbf{x}_{\mathrm{next}})\}$ to the training set D_T.

10. Repeat from step 2 until a stopping criterion is met.

Regarding the stopping criteria, possible options can be based on the current number of samples, the maximum value of the refinement metric R which is a rough metric of the representation quality of the training set D_T, or the maximum value of the exploration metric E which is an index of the spreading distance between samples.

NNAS-B can be efficiently implemented by leveraging the k-nearest-neighbors search (kNN-S) algorithm and by computing R and E only at the neighborhoods affected by the addition of the new samples. Even if some parts of the algorithm can be easily parallelized, the scheme presented and discussed in this section (algorithm 1) is a serial implementation of NNAS-B which has been shown to be extremely fast even for high dimensional applications. The NNAS-B algorithm inputs are the black-box or the code used to evaluate the unknown function f, a value k which specifies the additional neighbors considered to avoid singularity in the NLI evaluation, and a stopping criterion SC; the output of the algorithm is obviously the training dataset D_T. As previously introduced, a value of $k = 2$ was sufficient to avoid matrix singularity issues in all the completed tests. If not adequate,

the algorithm implementation can easily incorporate a temporary increase of k in case of singularity warnings during the NLI evaluation (row 16).

Considering algorithm 1, rows 1 to 5 perform the preliminary operations of variable initializations:

- Step 1 creates an initial set of $n_{T,\text{in}}$ training points ($\mathbf{X}_{T,\text{in}}$) by a space-filling algorithm like LHS. Since NNAS-B uses $D + k$ neighbors, $n_{T,\text{in}}$ must be greater than $D + k$.
- Step 2 evaluates the training response values $\mathbf{y}_{T,\text{in}} = f(\mathbf{X}_{T,\text{in}})$
- Step 3 combines $\mathbf{X}_{T,\text{in}}$ and $\mathbf{y}_{T,\text{in}}$ to create the initial training dataset.
- Step 4 initializes the N_{st} matrix where the algorithm will store the indeces of the neighbors used to compute the NLI at each training point.
- Step 5 initializes the NLI vector where the algorithm will store the values of the computed NLI.

N_{st} stores the indices of the $D+k$ nearest neighbors of each samples in D_T (row 17), and it is used in row 15 to prevent the computation of NLI for points whose neighborhoods have been unaffected by the addition of the new sample at the end of the previous iteration (row 26).

The while loop within rows 6 to 27 is the core of the NNAS-B algorithm, and it is entered until the stopping criterion (SC) is not satisfied. More specifically:

- Row 7 updates the sample-to-sample distance matrix Δ by computing the distances between the training point added at the end of the previous iteration and the samples already in D_T (at the first iteration, Δ must be initialized with the distances between points in the initial training set).
- The for loop in rows 8 to 10 identifies the $D + k$ neighbors of the new training point by using the efficient kNN-S algorithm, and it stores the neighbors indices in the

matrix N_{st}. The kNN-S takes as input the distance matrix Δ previously updated to reduce the algorithm computational cost, as described in section 5.7.

- The for loop in rows 11 to 13 checks if the addition of the new training sample at the end of the previous iteration has affected the neighborhoods of the existing training samples. This operation is efficiently implemented by using Δ and the neighbor indices stored in N_{st}, without the need of running a completely new kNN-S.

- The third for loop (rows 14 to 19) is responsible for the update of the NLI vector. As it possible to see, the actual NLI evaluation (row 16) is done only if the updated neighborhood ($N^{(\mathrm{i})}$) differs from what was computed at the previous iteration ($N_{\mathrm{st}}^{(\mathrm{i})}$) (condition of the inner if statement (row 15)). The k value can be temporarily increased depending on the condition number of eq.(5.13) or if the solution of eq.(5.13) faces some singularity issues.

- Rows 20 and 21 use NLI, Δ and D_T to compute the refinement and exploration metrics.

- Rows from 22 to 25 identify the new training sample by using the stochastic Pareto-ranking selection and the approximate random search within the chosen Voronoi cell.

- Rows 26 appends the new training information to the training dataset D_T.

5.3 Nearest Neighbors Adaptive Sampling with Directional Sampling

As highlighted by the results (chapter 6), the simple NNAS-B refinement metric formulation may lead in same cases to a lower sample distribution efficiency if compared with techniques that have more advanced approaches to define neighborhoods or to compute the refinement metric. On the other hand, those advanced sampling formulations have a higher computational cost (e.g. $\mathcal{O}(n_T^2)$ and $\mathcal{O}(n_T^4)$ for CVVAS and LOLA-V, respectively) in

input : $f(\mathbf{x})$, k, SC
output: D_T

Create $n_{T,\text{in}}$ initial samples ($X_{T,\text{in}}$) using a space-filling algorithm ($n_{T,\text{in}} > D + k$);
Evalute $\mathbf{y}_{T,\text{in}} = f(X_{T,\text{in}})$;
Create the initial $D_T = \{X_{T,\text{in}}, \mathbf{y}_{T,\text{in}}\}$;
Initialize N_{st} as a zeros matrix of size $n_{T,\text{in}} \times D + k$;
Initialize NLI as a zeros vector of size $n_{T,\text{in}}$;
while *SC **is** false* **do**
    $\Delta \leftarrow$ update `distance(`X_T`)`;
    for $i = n_T - n_{\text{add}} + 1 : n_T$ **do**
        $N^{(i)} \leftarrow$ `knnsearch(`i,Δ`)`;
    end
    for $i = 1 : n_T - n_{\text{add}}$ **do**
        $N^{(i)} \leftarrow$ `neighborhoodUpdate(`i,N_{st},Δ`)`;
    end
    for $i = 1 : n_T$ **do**
        if $N^{(i)} \neq N_{\text{st}}^{(i)}$ **then**
            $\text{NLI}_i \leftarrow$ `NLIeval(`$N^{(i)}, X_T(N^{(i)}), \mathbf{y}_T(N^{(i)})$`)`;
            $N_{\text{st}}^{(i)} \leftarrow N^{(i)}$;
        end
    end
    $R \leftarrow (\text{NLI}, D_T)$;
    $E \leftarrow (\Delta)$;
    $\text{PF}^* \leftarrow$ `ParetoRanking(`R,E`)`;
    $\mathbf{x}_T^* \leftarrow$ `invCumulative(`PF^*`)`;
    $\mathbf{x}_{T,\text{new}} \leftarrow$ `randomSearch(`$\mathbf{x}_T^*$`)`;
    $y_{T,\text{new}} \leftarrow$ `eval(`$f(\mathbf{x}_{T,\text{new}})$`)` ;
    $D_T \leftarrow$ `append(`$\mathbf{x}_{T,\text{new}}$,$y_{T,\text{new}}$`)` ;
end

Algorithm 1: NNAS-B algorithm

comparison to the quasilinear complexity of NNAS-B. Even though this sampling performance degradation occurs only in a limited number of cases (chapter 6), directional NNAS (NNAS-D) is conceived in the attempt to mitigate or possibly solve this side effect.

The following sections provide the details about the NNAS-D refinement and exploration metrics (Sections 5.3.1 and 5.3.2), the directional sampling approach used to select the next sample $\mathbf{x}_{\text{next}}$ once $\mathbf{x}_T^*$ has been chosen by the Pareto-ranking based criterion (section 5.3.3), and the overall NNAS-D algorithm and implementation (section 5.3.4).

5.3.1 Refinement Metric: Local Root Mean Squared Error

Similarly to NNAS-B, the NNAS-D refinement metric has been devised to produce finer sample distribution in design regions with higher response departure from linearity. Those regions are identified by using hyperplanes created in the neighborhood of each training point, thereby keeping NNAS-D model independent. Specifically, R at the ith training point $\mathbf{x}_{T,i}$ is the RMSE of the hyperplane $\Pi^{(i)}$ that passes through $\mathbf{x}_{T,i}$ and is created via weighted least squares regression (WLSR) of the n_N nearest neighbors of $\mathbf{x}_{T,i}$ that constitute its neighborhood $N^{(i)}$ (figure 5.6). The number of neighbors to consider (n_N) is an user choice, but the analysis conducted and reported in chapter 6 shows that $n_N = 2D$ is an appropriate value (where D is the design space dimensionality). The approach of forcing the hyperplane to pass through the training point where R has to be computed is borrowed from LOLA Voronoi (LOLA-V) technique [20], which however has a more complex approach to define the neighborhood $N^{(i)}$.

Formally, R for NNAS-D is computed as:

$$\text{RMSE}_i = \sqrt{\frac{1}{n_N}\sum_{j=1}^{n_N}\left(y_T^{(i,j)} - \Pi^{(i)}(\mathbf{x}_T^{(i,j)})\right)^2} \tag{5.20}$$

$$= \sqrt{\frac{1}{n_N}\sum_{j=1}^{n_N}(\text{PE}_j^{(i)})^2} \tag{5.21}$$

where $\mathrm{PE}_j^{(i)}$ is the prediction error at the jth neighbor of ith training point, $\Pi^{(i)}$ is

$$\Pi^{(i)}(\mathbf{x}) = c_0^{(i)} + \sum_{q=1}^{q=D} c_q^{(i)} x_q \tag{5.22}$$

and $c_q^{(i)}$ are the coefficients returned by the WLSR. Based on eq.(5.22), the WLSR linear system is

$$\begin{bmatrix} w^{(i,1)} & & \\ & \ddots & \\ & & w^{(i,n_N)} \end{bmatrix} \begin{bmatrix} x_1^{(i,1)} - x_{1;T,i} & \cdots & x_D^{(i,1)} - x_{D;T,i} \\ \vdots & \vdots & \vdots \\ x_1^{(i,n_N)} - x_{1;T,i} & \cdots & x_D^{(i,D+k)} - x_{D;T,i} \end{bmatrix} \begin{bmatrix} c_1^{(i)} \\ \vdots \\ c_D^{(i)} \end{bmatrix} = \begin{bmatrix} w^{(i,1)} & & \\ & \ddots & \\ & & w^{(i,n_N)} \end{bmatrix} \begin{bmatrix} y_T^{(i,1)} - y_{T,i} \\ \vdots \\ y_T^{(i,n_N)} - y_{T,i} \end{bmatrix} \tag{5.23}$$

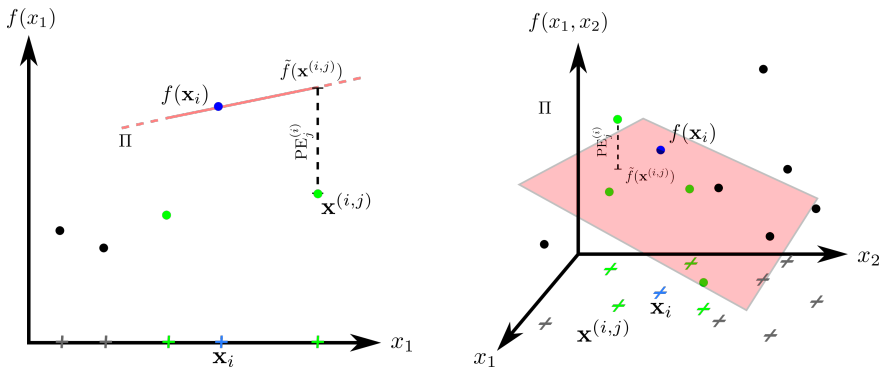
and

$$c_0^{(i)} = y_{T,i} - \sum_{q=1}^{D} c_q^{(i)} x_{q;T,i} \tag{5.24}$$

where the notation is the same as that in eq.(5.13), $x_{k;T,i}$ indicates the kth coordinate of the ith training point, and the weights are defined as in eq.(5.15).

Finally, the NNAS-D refinement metric at $\mathbf{x}_{T,i}$ ($R_{T,i}$) is defined as the RMSE_i normalized by the range of the collected responses:

$$R_{T,i} = \frac{\mathrm{RMSE}_i}{\max(\mathbf{y}_T) - \min(\mathbf{y}_T)} \tag{5.25}$$



(a) 1D (b) 2D

Figure 5.6: Graphical representation of NNAS-D R evaluation. Crosses denote the $\mathbf{X}_T$ location in the design space, while dots represent the $[\mathbf{X}_T, \mathbf{y}_T]$ pairs. The $\mathbf{x}_{T,i}$ sample where R has to be computed is displayed in blue, while the points in its neighborhood $N^{(i)}$ are in green.

5.3.2 Exploration Metric: Equivalent Voronoi Cell Edge Length

As explained in the following section, the introduction of the directional sampling enhances the sampling performance by giving the possibility of local refinement within the Voronoi cell of $\mathbf{x}_T^*$ (VC^*), which is selected via the stochastic Pareto selection. On the other hand, this NNAS-D capability of local refinement requires the use of a more accurate sample clustering estimator in comparison to NNAS-B formulation. Indeed, NNAS-B locates the next sample at the farthest estimated corner of VC^* without considering any local refinement metric; this "only-local-exploration" behavior compensates for the underestimation of the sample clustering in the NNAS-B exploration metric in eq.(5.17).

The NNAS-D exploration metric E at the ith training point $\mathbf{x}_{T,i}$ is the normalized edge length of the hypercube that has the same volume of the $\mathbf{x}_{T,i}$ Voronoi cell ($V_{T,i}$)

$$E_{T,i} = \frac{V_{T,i}^{\frac{1}{D}}}{\sqrt{\sum_{d=1}^{D}(\mathrm{UB}_d - \mathrm{LB}_d)^2}} \tag{5.26}$$

Due to the extremely high computational complexity of Voronoi tessellation algorithm (at worst $\Omega(n^{\frac{D}{2}})$ [51]), $V_{T,i}$ must be estimated via a Monte-Carlo approach with a number of random points proportional to n_T, as suggested in LOLA-V formulation [20].

5.3.3 Directional Sampling: Identification of the Next Sample

Similarly to NNAS-B formulation, NNAS-D requires the computation of R and E for every sample in D_T, and the use of the stochastic Pareto-ranking procedure (section 5.1) to select $\mathbf{x}_T^*$ which identifies the region where $\mathbf{x}_{\text{next}}$ should be placed to improve the response representation by keeping a target exploration-refinement balance.

At this point, two questions motivate the development of a directional sampling formulation:

1. is it possible to devise a strategy that prioritizes some points within the Voronoi cell VC^* with respect to the degree of desired local exploration/refinement?

2. how is the degree of local exploration/refinement within VC^* defined?

A simple and straightforward approach would fit an SM on the local prediction errors (PE) of $\mathbf{x}_T^*$ neighbors, and use a Monte-Carlo approach to identify a point within VC^* that maximizes a weighted sum of the PE estimated via SM and the distance from $\mathbf{x}_T^*$. Unfortunately, such an approach requires the selection of an SM functional form, and therefore it would turn NNAS-D into a model dependent sampling strategy. Therefore, NNAS-D addresses the second question by using a binary rule that determines if pursuing either full local exploration or full local refinement. The rule is based on the t coordinate of the $\mathbf{x}_T^*$ which is the projection of the $[R(\mathbf{x}_T^*), E(\mathbf{x}_T^*)]$ point on the line connecting the extremes of the Pareto frontier representing the solution of eq.(5.1) (figure 5.2). The binary rule is:

$$\begin{cases} \text{exploration}, & \text{if } t^* \leq 0.5 \\ \text{refinement}, & \text{otherwise} \end{cases} \tag{5.27}$$

Regarding the first question, assume that the exact Voronoi tessellation has been computed and the coordinates of Voronoi corners are available. NNAS-D considers VC^* corners as the potential $\mathbf{x}_{\mathrm{next}}$ locations because they minimize the risk of future point alignment and are located at the farthest distance from $\mathbf{x}_T^*$ (figure 5.7a); the final selection of the corner which will become $\mathbf{x}_{\mathrm{next}}$ depends on the outcome of the binary rule in eq.(5.27). In case of "refinement", NNAS-D firstly identifies the $\mathbf{x}_T^*$'s neighbor with the maximum prediction error (PE) among those that are contiguous to VC^*, and then selects as $\mathbf{x}_{\mathrm{next}}$ the corner on the $\mathbf{x}_T^*$-$\mathbf{x}_{\mathrm{max,PE}}$ bisectional hypersurface that has the smallest distance to $\mathbf{x}_T^*$ (figure 5.7b). It is important to underline that the PE values have been already computed for the evaluation of the refinement metric in eq.(5.21). If instead eq.(5.27) returns "exploration", NNAS-D selects as $\mathbf{x}_{\mathrm{next}}$ the VC^* corner which maximizes its distance from $\mathbf{x}_T^*$ (figure 5.7c). The requirement of "Voronoi contiguity" between the considered neighbors and $\mathbf{x}_T^*$ is introduced because it is not guaranteed that the n_N nearest neighbors coincide with the Voronoi neighbors (figure 5.7a).

As already mentioned several times, the Voronoi tessellation has a prohibitive computational complexity with respect to the problem dimensionality (at worst $\Omega(n^{\frac{D}{2}})$ [51]), and therefore an approximate Monte-Carlo approach is used to identify the Voronoi corner regions. The description of the procedure is available in appendix C.

5.3.4 Directional NNAS Algorithm

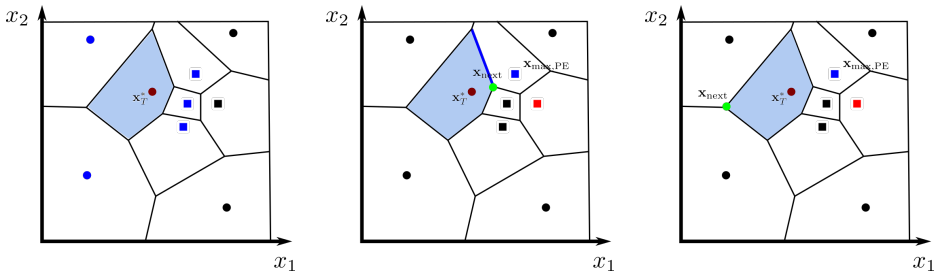
In summary, the overall NNAS-D algorithm can be qualitatively described as follows:

1. Initialize D_T with a set of initial training points ($n_{T,\mathrm{in}} > n_N > D$) using standard space-filling algorithms.

2. Compute the local RMSE for each training point in D_T (eq.(5.20)).

3. Compute E (eq.(5.17)) and R (eq.(5.25)) at each training point.

4. Identify the Pareto frontier PF^* for the $\max(E)$-$\max(R)$ problem .



(a) Contiguous vs Neighbor points

(b) Local refinement

(c) Local exploration

Figure 5.7: (a) shows the difference between the neighbor points considered in the local RMSE evaluation (squares) and the points contiguous to VC^* (blue markers). (b) illustrates all the elements involved in the local refinement: neighbor with maximum PE (red square), contiguous neighbor with maximum PE (blue square), bisectional hypersurface (blue line), and corner selected as $\mathbf{x}_{\text{next}}$ (green dot). (c) shows the local exploration case.

5. Randomly select $\mathbf{x}_T^*$ in PF^* based on the probability distribution function ϕ_{PF^*} (Eq.5.5) using the inverse cumulative distribution function method (eq.(5.8)).

6. Use the Monte-Carlo approximate procedure to identify the corner regions of the $\mathbf{x}_T^*$ Voronoi cell.

7. Select the next sample location $\mathbf{x}_{\text{next}}$ among the corner random points based on the binary t coordinate rule and the prediction error of the neighbors contiguous to $\mathbf{x}_T^*$.

8. Evaluate $f(\mathbf{x}_{\text{next}})$.

9. Add $\{\mathbf{x}_{\text{next}}, f(\mathbf{x}_{\text{next}})\}$ to the training set D_T.

10. Repeat from step 2 until a stopping criterion is met.

As for NNAS-B, possible stopping criteria can be based on the current number of samples, the maximum value of the refinement metric R which is a rough metric of the representation quality of the training set D_T, or the maximum value of the exploration metric E which is an index of the spreading distance between samples.

Similarly to NNAS-B, some parts of NNAS-D algorithm can be easily parallelized, but the scheme presented and discussed in this section (algorithm 2) is a serial implementation

of NNAS-D which has been shown to be extremely fast even for high dimensional applications. The NNAS-D algorithm inputs are the black-box or the code used to evaluate the unknown function f, a value n_N which specifies the number of nearest neighbors considered in the local RMSE evaluation, and a stopping criterion SC; the output of the algorithm is obviously the training dataset D_T. As shown in the results, a value of $n_N = 2D$ has shown to be an appropriate choice for n_N.

Rows 1 to 6 of algorithm 2 perform the preliminary operations of variable initializations:

- Step 1 creates an initial set of $n_{T,\text{in}}$ training points ($\mathbf{X}_{T,\text{in}}$) by a space-filling algorithm like LHS. Since NNAS-D uses n_N neighbors, $n_{T,\text{in}}$ must be greater than n_N.
- Step 2 evaluates the training response values $\mathbf{y}_{T,\text{in}} = f(\mathbf{X}_{T,\text{in}})$.
- Step 3 combines $\mathbf{X}_{T,\text{in}}$ and $\mathbf{y}_{T,\text{in}}$ to create the initial training dataset.
- Step 4 initializes the N_{st} matrix where the algorithm will store the indeces of the neighbors used to compute the local RMSE at each training point.
- Step 5 initializes the RMSE vector where the algorithm will store the values of the computed RMSE.
- Step 6 initialize the PE_{st} matrix where the algorithm will store the local predicted error at the neighbors of each $\mathbf{x}_{T,i}$.

N_{st} stores the indices of the n_N nearest neighbors of each samples in D_T (row 18), and it is used in row 16 to prevent the computation of local RMSE for points whose neighborhoods have been unaffected by the addition of the new sample at the end of the previous iteration (row 28). Similarly, PE_{st} stores the computed prediction errors at the neighbors of training points, and is updated only if there is a change in the neighborhood compositions caused by the previous $\mathbf{x}_{\text{next}}$ inclusion.

The while loop within rows 7 to 29 is the core of the NNAS-D algorithm, and it is entered until the stopping criterion (SC) is not satisfied. More specifically:

- Row 8 updates the sample-to-sample distances matrix Δ by computing the distance between the training point added at the end of the previous iteration and the samples already in D_T (at the first iteration, Δ must be initialized with the distances between points in the initial training set).

- The for loop in rows 9 to 11 identifies the n_N neighbors of the new training point by using the efficient kNN-S algorithm, and it stores the indices of the neighbors in the matrix N_{st}. The kNN-S takes as input the distance matrix Δ previously updated to reduce the algorithm computational cost, as described in section 5.7.

- The for loop in rows 12 to 14 checks if the addition of the new training sample at the end of the previous iteration has affected the neighborhoods of the existing training samples. This operation is efficiently implemented by using Δ and the neighbor indices stored in N_{st}, without the need of running a completely new kNN-S.

- The third for loop (rows 15 to 20) is responsible for the update of the local RMSE vector. As it possible to see, the actual RMSE evaluation (row 17) is done only if the updated neighborhood ($N^{(\mathrm{i})}$) differs from what was computed at the previous iteration ($N_{\mathrm{st}}^{(\mathrm{i})}$) (condition of the inner if statement (row 16)).

- Rows 21 and 22 use RMSE, Δ and D_T to compute the refinement and exploration metrics, respectively.

- Rows from 23 to 27 identify the new training sample by using the stochastic Pareto-ranking selection and the approximate corner search described in section 5.3.3 and appendix C.

- Rows 26 appends the new training information to the training dataset D_T.

input : $f(\mathbf{x})$, n_N, SC
output: D_T

Create $n_{T,\text{in}}$ initial samples ($\mathbf{X}_{T,\text{in}}$) using a space-filling algorithm ($n_{T,\text{in}} > n_N$);
Evalute $\mathbf{y}_{T,\text{in}} = f(\mathbf{X}_{T,\text{in}})$;
Create the initial $D_T = \{\mathbf{X}_{T,\text{in}}, \mathbf{y}_{T,\text{in}}\}$;
Initialize N_{st} as a zeros matrix of size $n_{T,\text{in}} \times n_N$;
Initialize RMSE as a zeros vector of size $n_{T,\text{in}}$;
Initialize PE_{st} as a zeros matrix of size $n_{T,\text{in}} \times n_N$;
while *SC **is** false* **do**
    $\Delta \leftarrow$ update `distance(`$\mathbf{X}_T$`)`;
    for $i = n_T - n_{\text{add}} + 1 : n_T$ **do**
        $N^{(i)} \leftarrow$ `knnsearch(`i,Δ`)`;
    end
    for $i = 1 : n_T - n_{\text{add}}$ **do**
        $N^{(i)} \leftarrow$ `neighborhoodUpdate(`i,N_{st},Δ`)`;
    end
    for $i = 1 : n_T$ **do**
        if $N^{(i)} \neq N_{\text{st}}^{(i)}$ **then**
            $[\text{RMSE}_i, \text{PE}_i] \leftarrow$ `RMSEeval(`$N^{(i)}, \mathbf{X}_T(N^{(i)}), \mathbf{y}_T(N^{(i)})$`)`;
            $N_{\text{st}}^{(i)} \leftarrow N^{(i)}$;
        end
    end
    $R \leftarrow (\text{RMSE}, D_T)$;
    $E \leftarrow$ `VoronoiVolEst(`$\mathbf{X}_T$`)`;
    $\text{PF}^* \leftarrow$ `ParetoRanking(`R,E`)`;
    $[\mathbf{x}_T^*, t^*] \leftarrow$ `invCumulative(`PF^*`)`;
    $\mathbf{X}_{\text{crn}} \leftarrow$ `VoronoiCornerApprox(`$\mathbf{x}_T^*$,E_{i^*}`)`;
    $\mathbf{x}_{T,\text{new}} \leftarrow$ `cornerSelection(`$\mathbf{x}_T^*$,t^*`)`;
    $y_{T,\text{new}} \leftarrow$ `eval(`$f(\mathbf{x}_{T,\text{new}})$`)` ;
    $D_T \leftarrow$ `append(`$\mathbf{x}_{T,\text{new}}$,$y_{T,\text{new}}$`)` ;
end

Algorithm 2: NNAS-D algorithm

5.4 Avoiding Solver Critical Errors

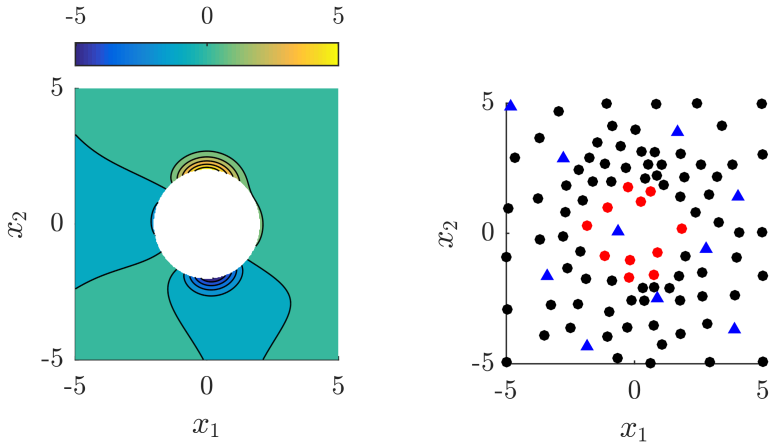
Sampling algorithms usually sample the design space of an unknown response, and therefore it could happen that some designs are infeasible or their performance cannot be assessed using standard settings in the solver. In these cases, critical errors may occur in the solver, and consequentially no response y_T is returned to the sampling algorithm. Therefore, a robust sampling algorithm should be able to continue the sampling process without being affected by critical errors in the computational simulation solver, and it should also give to the user the opportunity of adjusting the simulation and refreshing the training dataset without stopping the sampling process.

Both NNAS formulations can be adapted to handle critical solver error situations by a minimal modification in the sampling algorithm. In particular, the training samples with a missing response value are not considered during the definition of the neighborhoods that will be used for either NLI or local RMSE evaluation. However, the whole training set – even the samples with missing response – must be considered for the evaluation of the exploration metric. Indeed, the expected sample distribution should continue to concentrate samples in region with high departure from linearity, but the algorithm must continue the exploration of the entire design space even in region with missing response.

An example of sample distribution obtained by the application of NNAS-D for the design space sampling of an analytic test case is shown in figure 5.8.

5.5 Multi-response Formulation

Real-world engineering problems often involve analyses with multiple responses: consider for example the aerodynamic lift and drag of a wing, or the static and dynamic responses of a structure. In this context, sampling techniques that balance exploration and refinement based on consideration of all of the responses simultaneously are particularly valuable.



(a) Function contour

(b) Sample distribution obtained with NNAS-D

Figure 5.8: Example of sample distribution obtained by the application of NNAS-D for the design space sampling of an analytic test case with infeasible region. (a) shows the contour plot of the response with a circular infeasible region in the center of the domain. (b) plots the resulting sample distribution with red markers representing the samples in the infeasible region

The applicability of both NNAS formulations can be easily extended to n_f-response applications by considering the maximum of the refinement metrics individually computed for each response f_j ($R_{T,i}^{(j)}$) as refinement metric $R_{T,i}$:

$$R_{T,i} = \max(R_{T,i}^{(1)}, \ldots, R_{T,i}^{(n_f)}) \tag{5.28}$$

where $R_{T,i}^{(j)}$ is the refinement metric computed for the jth response at the ith training point.

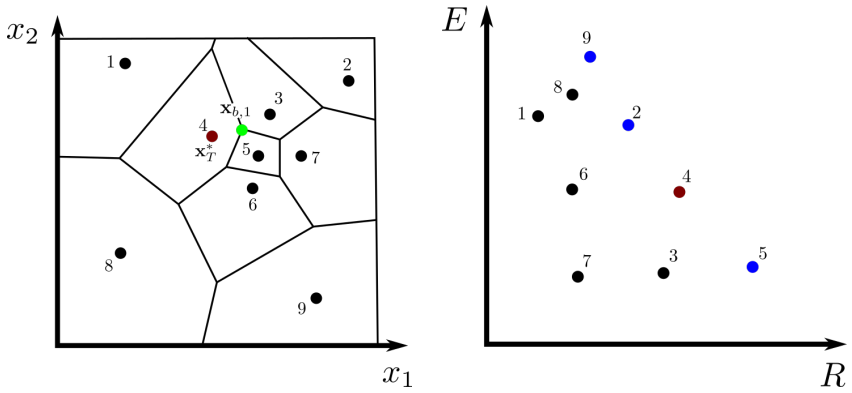
5.6 Batch-mode Formulation

The high computational cost of current high fidelity solvers for engineering applications leads to a frequent use of high performance computing (HPC) resources, in particular distributed computing architectures. These architectures give the opportunity to simultaneously test several designs by running each simulation in a different computer node of the architecture, thereby reducing the total runtime required to complete the analysis. There-

fore, NNAS must have an algorithm architecture able to identify n_B samples at each iteration ("batch-mode") to efficiently use distributed computing HPC resources. The proposed "batch-mode" NNAS assumes that the evaluation of all the n_B designs is completed before the next batch of n_B training points is generated by the sampling algorithm. The author is aware that in a real problem the evaluation time may significantly vary across the design space, but the development of a sampling strategy able to efficiently deal with this kind of situation is out of the scope of this dissertation, and it could be pursued in a future work.

The simplest approach to create a batch of n_B training points at each sampling iteration is to introduce a for loop that repeats n_B times the stochastic selection of $\mathbf{x}_T^*$ and the identification of $\mathbf{x}_{\text{next}}$. Once $\mathbf{x}_{\text{next}}$ is appended to the batch, it is crucial to discard the "used" $\mathbf{x}_T^*$ from the possible stochastic selection candidates before repeating the process to identify the next $\mathbf{x}_{\text{next}}$ to include in the current batch. Indeed, if $\mathbf{x}_T^*$ is not discarded and it is selected again as the identifier of VC^*, the process returns a point at the same location of the previous point in the batch. Unfortunately, discarding $\mathbf{x}_T^*$ is not enough to prevent point clustering within the same batch. Consider the example in figure 5.9 in which the stochastic selection has chosen point 4 as $\mathbf{x}_T^*$ and $\mathbf{x}_{b,1}$ is the first point added to the current batch. If point 4 is removed from the R-E domain and the repeated stochastic selection chooses point 5 – which is in PF^* – as $\mathbf{x}_T^*$, the second batch point $\mathbf{x}_{b,2}$ can coincide with $\mathbf{x}_{b,1}$ because the former is located on a Voronoi corner that is shared by VC_4 and VC_5. In the actual NNAS implementation with approximate Voronoi tessellation, this may lead to an excessive point clustering. Therefore, to properly use NNAS in batch-mode it is necessary – during the creation of a batch – to remove both the "used" $\mathbf{x}_T^*$ and the training points whose VCs share with it the corner added as batch point. Considering the example in figure 5.9, training points 4,3, and 5 must be discarded before proceeding with the selection of $\mathbf{x}_{b,2}$.

The NNAS-D algorithm that makes it possible to create a batch of n_B samples at each sampling iteration as detailed in algorithm 3, but similar algorithm modifications can be



(a) Sample distribution (b) R-E domain NNAS-D

Figure 5.9: Neighborhoods affecting the NNAS batch selection

applied to algorithm 1 to obtain an NNAS-B batch-mode formulation. Small changes from algorithm 2 are the new input n_B representing the number of samples in the batch, and the modified indices in for loops at rows 9 and 12 to take into account that n_B samples are added to the dataset at every sampling iteration . The more consistent modification is the while loop from rows 23 to 30. This loop is entered until the entire batch of samples has been created or there are no remaining training points available as candidates for the stochastic selection. Some particular details of note:

- Row 24 applies the Pareto-ranking procedure to find the Pareto frontier representing the solution of eq.(5.1) considering only the $[R, E]$ pairs related to the training points not yet discarded.

- Row 29 discards $\mathbf{x}_T^*$ and all the training points whose Voronoi cell share with it the corner selected in row 26 as new batch sample from the possible candidates for the stochastic selection.

5.7 Computational Complexity

This section derives the computational complexity of NNAS-D algorithm, which is found to be $\mathcal{O}(n_T D^2 + n_T \log_2(n_T) + D^4)$, in which the first term is related to the neighborhood

input : $f(\mathbf{x})$, n_N, SC, n_B
output: D_T

Create $n_{T,\text{in}}$ initial samples ($\mathbf{X}_{T,\text{in}}$) using a space-filling algorithm ($n_{T,\text{in}} > n_N$);
Evalute $\mathbf{y}_{T,\text{in}} = f(\mathbf{X}_{T,\text{in}})$;
Create the initial $D_T = \{\mathbf{X}_{T,\text{in}}, \mathbf{y}_{T,\text{in}}\}$;
Initialize N_{st} as a zeros matrix of size $n_{T,\text{in}} \times n_N$;
Initialize RMSE as a zeros vector of size $n_{T,\text{in}}$;
Initialize PE_{st} as a zeros matrix of size $n_{T,\text{in}} \times n_N$;
while *SC **is** false* **do**
    $\Delta \leftarrow$ update `distance(`$\mathbf{X}_T$`)`;
    for $i = n_T - n_B + 1 : n_T$ **do**
        $N^{(i)} \leftarrow$ `knnsearch(`i,Δ`)`;
    end
    for $i = 1 : n_T - n_B$ **do**
        $N^{(i)} \leftarrow$ `neighborhoodUpdate(`i,N_{st},Δ`)`;
    end
    for $i = 1 : n_T$ **do**
        if $N^{(i)} \neq N_{\text{st}}^{(i)}$ **then**
            $[\text{RMSE}_i, \text{PE}_i] \leftarrow$ `RMSEeval(`$N^{(i)}, \mathbf{X}_T(N^{(i)}), \mathbf{y}_T(N^{(i)})$`)`;
            $N_{\text{st}}^{(i)} \leftarrow N^{(i)}$;
        end
    end
    $R \leftarrow (\text{RMSE}, D_T)$;
    $E \leftarrow$ `VoronoiVolEst(`$\mathbf{X}_T$`)`;
    while $n_{\text{T,new}} < n_B$ *and* $R \neq \varnothing$ **do**
        $\text{PF}^* \leftarrow$ `ParetoRanking(`R,E`)`;
        $[\mathbf{x}_T^*, t^*] \leftarrow$ `invCumulative(`PF^*`)`;
        $\mathbf{X}_{\text{crn}} \leftarrow$ `VoronoiCornerApprox(`$\mathbf{x}_T^*$,E_{i^*}`)`;
        $\mathbf{x}_{T,\text{new}} \leftarrow$ `cornerSelection(`$\mathbf{x}_T^*$,t^*,$\mathbf{X}_{\text{crn}}$`)`;
        $\mathbf{X}_{T,\text{new}} \leftarrow$ `append(`$\mathbf{x}_{T,\text{new}}$`)`;
        $[R, E] \leftarrow$ `removeContiguous(`R,E,$\mathbf{X}_T$,$\mathbf{x}_{T,\text{new}}$`)`;
    end
    $\mathbf{y}_{T,\text{new}} \leftarrow$ `eval(`$f(\mathbf{X}_{T,\text{new}})$`)` ;
    $D_T \leftarrow$ `append(`$\mathbf{X}_{T,\text{new}}$,$\mathbf{y}_{T,\text{new}}$`)` ;
end

Algorithm 3: NNAS-D algorithm

identifications, the second is a result of the Pareto-ranking procedure, and the last is due to the matrix inversion required for fitting the hyperplanes in the refinement metric evaluation. With a similar derivation is it possible to prove that NNAS-B is of the same order of computational complexity as NNAS-D.

Considering algorithm 3 as a reference algorithm, the operations that are relevant for the computational complexity estimation are:

- Row 8. The matrix Δ is continuously built up and updated during the NNAS algorithm (row 7) to store the euclidean distances between the points in D_T. The update of the Δ matrix has an algorithm complexity of $\mathcal{O}(n_B(n_T D))$.

- Rows 9 to 11. Row 10 identifies the n_N nearest neighbors for each of the new added points via kNN-S. Because the distances between training points are already available in Δ, this operation requires $\mathcal{O}(n_N n_T)$ operations. Since it is repeated n_B times, this loop has a computational complexity of $\mathcal{O}(n_B(n_N n_T))$.

- Rows 12 to 14. This for loop implies $n_T - n_{\text{add}}$ pairwise comparisons and therefore is completed in $\mathcal{O}(n_T - n_{\text{add}})$ operations.

- Rows 15 to 20. As described in the NNAS-D implementation section (section 5.3.4), local RMSE is computed only for the $\mathbf{x}_{T,i}$ whose neighborhoods $N^{(\text{i})}$ have been affected by the addition of the last training samples. Therefore, to quantify the computational cost of the RMSE evaluation, it is necessary to estimate the number of neighborhoods affected by the addition of $\mathbf{X}_{T,\text{new}}$ in D_T. Unfortunately, this number is obviously influenced by the non-linearity of the response f that drives the sample refinement across the design space. However, it is possible to obtain a rough estimation of the number of neighborhoods affected if we assume that the samples are uniformly randomly added to D_T. In this case, it is possible to prove that the average number of neighborhoods affected by the addition of a sample in D_T is equal to n_N, and it is independent from the number of samples in D_T. Since each local

RMSE evaluation (row 16) requires $\mathcal{O}(n_N^3)$ operation to solve the RMSE linear system (eq.(5.23)), the computational cost for the local RMSE evaluation is $\mathcal{O}(n_B n_N^4)$.

- Row 21 has negligible complexity.

- Row 22 estimates the Voronoi cell volumes. As already explained, the Voronoi tessellation algorithm is extremely computationally expensive (at worst $\Omega(n^{\frac{D}{2}})$ [51]), and NNAS uses a Monte-Carlo approach to estimate the Voronoi cell volumes. The Monte-Carlo approach requires to generate n_R random points in the domain, and to count the number of random points that have a particular training point as the closest point. As suggested in the LOLA-V formulation ([20]), the number of random points should be proportional to the number of training points in D_T, i.e. $n_R = k_V n_T$. Instead of creating fresh random points at each iteration that would have required $\mathcal{O}(n_T^2 D)$ operations, only $k_V n_B$ new random points are generated at each iteration and the process is completed in $\mathcal{O}(n_R n_B D)$.

- Row 24. The Pareto-ranking has a complexity of $\mathcal{O}(n_T \log_2(n_T))$ [54].

- Row 25 has negligible computational complexity.

- Row 26. The Voronoi corner estimation using the procedure described in appendix C requires the identification of the $D+1$ closest neighbors of $k_C D$ random points. The procedure is completed in $\mathcal{O}(k_C n_T D^2)$ via kNN-S.

- Rows from 27 to 29 have negligible computational complexity

Combining all the analyzed operations involved in NNAS algorithm and considering $n_N \propto D$, the overall computational complexity as a function of n_T and D is:

$$C_{\text{iter}} = \underbrace{\mathcal{O}(n_B(n_T D))}_{\text{row 8}} + \underbrace{\mathcal{O}(n_B(n_N n_T))}_{\text{rows 9-14}} + \underbrace{\mathcal{O}(n_T - n_B)}_{\text{rows 12-14}} + \underbrace{\mathcal{O}(n_B n_N^4)}_{\text{rows 15-20}} +$$

$$+ \underbrace{\mathcal{O}(k_V n_T n_B D)}_{\text{row 22}} + \underbrace{\mathcal{O}(n_T \log_2(n_T))}_{\text{row 24}} + \underbrace{\mathcal{O}(k_C n_T D^2)}_{\text{row 26}} \tag{5.29}$$

$$= \mathcal{O}(n_T D^2 + n_T \log_2(n_T) + D^4) \tag{5.30}$$

Therefore, the NNAS algorithm has a quasilinear complexity with respect to number of training samples in D_T.

CHAPTER 6
NEAREST NEIGHBORS ADAPTIVE SAMPLING RESULTS

This chapter presents the results of numerous analyses conducted on sample distributions obtained by applying NNAS algorithms for the design space sampling of of several example functions. These analyses have been planned to stress different aspects of the proposed technique such as sampling efficiency with respect to number of samples and computational time, ability to deal with critical errors in the computational simulation solver, multi-response problems, and batch-mode applications.

All of the sampling processes have been completed using a surrogate modeling software framework developed specifically for this dissertation. The framework is written using an object-oriented implementation in the MATLAB language, and it has the following characteristics:

modular It is possible to add new components such as SM functional forms or sampling techniques without changing the overall structure of the simulation framework.

parallelizable The code makes it possible to run specific parts of the simulation using parallel computing resources.

process time bookkeeping The framework records the time required to perform each step of the surrogate modeling process for the assessment of t_{tot} performance.

Design spaces of real engineering applications are usually defined by design variables with a variety of units and frequently have several orders of magnitude of difference between them (e.g. the lift coefficient and the surface area of a wing). For this reason, it is generally viewed as good practice to perform the sampling process on a normalized design space to avoid any influence of the magnitude of design variables on the sampling performance and behavior. Even though the NNAS algorithm does not specifically require

the normalization of the design space, all the results presented in this chapter have been obtained by running NNAS on a 0-1 normalized design space.

The first part of this chapter describes the analysis plan that has been followed to generate the results presented in the following sections (Sections 6.2 to 6.8).

6.1 Analysis Plan

Both NNAS-B and NNAS-D techniques proposed in this dissertation are tested for the design space sampling of several test cases representing both analytic functions and black-box solvers, in particular:

- ten two-dimensional analytic functions;
- one five-dimensional engineering test case based on the XRotor code ([23]);
- one two-dimensional two-response analytical case;
- one five-dimensional three-response test case based on XRotor;
- one two-dimensional analytic function with infeasible region;
- one two-dimensional test case based on XRotor with infeasible region.

It is important to underline that the term "infeasible region" does not indicate the portion of the design space where designs violate some constraints, but instead the region of the design space where the evaluation of f returns a critical error. A detailed description of the test functions along with the variable ranges is provided in appendix D. The analytic test functions are selected among the ones usually considered as example problems in the surrogate modeling literature [20, 31, 47, 84]; they range from smooth behavior as the Branin function to responses with non-linear features clustered in a limiter region of the design space as in the low-noise-amplifier current function, thereby including the common response characteristics of real engineering problems [84]. The engineering test cases are

based on the XRotor code which is a popular physics-based black-box solver used to estimate propeller performance. XRotor is a valuable tool to test sampling techniques because it is extremely fast, it makes it possible to easily increase the problem dimensionality, and, since it is physics-based, it embodies all the practical issues that can be encountered using other black-box engineering software, i.e. solution divergence, critical errors.

Each sampling process is repeated 15 times with a different initial training dataset to remove the dependence of results from the specific set of initial samples, and it is stopped when D_T contains 200 and 5000 samples for the 2D and 5D cases, respectively. The initial training datasets $D_{T,0}$ are generated using a maximin Latin hypercube space-filling procedure, specifically the `lhsdesign` Matlab routine with 20 iterations. The representation quality of the resulting sample distributions is assessed by training a global SM, and computing the NRMSE with respect to a validation dataset D_V:

$$\mathrm{NRMSE} = \frac{\sqrt{\frac{1}{n_V}\sum_{i=1}^{n_V}\left(f(\mathbf{x}_{V,i}) - \tilde{f}(\mathbf{x}_{V,i})\right)^2}}{\max(f) - \min(f)} \tag{6.1}$$

where $\tilde{f}$ represents the SM, and the normalization factor is the response range of variability. The locations on the unit cube of the 15 sets of starting points and the sets of validation points are kept the same for problems with the same dimensionality. In this way, we avoid the introduction of additional aleatory effects which can be caused by the randomization of the validation points at each repetition. The validation datasets D_V consist of 10000 and 100000 points for the $2D$ and $5D$ problems, respectively.

The SMs considered to assess the quality of the sample distributions are RBFs with 4 different kernel functions, namely cubic, gaussian, inverse multiquadric, and multiquadric (appendix B). The RBF tuning coefficients θ_i are found during the SM training process by the minimization of the NRMSE that is conducted using a line search algorithm (`fmincon` Matlab routine) started from several different initial points in the θ domain. The number of θ initial points has been set equal to three times the problem dimensionality, and similarly

to what was done with $D_{T,0}$ and D_V, the initial θ values are kept fixed for cases with the same dimensionality.

The NNAS sampling performance is compared with the performance obtained using other state-of-the-art sampling techniques, namely an LHS (`lhsdesign` Matlab routine), the CV-based MDSAS with active SM selection presented in [31] and chapter 4, and a LOLA-Voronoi MISAS proposed in [20]. It should be noted that the MDSAS strategy is tested only on single-response 2D problems because of the computational time required by the computation of LOO-CV error at the beginning of each iteration. NNAS

All the plots with errorbars included in this chapter display the median and an errorbar ranging from the 25% to the 75% quantiles of the 15 repetition results. Furthermore, two-dimensional histograms (e.g. figure 6.6a) are used for the visualization of the average two-dimensional point distribution. These plots are obtained by dividing the design space into a 9×9 uniform grid, counting the number of samples which fall within each tile, normalizing the resulting count as $n_{\mathrm{T}}/(9 \times 9)$ (which represents the number of samples expected to fall within each tile if the samples were perfectly homogeneously distributed across the design space), and finally averaging over the 15 repetitions. Therefore, the resulting metric is an indicator of the departure of a sample distribution from a perfectly homogeneous one with the same number of points.

The experiment plan just described is designed to analyze different aspects of the sampling process that are crucial to validate the thesis hypotheses introduced in chapter 3 and some assumptions made in algorithm development. In particular, section 6.2 tests the influence of a $\hat{\rho} \neq 1$ on sampling behavior, section 6.3 shows the NNAS convergence, and section 6.5 illustrates the sampling efficiency and reduced computational cost of NNAS in comparison with other state-of-the-art sampling techniques.

6.2 Influence of ρ Target Value ($\hat{\rho}$)

As described in section 5.1, NNAS selects the design space region where $\mathbf{x}_{\text{next}}$ will be placed by means of a stochastic selection. This is based on a trapezoidal probability distribution (eq.(5.3)) which is intended to keep the ratio of the PF^* projections (ρ) as close as possible to a target value $\hat{\rho}$. A high-level discussion also suggested that a value of $\hat{\rho} = 1$ is a reasonable choice to obtain an exploration-refinement balance, due to the normalization factors proposed for the refinement and exploration metrics. This leaves two questions: is $\hat{\rho} = 1$ generally an appropriate value for target ρ? And does the stochastic selection in fact push the evolution of PF^* toward $\rho \approx \hat{\rho} = 1$?

To answer the first question, five sampling processes have been completed (as described in section 6.1) for the two-dimensional single-response test cases using five different values of $\hat{\rho}$, namely $0.25, 0.5, 1, 2, 4$. The average NRMSE value across the 15 repetitions and at different stages of the sampling process ($n_T = 30, 50, 100, 200$) is plotted versus $\hat{\rho}$ in figure 6.1. As expected, higher values of $\hat{\rho}$ are beneficial to responses that have high level or non-linearity concentrated in a restricted portion of the design space, i.e. f2DExponential (6.1d), f2DLNA (6.1e), f2DPeaks5 (6.1h), f2DPeaks8 (6.1i). However, if the response has a milder non-linearity behavior (e.g. f2DBranin in 6.1a, f2DNonPolySurf in 6.1b, f2DSixHumpCamelBack in 6.1c, f2DWitchHat in 6.1f, f2DPeaks3 in 6.1g), a high value of $\hat{\rho}$ may be detrimental to the sampling efficiency, and a value of $\hat{\rho} = 1$ leads to the best performance in most of the cases. More enlightening is the sampling of the f2DExponential2 response (figure 6.1j) which is characterized by two highly nonlinear regions at the opposite corners of the design space (figure 6.8q). In this case an appropriate exploration-refinement balance is crucial to achieve an accurate representation of both regions, and $\hat{\rho} = 1$ shows a substantial performance improvement in comparison with the other $\hat{\rho}$ values for an error level around $\text{NRMSE} \approx 10^{-2}$. Overall, this analysis suggests that – limited to the considered test cases – a value of $\hat{\rho} = 1$ is a reasonable choice to obtain an exploration-refinement

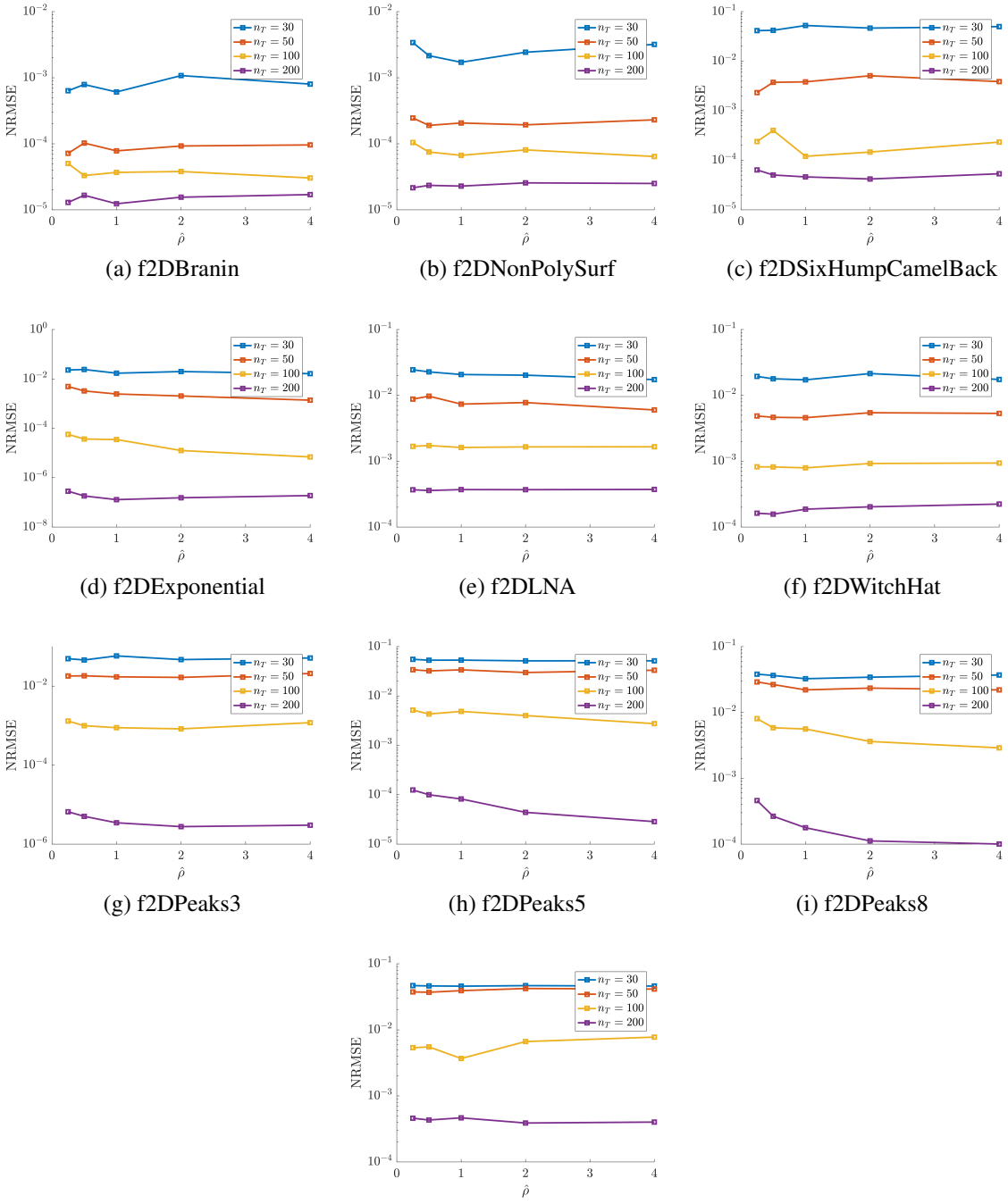
balance, and henceforth all the results presented in this chapter are obtained considering $\hat{\rho} = 1$.

Regarding the convergence of ρ toward $\hat{\rho} = 1$, figure 6.2 shows the evolution of PF^* during a sampling process of three different test functions. As samples are added to D_T, PF^* not only moves toward the center of the $R - E$ axes as expected, but its shape is also pushed toward the target value $\hat{\rho} = 1$. The convergence to $\rho \approx \hat{\rho} = 1$ is evident in figure 6.3, which displays the mean value of ρ over the 15 repetitions as function of n_T for the same three test functions.

6.3 Convergence of the Method

The plots in figures 6.4 and 6.5 show the evolution of NRMSE medians (across the 15 repetitions) obtained by training the four RBFs as the design spaces of the ten two-dimensional and one five-dimensional test functions are sequentially sampled by NNAS-B and NNAS-D (whiskers extend from 25% and 75% quantiles). As can be seen, all the trends have decreasing behavior except for the cases in which the specific SM itself is inadequate to represent the response (e.g. the cubic RBF for the f2DLNA). These set of results confirms the representation accuracy of training datasets sampled using both the NNAS algorithms, irrespective of the nonlinearity level of the response. The sampling performance comparison between NNAS algorithms and other state-of-the-art techniques is presented in section 6.5 after a description of some resulting sample distributions in section 6.4.

The other interesting aspect resulting from this analysis is the high variability in the rate of convergence across different SMs and test functions: for example, the cubic RBF is extremely efficient in modeling the f2DBranin response but completely unable to accurately represent the f2DLNA test function. The impossibility of finding an SM formulation that unequivocally outperforms all others in terms of modeling capability for any possible function has been already investigated in previous studies ([31]) and presented in chapter



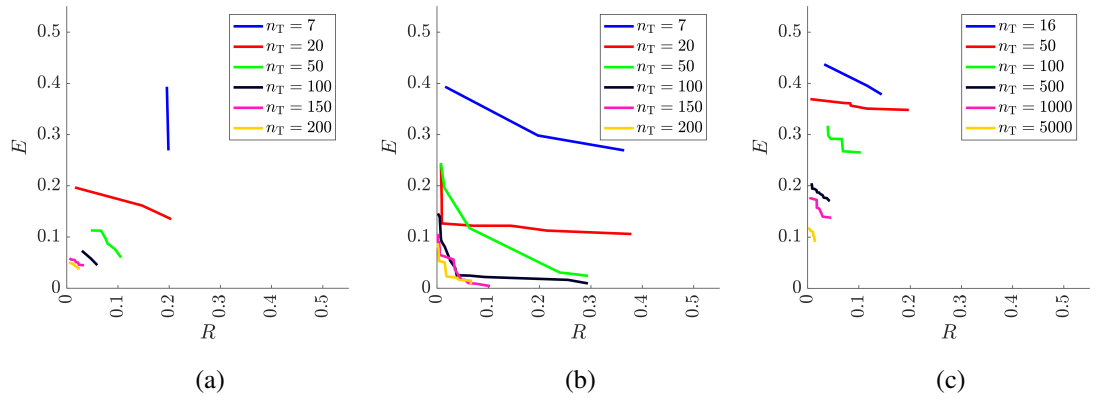
(a) f2DBranin (b) f2DNonPolySurf (c) f2DSixHumpCamelBack

(d) f2DExponential (e) f2DLNA (f) f2DWitchHat

(g) f2DPeaks3 (h) f2DPeaks5 (i) f2DPeaks8

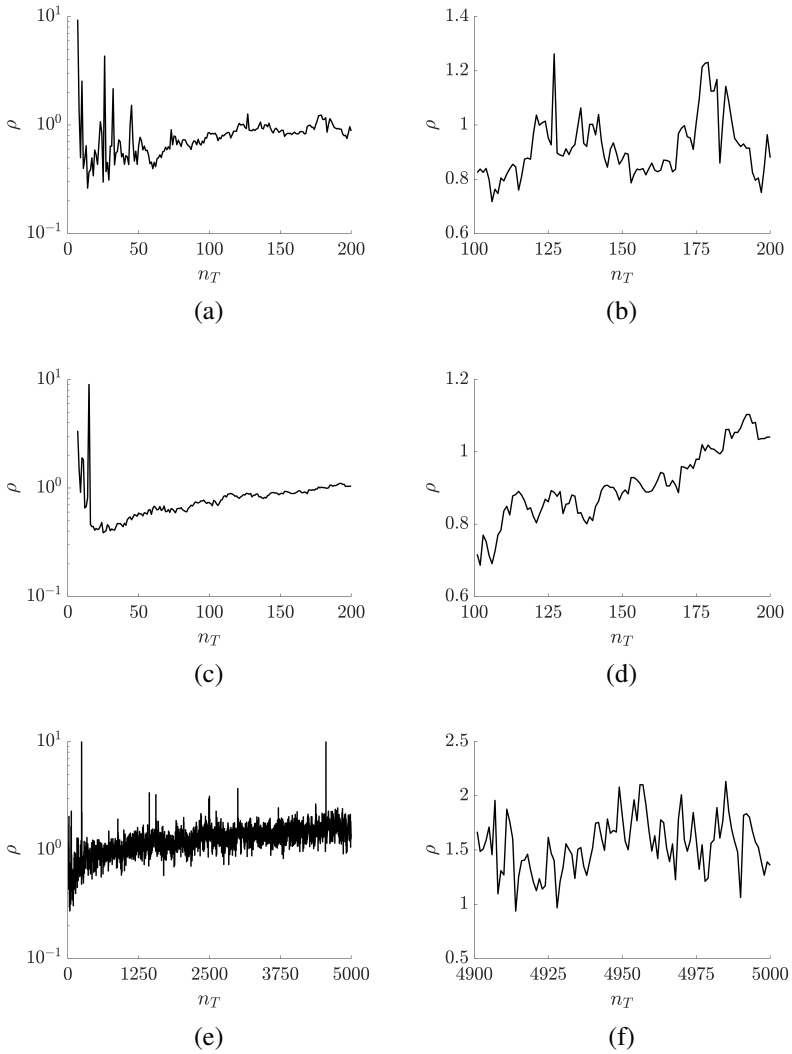
(j) f2DExponential2

Figure 6.1: Influence of $\hat{\rho}$ on the sampling performance assessed in terms of NRMSE



(a) (b) (c)

Figure 6.2: PF* evolution for f2DBranin (6.2a), f2DLNA (6.2b), and f5DXrotor (6.2c)



(a) (b) (c) (d) (e) (f)

Figure 6.3: ρ evolution for f2DBranin (6.3a, 6.3b), f2DLNA (6.3c, 6.3d), and f5DXrotor (6.3e, 6.3f). 6.3b, 6.3d, and 6.3f report the ρ evolution of the last 100 sampling iterations.

4. As discussed in chapter 3, this theoretical result is one of the main motivations behind the development of model independent sampling strategies.

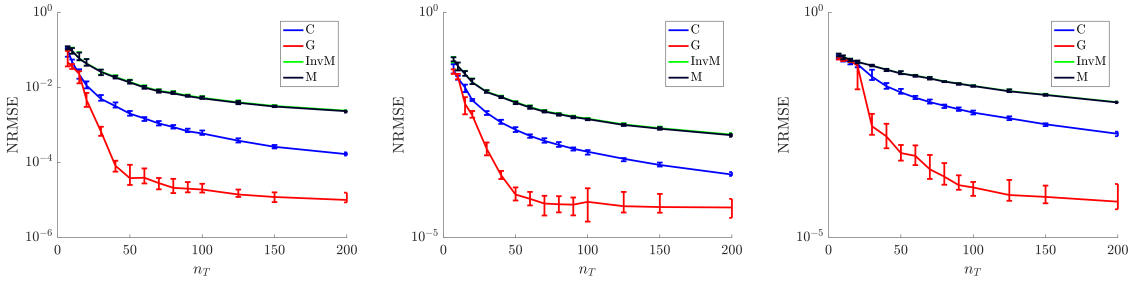
6.4 Example of Sample Distributions

This section presents an analysis of the resulting sample distributions obtained by using NNAS-B and NNAS-D for the design space sampling of three two-dimensional functions, specifically f2DBranin (figure 6.6), f2DLNA (figure 6.7), and f2DExponential2 (figure 6.8). These three example functions are representative of three different scenarios that are possible to encounter in the application of surrogate modeling techniques for engineering applications: a smooth response (f2DBranin), a situation with accentuated nonlinearity that is limited in a subregion of the domain (f2DLNA), and a case in which there are multiple portions of the design space where the response has a high departure from linearity (f2DExponential2).

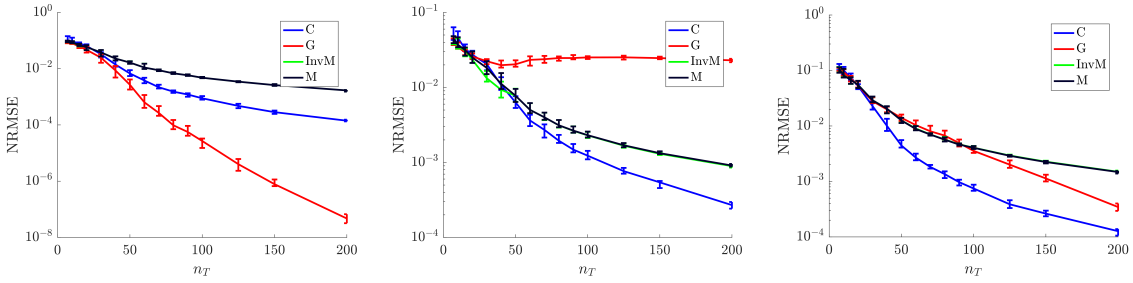
Figures 6.6 to 6.8 display the evolution of sample distribution as samples are sequentially added to the training set by NNAS-B and NNAS-D, specifically:

- the first two columns report an example of sample distribution for NNAS-B (first column) and NNAS-D (second column);
- the last two columns illustrate the average sample distribution across the 15 repetition for NNAS-B (third column) and NNAS-D (fourth column) (the description about the creation of these plots has been given in section 6.1);
- the first four rows are related to different number of samples in D_T, namely 20, 40, 80, 100;
- the contour of the response is displayed at the bottom of each figure.

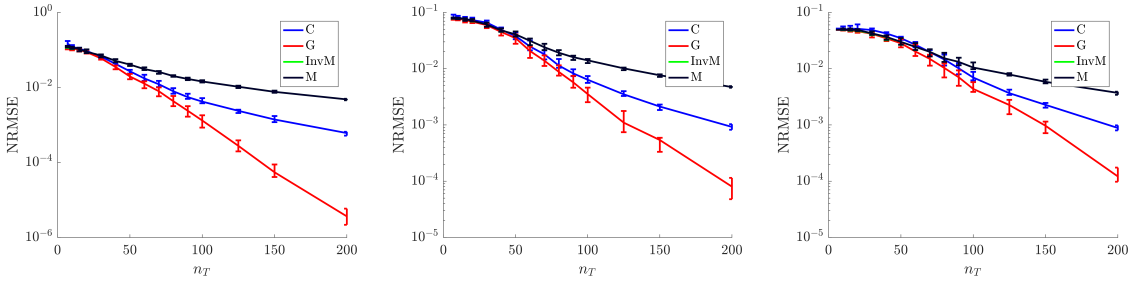
The effect of the NNAS refinement behavior is evident in figures 6.7 and 6.8: more samples are placed in the regions of the design space where the algorithm has indicated



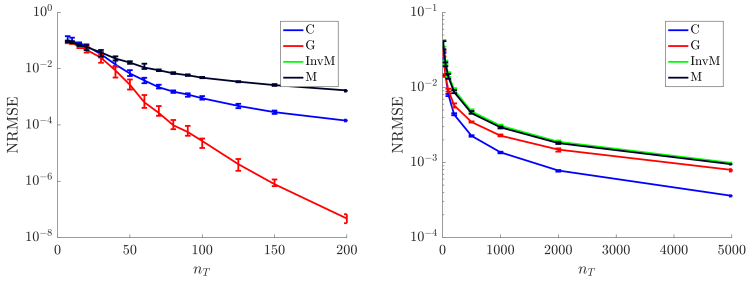
(a) f2DBranin (b) f2DNonPolySurf (c) f2DSixHumpCamelBack



(d) f2DExponential (e) f2DLNA (f) f2DWitchHat

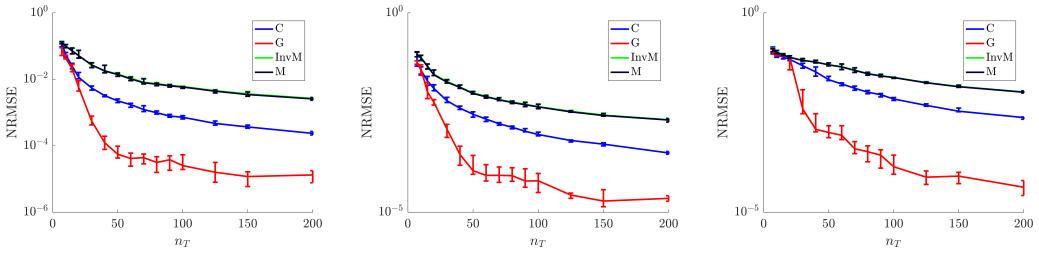


(g) f2DPeaks3 (h) f2DPeaks5 (i) f2DPeaks8

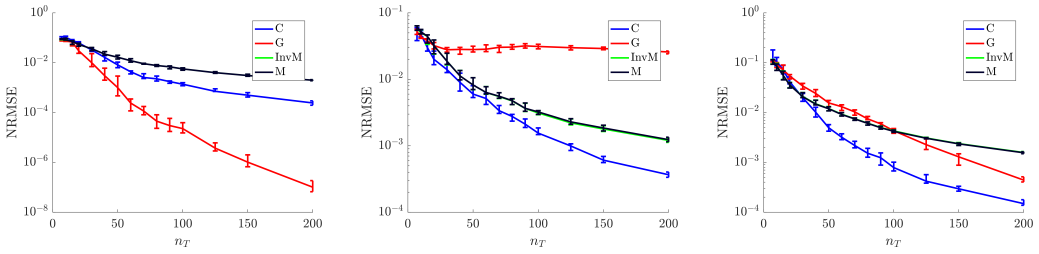


(j) f2DExponential2 (k) f5DXrotor

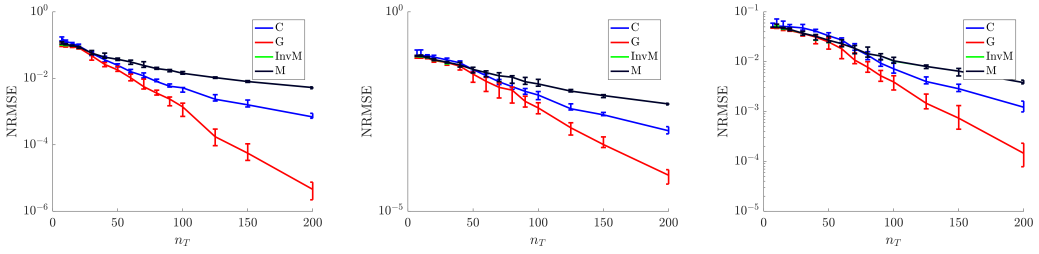
Figure 6.4: NNAS-B convergence using 4 RBFs, namely cubic (C), Gaussian (G), multiquadric (M), and inverse multiquadric (InvM)



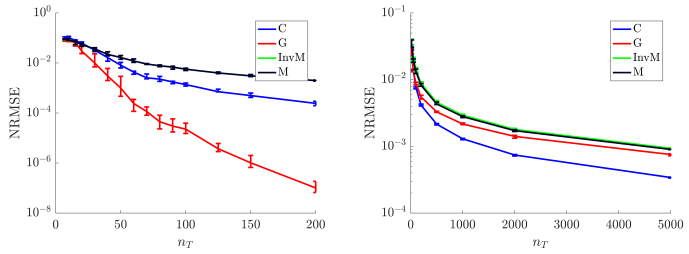
(a) f2DBranin (b) f2DNonPolySurf (c) f2DSixHumpCamelBack



(d) f2DExponential (e) f2DLNA (f) f2DWitchHat



(g) f2DPeaks3 (h) f2DPeaks5 (i) f2DPeaks8



(j) f2DExponential2 (k) f5DXrotor

Figure 6.5: NNAS-D convergence using 4 RBFs, namely cubic (C), Gaussian (G), multiquadric (M), and inverse multiquadric (InvM)

a higher departure from linearity of the response. By examining figure 6.8, it is possible to see how NNAS also detects the different degrees of response nonlinearity (figure 6.8q) and how it accordingly places more points in the top-right than in the bottom-left region. Instead, in the case of f2DBranin test function (figure 6.6), the refinement slightly pushes the samples away from the response plateau roughly along one diagonal of the design space. Additionally, the refinement effect is less notable than in the other test functions due to the milder variation of the f2DBranin response in the domain (figure 6.6q).

"Reading" the figures from the top to the bottom it is possible to obtain some insights about the evolution of sample distributions during the sampling process. In figure 6.6, the gentle variation of the f2DBranin test function leads to an almost uniform sample distribution with a light refinement on the left part of the design space where the response presents some nonlinearity (figure 6.6q). More interesting is the case of the f2DExponential2 test function which is characterized by two localized nonlinear regions (figure 6.8). The nonlinearity on the top-right corner of the design space is higher that the one in the bottom-left corner, but it is also confined to a smaller region. It is interesting to observe how NNAS initially focuses attention on refining the wider nonlinear bottom-left region until the exploration does detect the smaller but more prominent nonlinearity on the top-right of the design space.

Similar behaviors have been noticed in all the sampling processes completed for this study, showing the capability of both NNAS algorithms to adaptively distribute samples based on the nonlinearity characteristics of the response. In particular, the proposed refinement metrics and stochastic Pareto sampling approach are capable of sufficiently detecting the response nonlinearity and of successfully balancing the exploration and refinement behavior.

Some differences between the sampling behavior of NNAS-B and NNAS-D can be identified by comparing the resulting distributions of f2DLNA and f2DExponential2 example functions. Analyzing the last row of figures of f2DLNA and f2DExponential2, it is

possible to notice how NNAS-D leads to more clustered samples than NNAS-B. More important, the directional sampling criterion that guides NNAS-D seems to help the algorithm to better cluster the samples along the direction of maximum response nonlinearity, as it is evident by comparing figures 6.7m and 6.7n. As discussed in the next section, these differences in the sample distributions does not result in a substantial enhancement of NNAS sampling performance.

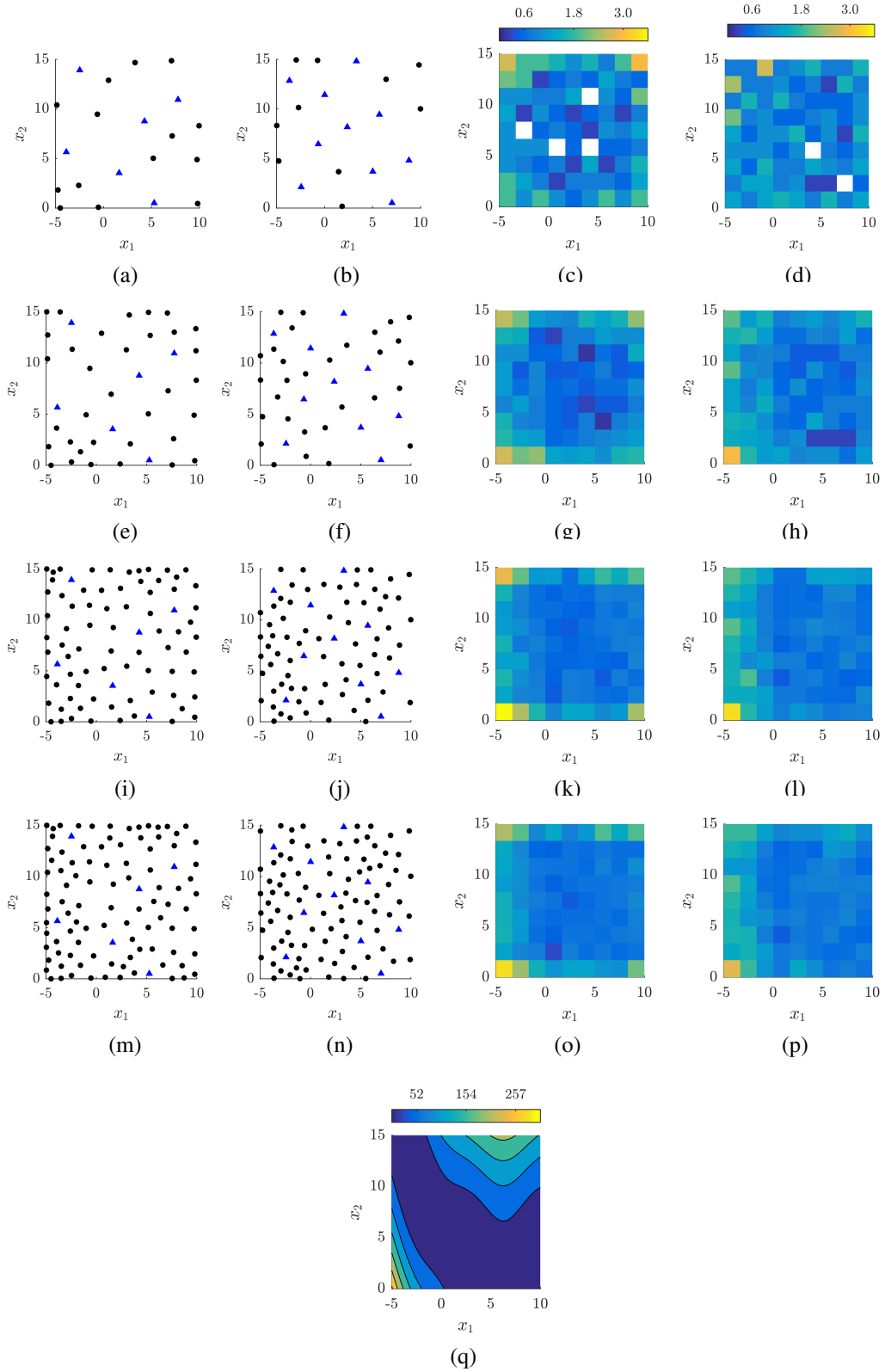


Figure 6.6: Example of samples distribution for f2DBranin function (q). The first two columns report an example of sample distribution for NNAS-B (a,e,i,m) and NNAS-D (b,f,j,n). The last two columns illustrate the average sample distribution across the 15 repetition for NNAS-B (c,g,k,o) and NNAS-D (d,h,l,m). The first four rows are related to different number of samples in D_T, namely 20, 40, 80, 100.

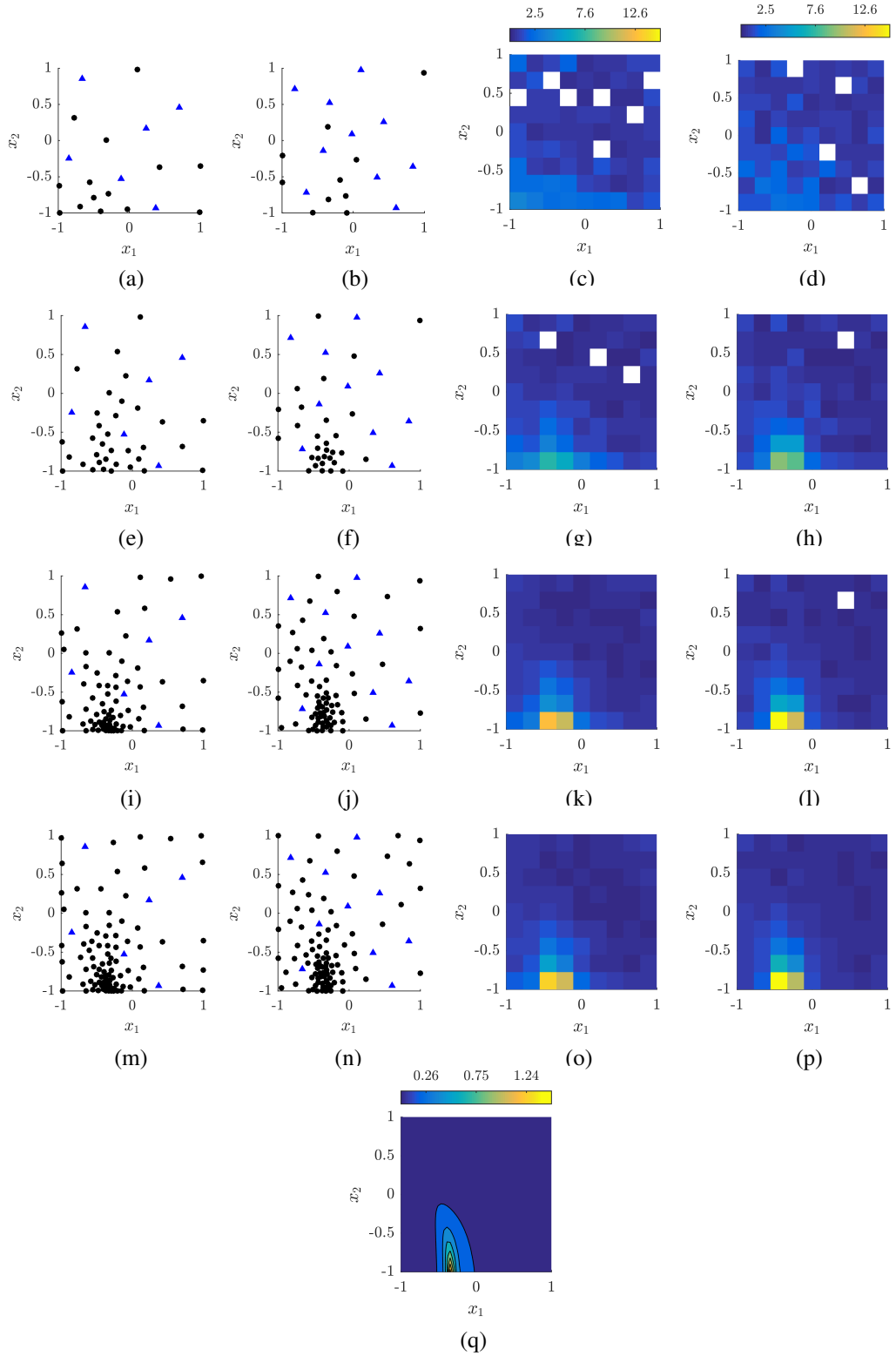


Figure 6.7: Example of samples distribution for f2DLNA function (q). The first two columns report an example of sample distribution for NNAS-B (a,e,i,m) and NNAS-D (b,f,j,n). The last two columns illustrate the average sample distribution across the 15 repetition for NNAS-B (c,g,k,o) and NNAS-D (d,h,l,m). The first four rows are related to different number of samples in D_T, namely 20, 40, 80, 100.

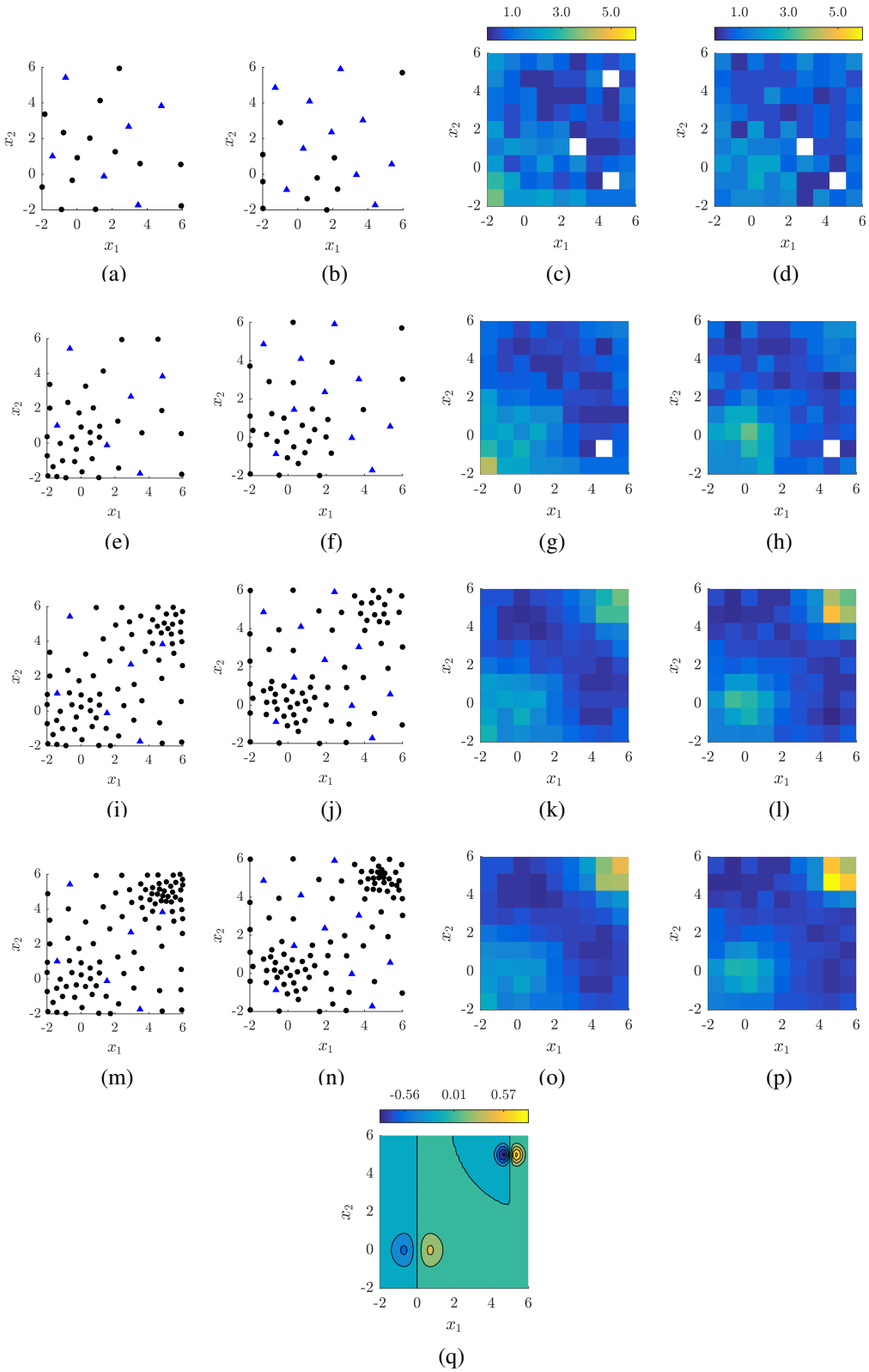


Figure 6.8: Example of samples distribution for f2DExponential2 function (q). The first two columns report an example of sample distribution for NNAS-B (a,e,i,m) and NNAS-D (b,f,j,n). The last two columns illustrate the average sample distribution across the 15 repetition for NNAS-B (c,g,k,o) and NNAS-D (d,h,l,m). The first four rows are related to different number of samples in D_T, namely 20, 40, 80, 100.

6.5 Comparison with Other Sampling Techniques

This section compares the NNAS algorithms with other state-of-the-art sampling techniques in terms of NRMSE as a function of number of samples (figure 6.9) and sampling time (figure 6.10). As introduced in section 6.1, the three additional sampling techniques considered for this comparison are LHS, LOLA-V, and CVVAS with active SM selection. The CVVAS with active SM selection architecture presented in chapter 4 is a model dependent strategy which uses NRMSE to select the most appropriate SM at the beginning of each iteration. Instead, NNAS, LOLA-V and LHS are model independent techniques, and therefore there is no information regarding the SM formulation that better describes the training set available at a particular stage of the sampling phase. For this reason, the NRMSE values reported in figures 6.9 and 6.10 for the model independent techniques (NNAS, LOLA-V, and LHS) are obtained by performing the SM selection and the NRMSE evaluation at the end of the sampling phase. Additionally, it is important at this point to clarify that even though extreme attention has been paid to efficiently implement all the algorithms, a quantitative analysis of the computational time performance can be challenging since it would be affected by the specific code implementation. On the other hand, the qualitative analysis presented in this chapter is valuable to understand how some aspects of each sampling algorithm impact the computational time. In the following results, LHS is not included in the time comparison plots because it is an *a priori* method and it does not have an appreciable sampling time, and CVVAS is considered only on the two-dimensional cases due to its extremely high computational cost for high dimensional applications.

Figure 6.9 displays the evolution of NRMSE median as a function of the number of training points for the ten two-dimensional and one five-dimensional test cases, and the five different sampling strategies evaluated. As it is possible to notice, all the adaptive sequential strategies have similar performance in terms of NRMSE vs n_T for most of the test functions, while – as expected – the sample distributions obtained with LHS are in-

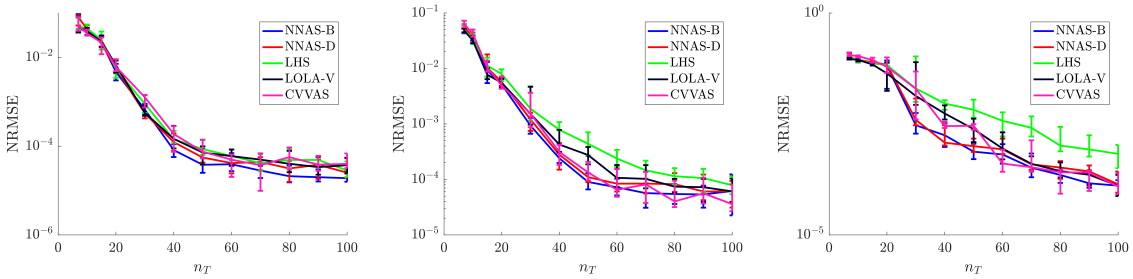
adequate to represent highly non linear responses like f2DExponential (6.9d), f2DLNA (6.9e), f2DWitchHat (6.9f), f2DPeaks5 (6.9g), f2DPeaks8 (6.9i), and the f2DExponential2 (6.9j). These results show the ability of the proposed NNAS techniques to efficiently sample the design space of different responses with performance comparable to other sampling techniques, and without the support of a global SM (as in CVVAS), thereby limiting the computational complexity of the algorithm.

However, both NNAS algorithms show a degraded sampling performance in comparison to LOLA-V (the other MISAS) when the algorithms are used for the sampling of responses with high and localized nonlinearity like f2DExponential (6.9d) and f2DExponential2 (6.9j). This advantage in terms of n_T of LOLA-V is related to its more advanced formulation of the neighborhoods used for the evaluation of the local nonlinearity, which however causes an increase in computational complexity ($\mathcal{O}(n_T^2)$) and sampling time (figure 6.10). Results in figure 6.9 also highlight how the directional sampling behavior of NNAS-D does not generally lead to a performance enhancement in comparison with NNAS-B; a clear reduction of number of samples required to reach a given level of accuracy is limited to f2DExponential and f2DExponential2 test cases.

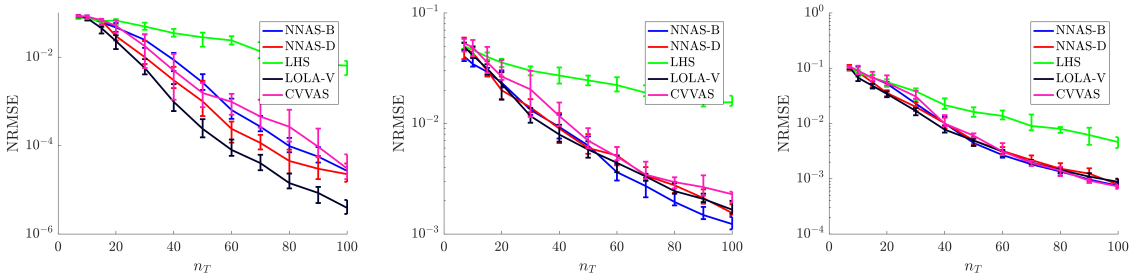
Figure 6.10 shows the evolution of NRMSE as a function of the total sampling time (t_{sampl}) for all the considered example cases. Although it is negligible, function evaluation time is not included in t_{sampl} to focus the comparison between the different techniques only on the computational time required by the sampling algorithm. As derived in section 5.7, NNAS techniques have a quasilinear complexity with respect to n_T ($\mathcal{O}(n_T \log_2(n_T))$) that is caused by the Pareto ranking in the E-R domain. In comparison, MDSASs like CVVAS have a complexity of at least $\mathcal{O}(n_T^3)$ if the supervising SM has a linear training process, moving to higher order dependences in cases of nonlinear SM techniques such as artificial neural network (ANN) or KRG. LOLA-V – which is the most similar MISAS to NNAS – has instead a computational complexity of $\mathcal{O}(n_T^2)$ due to the algorithm used to define the local neighborhoods involved in the refinement metric evaluation. Therefore, as expected,

the CVVAS has the lowest rate of convergence due to the need of retraining a full SM after each iteration to compute the refinement metric. Comparing NNAS and LOLA-V techniques, the difference in sampling time is evident in all the test cases, and it increases as more samples are added to the training set due to the quadratic dependence on n_T of LOLA-V complexity. For example, there is a difference of about two orders of magnitude in t_{sampl} required to reach a NRMSE level of 10^{-3} for the f5DXrotor problem. In this case, the advantage of NNAS against LOLA-V is partially due to the fewer samples required to reach this accuracy level (figure 6.9k), but is also due to the reduced algorithm complexity. Another interesting observation can be made by comparing figures 6.9d with 6.10d, and figures 6.9j with 6.10j. For both f2DExponential and f2DExponential2 cases, LOLA-V requires fewer samples than NNAS to reach $\mathrm{NRMSE} < 10^{-2}$, but this advantage in term of n_T vanishes or it is overturned when sampling time performance is considered. The main reason why LOLA-V preserves its advantage in t_{sampl} for f2DExponential and not for f2DExponential2 is the number of samples required to reach $\mathrm{NRMSE} < 10^{-2}$ in the two cases that is 30 and 70, respectively. The higher n_T together with the $\mathcal{O}(n_T^2)$ complexity overturns the LOLA-V advantage in favor of NNAS for f2DExponential2 case.

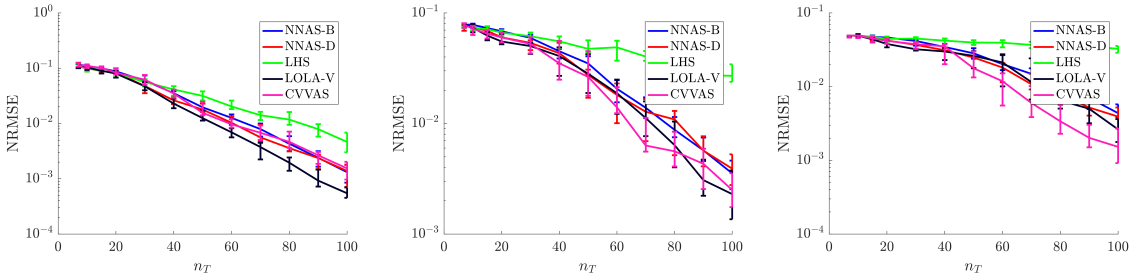
This final analysis of the effect of algorithm complexity on sampling time is extremely important if related with the study about total sampling time presented at the beginning of this dissertation in section 1.3. While a difference between 10^0 and 10^1 seconds has probably a negligible effect in the total SM creation time, completing the sampling process in 10^1 or 10^3 second (as in the f5DXrotor case for $\mathrm{NRMSE} < 10^{-3}$) may lead to a considerable reduction in overall surrogate modeling time and cost. This consequence is especially true for engineering applications with a low function evaluation time, and therefore the τ-effect in eq.(1.5) prevails.



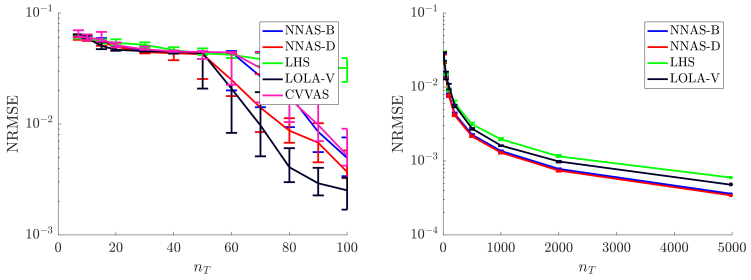
(a) f2DBranin (b) f2DNonPolySurf (c) f2DSixHumpCamelBack



(d) f2DExponential (e) f2DLNA (f) f2DWitchHat

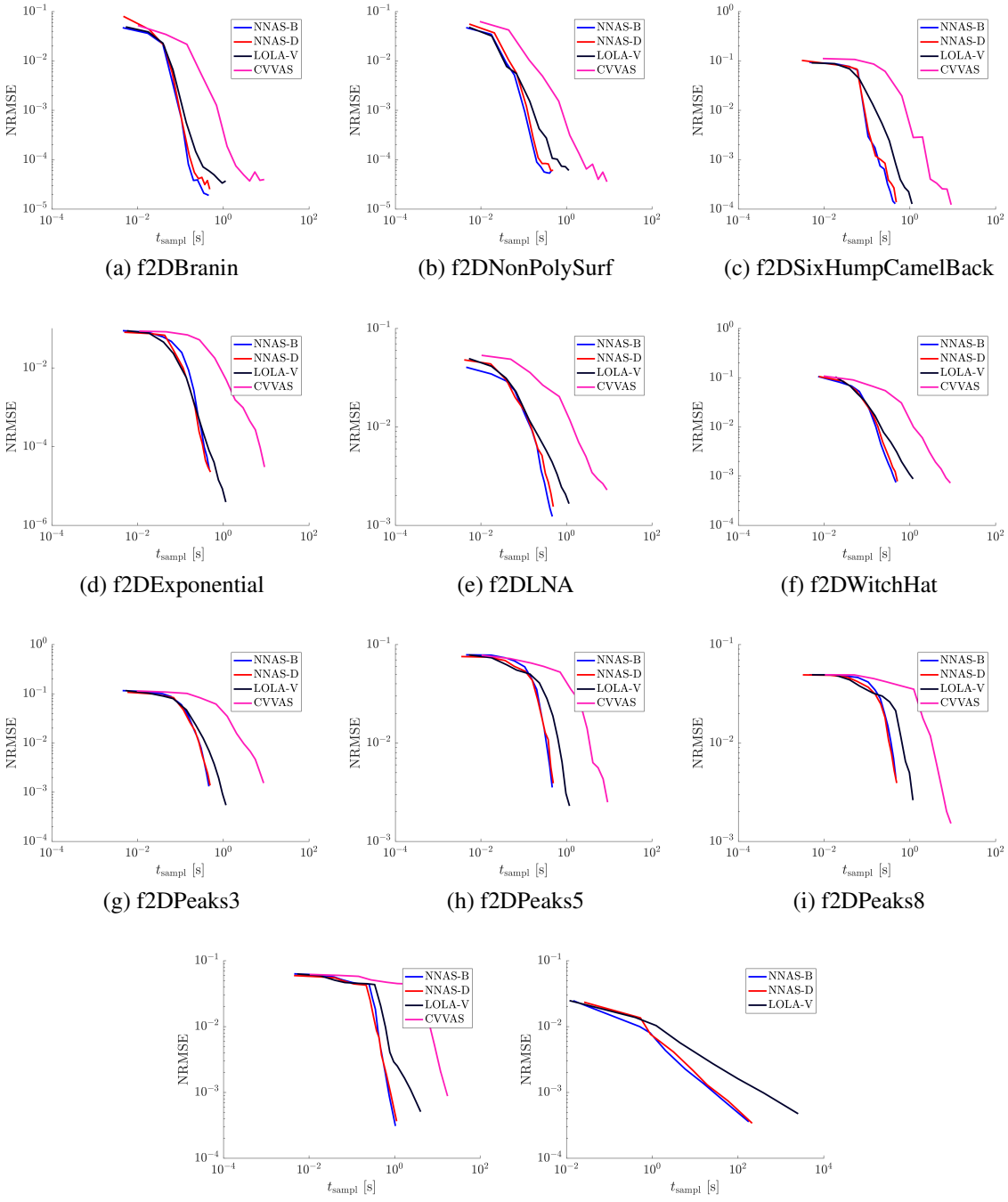


(g) f2DPeaks3 (h) f2DPeaks5 (i) f2DPeaks8



(j) f2DExponential2 (k) f5DXrotor

Figure 6.9: Comparison with respect to n_T



(a) f2DBranin (b) f2DNonPolySurf (c) f2DSixHumpCamelBack

(d) f2DExponential (e) f2DLNA (f) f2DWitchHat

(g) f2DPeaks3 (h) f2DPeaks5 (i) f2DPeaks8

(j) f2DExponential2 (k) f5DXrotor

Figure 6.10: Comparison with respect to t_{sampl}

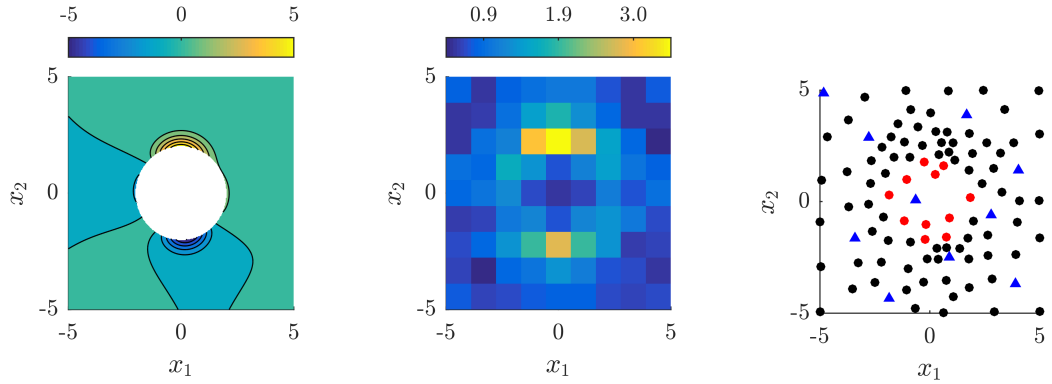
6.6 Sampling in Presence of Solver Critical Errors (Infeasible Region)

This section shows the results obtained by using the NNAS-D formulation presented in section 5.4 for the design space sampling of two two-dimensional responses with infeasible regions. The choice of limiting the analysis on two-dimensional design spaces is necessary for a clear visualization of the resulting sample distributions. The first test case is the analytic f2DPeaks5 function with a circular infeasible region at the center of the domain (figure 6.11a), while the second one is based on XRotor solver in which the infeasibility of the response is caused by impossibility of the software to solve supersonic conditions (figure 6.12a).

Figures 6.11 and 6.12 display the contour plots of the two responses (6.11a and 6.12a), the average sample densities across the 15 repetitions (6.11b and 6.12b), and sample distribution examples where the training points in the infeasible region are marked in red (6.11c and 6.12c). As it possible to see, NNAS-D succeeds in efficiently sampling the design spaces without being affected by the unavailable response values in the infeasible regions. Furthermore, the algorithm appropriately refines the portions of the domain with higher nonlinearity, and it keeps exploring the infeasible region. The continuous exploration of the entire design space is necessary because the algorithm does not know a priori the exact extension of the infeasible region.

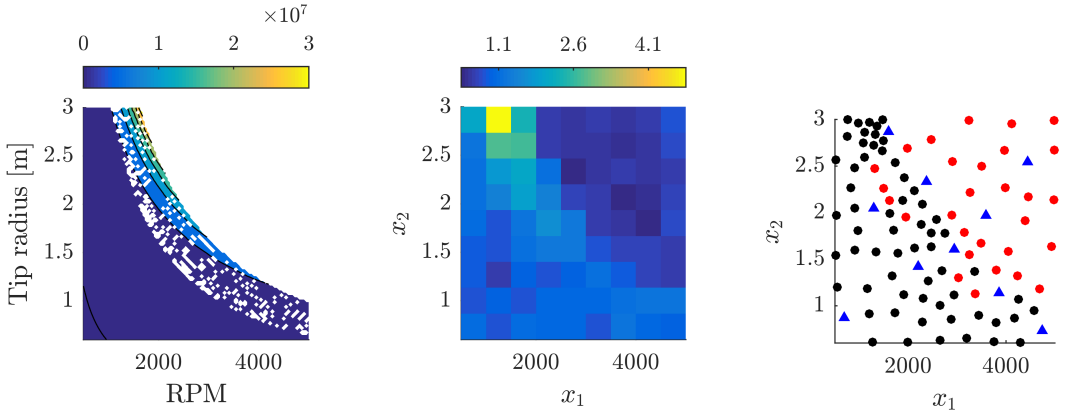
6.7 Multi-response Sampling

This section analyzes the NNAS-D sampling performance when used for multi-response applications (formulation described in section 5.5). Specifically, the two example problems considered are a two-dimensional two-response analytic function (f2DExponential2Split) and a five-dimensional three-response case based on XRotor solver. In both cases, the two responses are assumed equally important, and therefore the sampling algorithm has no priority in enhancing the representation of one response against the other.



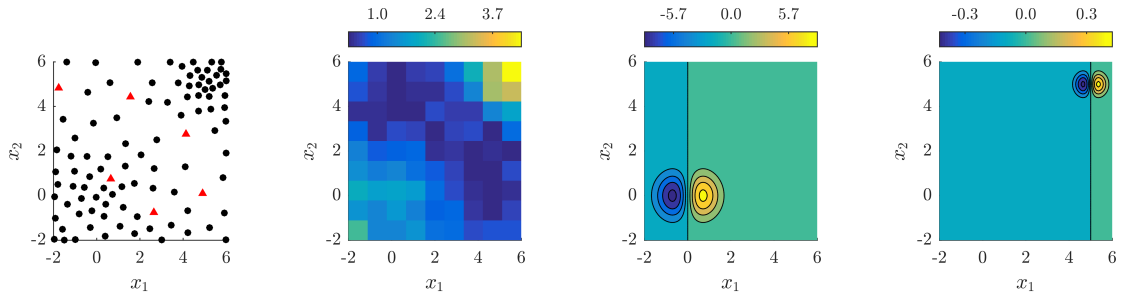
(a) (b) (c)

Figure 6.11: f2DPeaks5 with infeasible region. Response contours (a), average sample density (b), and a resulting sample distribution (c) with initial points marked in blue and training sample in the infeasible region marked in red



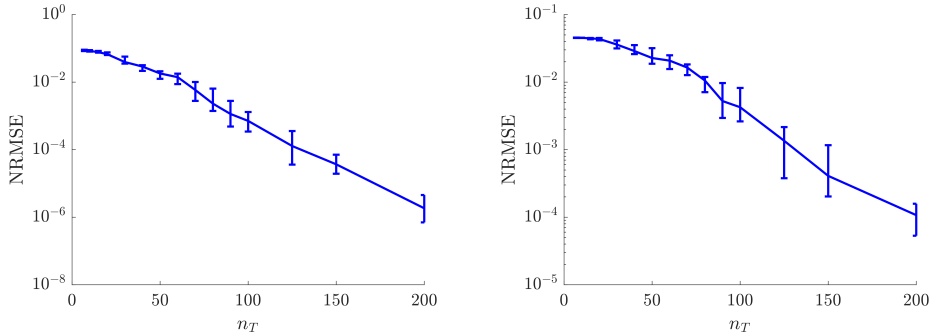
(a) (b) (c)

Figure 6.12: f2DXrotor with infeasible region. Response contours (a), average sample density (b), and a resulting sample distribution (c) with initial points marked in blue and training sample in the infeasible region marked in red



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

Figure 6.13: Multi-response sampling results for f2DExponential2split test function: single sample distribution (6.13a), average sample distribution (6.13b), contour plots of first (6.13c) and second response (6.13d)

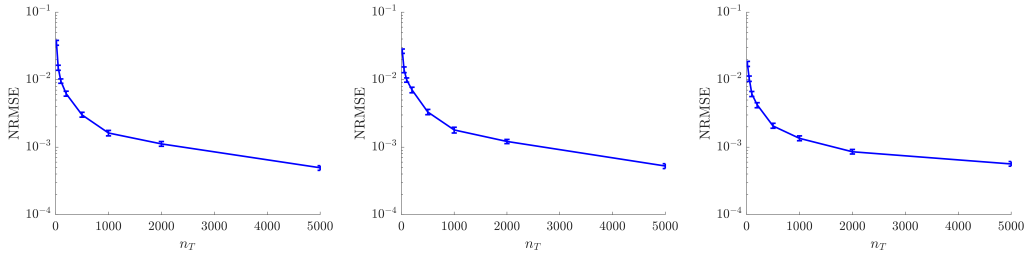


(a) (b)

Figure 6.14: NRMSE evolution for the first (6.14a) and second response (6.14b) of f2DExponential2Split test function.

The sampling results for the f2DExponential2Split in figure 6.13 clearly show how the NNAS multi-response formulation is able to efficiently sample the design space of this two-responses problem, even though the range of two outputs differs by roughly an order of magnitude (Figures 6.13c and 6.13d). The need for the capability of dealing with multi-response problems with a large difference in output ranges is the reason for the refinement metric normalization factor in eq.(5.16).

As additional confirmation of the accurate sampling of the multi-response design space, figures 6.14 and 6.15 display the decreasing NRMSE of response representations for both test cases as samples are sequentially added to D_T by NNAS.



(a) (b) (c)

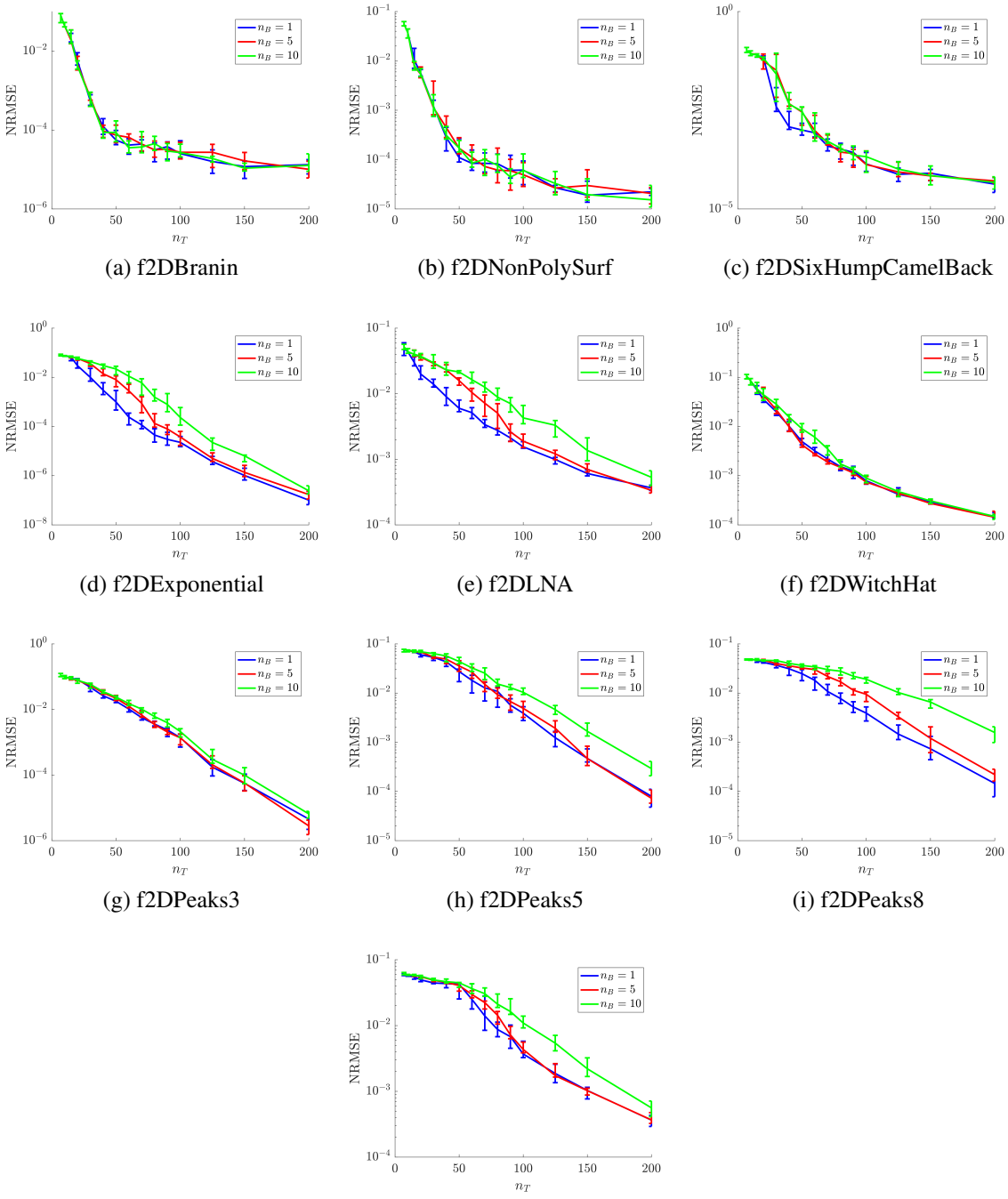
Figure 6.15: NRMSE evolution for the first (6.15a), second (6.15b), and third response (6.15c) of five-dimensional three-response test case based on XRotor solver.

6.8 Batch-mode Sampling

This section analyzes the sampling performance of the NNAS-D batch-mode formulation presented in section 5.6. As described in section 5.6, batch-mode sampling algorithms are extremely valuable because they identify n_B new samples per sampling iteration, thereby allowing the use of nowadays available high performance computing (HPC) architectures. Batch-NNAS-D is used with different batch sizes ($n_B = 5, 10$) for the design space sampling of the two-dimensional example functions, and the results in terms of NRMSE as function of number of samples in the training set (n_T) and number of simulated batches (n_S) are reported in figures 6.16 and 6.17, respectively. The batch results are also compared with those obtained by the one-sample-at-time algorithm ($n_B = 1$). It is important to remember that batch-NNAS waits the complete evaluation of the previous batch of samples before generating the next one because the current formulation does not have the ability to handle situations in which there is a significant difference in evaluation time across the design space.

Figure 6.16 clearly shows that an increase in batch size leads to a degradation in sample usage efficiency as indicated by the lower convergence rate of green and red lines in comparison with the blue line. This performance decay with increase in batch size was expected because the batch size n_B represents the knowledge update rate of the algorithm. In other words, the algorithm has to wait the complete evaluation of n_B points before it

can "refresh" is knowledge about the response, and therefore lower n_B means more frequent knowledge update, and consequentially a better sampling efficiency in terms of n_T. However, the main objective of a batch-mode algorithm is to efficiently leverage HPC architecture by simultaneously running several function evaluations, thereby reducing the total sampling time. Indeed, if the architecture has n_B nodes available, the time cost for the evaluation of n_B designs is equivalent to the time required to evaluate a single design in a single node architecture. Figure 6.17 shows the same results of figure 6.16 as a function of evaluated batch n_S. If we assume that the evaluation time for each batch is constant, the lines in figure 6.17 also represent the NRMSE behavior as function of process time. These results clearly show the remarkable reduction in simulation time that is achievable by using the batch-mode NNAS algorithm coupled with an HPC architecture: the decrease in sample usage efficiency in figure 6.16 is dominated by the reduction in simulation time due to the simultaneous evaluation of n_B designs.



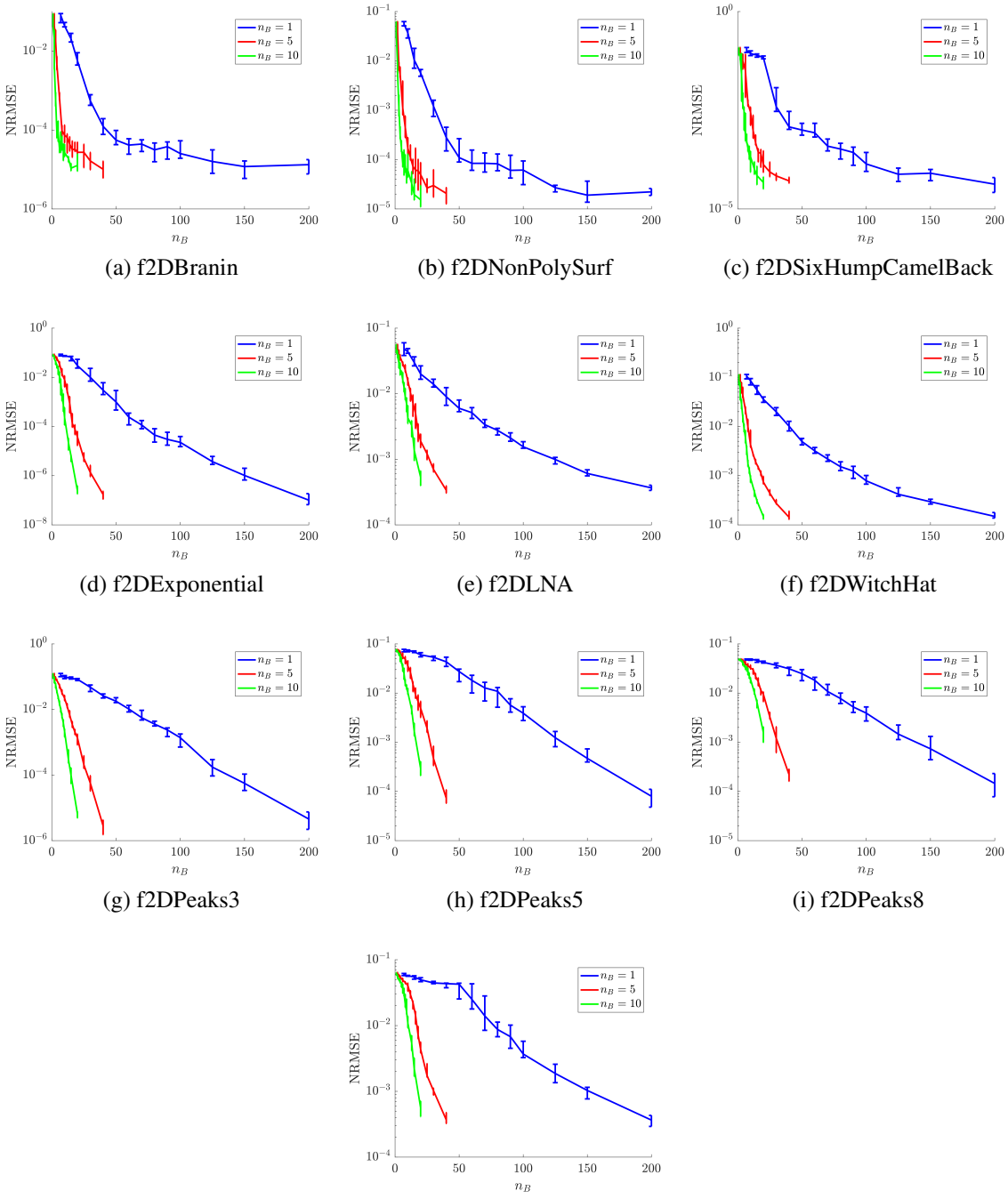
(a) f2DBranin (b) f2DNonPolySurf (c) f2DSixHumpCamelBack

(d) f2DExponential (e) f2DLNA (f) f2DWitchHat

(g) f2DPeaks3 (h) f2DPeaks5 (i) f2DPeaks8

(j) f2DExponential2

Figure 6.16: Comparison of batch sampling performance with respect to n_T



(a) f2DBranin (b) f2DNonPolySurf (c) f2DSixHumpCamelBack

(d) f2DExponential (e) f2DLNA (f) f2DWitchHat

(g) f2DPeaks3 (h) f2DPeaks5 (i) f2DPeaks8

(j) f2DExponential2

Figure 6.17: Comparison of batch sampling performance with respect to n_T

CHAPTER 7
CONCLUSIONS

This dissertation presented and tested a new class of model independent sequential adaptive strategies (MISASs) for early phases of engineering design, named Nearest Neighbors Adaptive Sampling (NNAS). During the last decades, the increase in popularity of surrogate models (SMs) for engineering application encouraged the development of more advanced and efficient surrogate modeling techniques; in particular, several studies underlined how an accurate sampling phase is crucial to obtain SMs with a high representation quality. Indeed, a poor distribution of training samples across the design space may lead to a complete misinterpretation of the response that has to be modeled. The discussion about the sampling phase for early phases of engineering design provided in chapter 1 resulted in a list of requirements for effective sampling strategies:

- be *adaptive* to guide the sampling behavior based on the response information initially available or collected during the process;
- be *sequential* to permit in-the-loop SM quality checks and the suspension of the sampling process whenever a certain accuracy level is reached;
- be *flexible* to be applicable to a wide range of problems;
- be *simple and robust* to be implementable by users with limited know-how in technical statistics and numerical methods;
- have *low computationally* complexity and good scaling behavior with respect to the problem dimensionality to make the method usable in high-dimensional engineering design problems;

- be *model independent* to remove the impact of the particular SM choice on the sampling efficiency and consequentially on the surrogate modeling representation quality.
- include a *batch-mode* formulation to efficiently use HPC resources;
- be able to deal with *multi-response* problems;
- have the ability to handle *critical errors* in the simulation code;
- be able to leverage *derivative information* which may be returned by the simulation code.

The extensive literature review of available sampling techniques (chapter 2) revealed how none of those is able to fulfill all the requirements listed above, thereby motivating the development of a new sampling technique which is the main objective of this dissertation.

As discussed in section 2.1.3, the reason for model independence requirement is the hypothesis that sampling performance of model dependent sequential adaptive strategies (MDSASs) may deteriorate if the chosen supervising SM formulation is inaccurate to describe the response that has to be modeled. However, the literature about MDSAS lacks of a comprehensive study regarding the effect of model dependency in the sampling performance of such techniques. This motivates the first objective of this thesis that corresponds to the analysis presented in chapter 4 in which an MDSAS was used for the design space sampling of several example functions considering different SM formulations. The results (section 4.3.2) validated the hypothesis showing that the sampling performance enhancement expected by the use of an MDSAS technique may vanish if an inappropriate SM is chosen at the beginning of the process. Therefore, we proposed a new sampling architecture in which the MDSAS is coupled with an active SM selection. In this way, the algorithm actively chooses the SM formulation to use in the sampling phase based on its representation accuracy of the available training dataset. The analysis of sample distributions obtained with the new architecture (section 4.3.3) showed how coupling an MDSAS

with an active SM selection criterion drastically improves the sampling performance and robustness of the overall surrogate modeling process, and it relieves the user from selecting the SM formulation at the beginning of the process when there is usually limited knowledge about the function to model. Even though the proposed architecture is able to improve the sampling performance of MDSASs, the need of those techniques to retrain the supervising SM every time new information about the response becomes available results in an excessive computational time, particularly for high dimensional applications. Furthermore, any robustness or convergence issues, or other shortcomings affecting the supervising SM formulation are inherited by the sampling algorithm. These aspects are the reasons why "model independent" is one of the requirement for sampling techniques for engineering applications.

The second objective of this dissertation was the development of a MISAS based on local linear model, named basic NNAS (NNAS-B). The hypothesis was that the usage of local linear model for the computation of the refinement metric efficiently spreads samples across the domain accordingly to the response nonlinearity, resulting in a model independent technique with a lower computational complexity in comparison with other state-of-the-art MISAS. This goal has been achieved by the definition of a new refinement metric that uses local hyperplanes to estimate the level of nonlinearity in the neighborhoods of training samples. The application of hyperplanes as local approximations of the response limits the complexity of the sampling algorithm (section 5.7) and it increases the ease of strategy implementation (section 5.2.4). To assess NNAS-B sampling efficiency, NNAS-B has been tested for the design space sampling of several example functions with different level of feature complexity. In particular, the analytic test functions have been selected among the most popular ones in the surrogate modeling research field, while the engineering test case is based on the well-known XRotor software which estimates propeller performance. The comprehensive set of results reported in chapter 6 showed how NNAS-B has sampling performance similar to other state-of-the-art techniques, but with a lower

computational cost thanks to the lower algorithm complexity. These results validate the hypothesis that the proposed adaptive sampling technique based on local linear models is model independent, robust, computationally efficient and ease to implement thanks to the simple operations involved in the algorithm.

The third objective of this dissertation asked for a further development of the basic NNAS to include a directional sampling criterion that was supposed to improve the sampling performance. This goal has been achieved by a simple modification of the neighborhood and hyperplane definitions used in the refinement metric evaluation, without affecting the overall algorithm complexity. More specifically, the sampling phase in directional NNAS (NNAS-D) is locally guided by the direction of the max hyperplane prediction error. The opportunity to obtain sampling directionality by using an SM trained on the neighborhood prediction errors has not been considered because it would have led to a model dependent technique. Results (chapter 6) showed that the directional sampling criterion in NNAS-D did not lead to the expected sampling performance enhancement in comparison with NNAS-B, except for a limited number of cases. However, the NNAS-D formulation makes it possible to directly use the response derivative information which may be returned by the solver for the hyperplane definitions, thereby removing the need of solving the linear system in eq.(5.23).

Both NNAS formulations introduced also a new approach to achieve the refinement-exploration balance. As described in section 2.1.1, an efficient sampling technique should both explore the design space to detect as many response features as possible, and refine the sample distribution in portions of the design space where particular features are detected. While most state-of-the-art techniques achieve the refinement-exploration balance either by forming a convex sum of the two metrics or by combining the refinement metric with a distance penalty function, NNAS techniques approach the refinement-exploration balance as a multiobjective problem: the sampling algorithm should place the samples in locations that maximize both the exploration and the refinement metrics. Specifically, NNAS achieves the

exploration-refinement balance by introducing a stochastic Pareto-ranking based selection criterion in which the Pareto-ranking assures to identify the design space regions equally optimal for the maximization of both exploration and refinement, while the stochastic flavor mitigates any inductive bias which may be caused by an excessive confidence on the collected data. Results in chapter 6 confirm the ability of the proposed stochastic Pareto-ranking selection criterion to efficiently balance the exploration and refinement behaviors of the sampling algorithm.

Regarding the computational complexity of the proposed NNAS algorithms, section 5.7 proved that both have a quasilinear computational complexity with respect to the number of samples n_T ($\mathcal{O}(n_T \log_2(n_T))$) which is lower than the complexity of other adaptive sampling techniques like CVVAS ($\mathcal{O}(n_T^n)$ with $n \geq 3$) and LOLA-V ($\mathcal{O}(n_T^2)$). Therefore NNAS formulations meet the requirement of low computational complexity, at least in comparison with other state of the art adaptive sequential techniques.

A further development of NNAS techniques included the batch operational mode, a formulation for of multi-response applications, and the ability to deal with solver critical errors, that are the fourth, fifth, and sixth dissertation objectives, respectively. As illustrated in the results sections from 6.6 to 6.8, the final version of NNAS algorithms is able to efficiently sample design spaces in all these three situations.

Overall, results showed that the proposed NNAS class of sampling techniques is able to successfully fulfill all the requirements for early engineering design phase sampling technique listed at the beginning of this chapter. It is opinion of the author that the reduced computational complexity, together with the ability of operating in batch-mode and being unaffected by critical solver errors, make NNAS a valuable tool for high-dimensional SM applications in early phase of engineering design. Summarizing, this dissertation provides three main contributions to the research community: a comprehensive study about model dependence effect on sampling efficiency, a new class of MISAS suitable for SM appli-

cations in early engineering design phase, and a new approach to achieve an exploration-refinement balance.

The research presented in this dissertation leaves open several starting points for future development. First, a "more accurate approximation" of the Voronoi tessellation should be included to remove the Monte-Carlo approach currently used in NNAS to estimate the Voronoi cell volumes and corners. The use of a complete approximate Voronoi tessellation would open the opportunity for a better definition of the neighborhoods, with expected benefit in the overall sampling efficiency. Second, the NNAS techniques should be expanded to incorporate response uncertainties, thereby making possible to apply NNAS also for real experiment applications. Such a technique will be extremely valuable for hardware-in-the-loop application like highly automated wind tunnel tests. Third, the usage of local linear models to compute response approximation together with the stochastic Pareto-ranking criterion to achieve exploration-refinement balance can be extended for optimization purposes, thereby creating a model independent global optimization algorithm useful for problem with high dimensionality and/or limited function evaluation time. If coupled with the ability of handling response uncertainties, such a technique will be a powerful optimization tool for hardware-in-the-loop applications.

Appendices

APPENDIX A
SURROGATE MODEL QUALITY ASSESSMENT

This appendix illustrates the formulation of the GEE and LEE used in this dissertation.

A.1 Global Error Estimators

Conventional strategies to assess the global quality of a SM are cross-validation (CV) errors [13], jackknife and bootstrap [26], Akaike's information criterion (AIC) [12], and R^2([61, 71, 87]); the two metrics considered in this dissertation are CV errors and R^2.

CV is a common technique that investigates the SM quality using only the data available in the training dataset D_T, without the need of a separate validation dataset D_V. As described in [69, 71], there are two prevalent CV strategies: the k-fold CV and the leave-p-out CV. Both approaches firstly create certain subsets of D_T, and then they train all the SMs obtainable by leaving out one subset each time from the training set. The subset that is not included in the training set is then used as validation data set to compute the CV error, as described in [105]. The difference between k-fold and leave-p-out is the way the subsets are created; whereas the k-fold randomly splits D_T in k roughly equal subsets, leave-p-out considers all the possible subsets obtainable leaving p points out. Obviously, the leave-p-out approach is more computationally expensive than the k-fold, and several authors ([13, 61, 71, 105, 113]) suggest the use of $p = 1$ to reduce the computational time. The special case of $p = 1$ is called *leave-one-out CV* (LOO-CV), which is the cross validation approach considered in this study.

Once all the SMs required by the selected CV strategy are trained (n_T in the case of LOO-CV), it is possible to evaluate various GEEs. The $\mathrm{RMSE}_{\mathrm{CV}}$, also called $\mathrm{PRESS}_{\mathrm{RMS}}$

[71], $\mathrm{AVE}_{\mathrm{RMSE_{CV}}}$ [69], or σ_{PRESS} [33] is defined as

$$\mathrm{RMSE_{CV}} = \sqrt{\frac{1}{n_T}\sum_{i=1}^{n_T}\left(y_{T,i} - \tilde{y}_i^{(-i)}\right)^2} \tag{A.1}$$

where $\tilde{y}_i^{(-i)} = \tilde{f}^{(-1)}(\mathbf{x}_{T,i})$ and $\tilde{f}^{(-1)}(\cdot)$ is the SM trained leaving the ith point out. Liu *et al.*[61] define also the normalized $\mathrm{RMSE_{CV}}$:

$$\mathrm{NRMSE_{CV}} = \frac{\mathrm{RMSE_{CV}}}{\max(\mathbf{y}_T) - \min(\mathbf{y}_T)} \tag{A.2}$$

Another possible CV GEE is the CV maximum absolute error ($\mathrm{MAE_{CV}}$) [71] in absolute (A.3) and normalized version (A.4):

$$\mathrm{MAE_{CV}} = \max|y_{T,i} - \tilde{y}_i^{(-i)}| \qquad i = 1 \dots n_T \tag{A.3}$$

$$\mathrm{NMAE_{CV}} = \frac{\mathrm{MAE_{CV}}}{\max(\mathbf{y}_T) - \min(\mathbf{y}_T)} \tag{A.4}$$

Considering methods that require additional validation data D_V, it is also possible to define the corresponding versions of the metrics introduced above for cross validation:

$$\mathrm{RMSE}_V = \sqrt{\frac{1}{n_V}\sum_{i=1}^{n_V}\left(f(\mathbf{x}_{V,i}) - \tilde{f}(\mathbf{x}_{V,i})\right)^2} \tag{A.5}$$

$$\mathrm{NRMSE}_V = \frac{\mathrm{RMSE}_V}{\max(\mathbf{y})_T - \min(\mathbf{y})_T} \tag{A.6}$$

$$\mathrm{MAE}_V = \max|f(\mathbf{x}_{V,i}) - \tilde{f}(\mathbf{x}_{V,i})| \qquad i = 1 \dots n_V \tag{A.7}$$

$$\mathrm{NMAE}_V = \frac{\mathrm{MAE}_V}{\max(\mathbf{y})_T - \min(\mathbf{y})_T} \tag{A.8}$$

Another common GEE requiring validation data is R-square R^2 ([46]):

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n_V}(f(\mathbf{x}_{V,i}) - \tilde{f}(\mathbf{x}_{V,i}))^2}{\sum_{i=1}^{n_V}(f(\mathbf{x}_{V,i}) - \bar{y}_V)^2} \tag{A.9}$$

where $\bar{y}_V$ is the mean value of the true function f evaluated at validation points.

Except for R^2that indicates a good SM when it approaches 1 as it described by Equation (A.9), a GEE value close to 0 is desirable as accurate SM indicator, as it is possible to infer from Equation (A.1) to (A.8).

A.2 Local Error Estimators

Local error estimators provide metrics to assess the SM quality at specific locations in the design space. They are commonly used to identify domain regions where the SM has a low accuracy and additional training points are required for a SM refinement (section 2.1).

The relative absolute error (RAE) described in [71] can be defined either when the validation data is available ($\mathrm{RAE_V}$) or when a CV approach is used ($\mathrm{RAE_{CV}}$) ([113]):

$$\mathrm{RAE}_{V,i} = \left| \frac{f(\mathbf{x}_{V,i}) - \tilde{f}(\mathbf{x}_{V,i})}{f(\mathbf{x}_{V,i})} \right| \tag{A.10}$$

$$\mathrm{RAE}_{\mathrm{CV},i} = \left| \frac{f(\mathbf{x}_{T,i}) - \tilde{f}(\mathbf{x}_{T,i})^{(-i)}}{f(\mathbf{x}_{T,i})} \right| \tag{A.11}$$

$$\tag{A.12}$$

CV-variance (σ^2_{CV}) is another LEE inspired by [16, 53]. The σ^2_{CV} value at any location $\mathbf{x}$ in the design space is the variance of the CV SM approximations $\tilde{f}(\mathbf{x})^{(-i)}$ as described in Equation (A.13):

$$\sigma^2_{\mathrm{CV}}(\mathbf{x}) = \frac{1}{n_T} \sum_{i=1}^{n_T} \left(\tilde{f}(\mathbf{x})^{(-i)} - \tilde{f}(\mathbf{x}) \right)^2 \tag{A.13}$$

An example of σ^2_{CV} evaluation is illustrated in Figures A.1 and A.2.

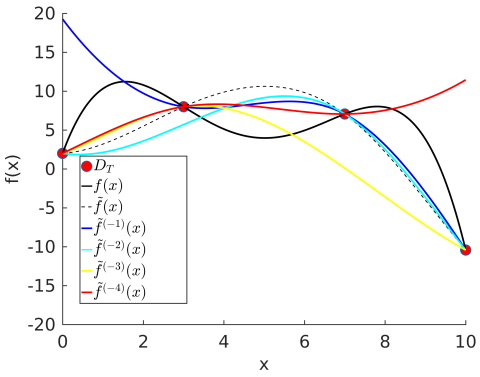


Figure A.1: Example of SM involved in the CV process $\tilde{f}^{(-i)}$, complete SM $\tilde{f}$ and true function f

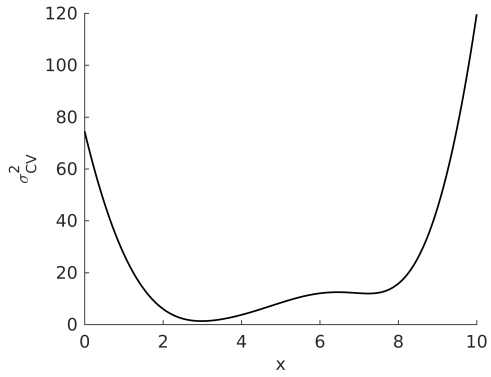


Figure A.2: Resulting σ^2_{CV} for the example reported in figure A.1

APPENDIX B

SM FUNCTIONAL FORM FORMULATIONS

This appendix provides the mathematical formulations of the SM functional forms considered in the dissertation: radial basis function (RBF) SMs for the linear class, and a Kriging (KRG) SM for the non-linear class. A comprehensive survey of available SM techniques is out of the scope of this appendix, and overviews of most common SMs used in engineering applications are available in [46, 87, 102].

B.1 Radial Basis Functions

The RBF models used in this study have 4 different kernel functions which are listed in table B.1, and they consider a single basis function $b(\tilde{r})$ and a constant mean value ([25, 43, 47, 73]):

$$\tilde{f}(\mathbf{x}) = \mu + \sum_{i=1}^{n_T} w_i b\left(\tilde{r}(\mathbf{x}, \mathbf{x}_{T,i})\right) \tag{B.1}$$

$$= \mu + \mathbf{b}^T \mathbf{w} \tag{B.2}$$

where n_T is the number of samples in the training dataset D_T, and

$$\mu = \text{mean}(\mathbf{y}_T) \tag{B.3}$$

The training process required to find the model coefficients w_i involves the solution of a linear system derived by the fact that $\tilde{f}(\mathbf{X}_T) = \mathbf{y}_T$. The linear system equation is:

$$\mathbf{w} = \mathbf{B}^{-1}(\mathbf{y}_T - \mu) \tag{B.4}$$

Table B.1: Radial Basis Functions considered, [73]

Name	Expression
Multiquadric	$(1+(r/r_0))^{1/2}$
Inverse Multiquadric	$(1+(r/r_0))^{-1/2}$
Gaussian	$\exp(-(r/r_0)^2)$
Cubic	$1+(r/r_0)^3$
Thin Plate Spline	$(r/r_0)^2\log(r/r_0)$

where $\mathbf{B}$ is the square symmetric matrix defined as:

$$\mathbf{B} = \begin{pmatrix} b(\tilde{r}(\mathbf{x}_{T,1},\mathbf{x}_{T,1})) & \cdots & b(\tilde{r}(\mathbf{x}_{T,1},\mathbf{x}_{T,n_T})) \\ \vdots & \ddots & \vdots \\ b(\tilde{r}(\mathbf{x}_{T,n_T},\mathbf{x}_{T,1})) & \cdots & b(\tilde{r}(\mathbf{x}_{T,n_T},\mathbf{x}_{T,n_T})) \end{pmatrix} \tag{B.5}$$

and the weighted radius $\tilde{r}$ between generic points $\mathbf{x}_A$ and $\mathbf{x}_B$ is:

$$\tilde{r}(\mathbf{x}_A,\mathbf{x}_B) = \sqrt{\sum_{i=1}^{D}\left(\frac{x_{A,i}-x_{B,i}}{\theta_i}\right)^2} \tag{B.6}$$

The tuning coefficients θ_i are found during the SM training process by the minimization of the NRMSE. The minimization process is conducted using a line search algorithm (`fmincon` Matlab routine) started from several different initial points in the θ domain. The number of θ initial points has been set equal to three times the problem dimensionality, and similarly to what was done with $D_{T,0}$ and D_V, the initial θ values are kept fixed for cases with the same dimensionality.

B.2 Kriging Surrogate Models

The ordinary Kriging (OKRG) formulation adopted in this dissertation is based on [89], but equivalent formulations are available in [33, 67, 87, 96, 98]. OKRG emulates the response

$f(\mathbf{x})$ as a sum of a constant value μ and a departure term $Z(\mathbf{x})$ which is a stochastic process built with a distance-based correlation function. The model expectation and variance at several locations $X_* = [\mathbf{x}_{*,1}; \ldots; \mathbf{x}_{*,n}]$ – representing the function prediction $\tilde{f}(X_*)$ and a measure for the local accuracy estimation respectively – can be computed as:

$$\begin{aligned} \tilde{y}(X_*) &= \mu + \mathbb{E}(Z(\mathbf{x})) \\ &= \mu + K(X_*, X_T)K(X_T, X_T)^{-1}(\mathbf{y}_T - \mu) \end{aligned} \tag{B.7}$$

$$\begin{aligned} \sigma^2(X_*) =& \text{var}(\check{f}(X_*)) \\ =& \text{diag}(K(X_*, X_*) \\ & - K(X_*, X_T)K(X_T, X_T)^{-1}K(X_T, X_*)) \end{aligned} \tag{B.8}$$

where $K(X_i, X_j)$ is the covariance matrix between two sets of sample locations X_i and X_j, and μ is equal to the mean of the training response values (B.3).

The choice of the correlation function $k(\mathbf{x}_A, \mathbf{x}_B)$ required to fully define a KRG model has been proven to have a sensible impact on the resulting KRG model [24, 84, 89, 113, 114]. Although a proper choice of correlation function is crucial for the accurate modeling of $f(\mathbf{x})$ via a KRG model, a comprehensive discussion about this topic departs from the scope of this dissertation, and detailed investigations are available in [24, 67, 84, 89, 113, 114]. The correlation function considered in this study is the widely used "Sum Squared Exponential" (SSE) function ([67, 75, 89]):

$$k(\mathbf{x}_A, \mathbf{x}_B) = \tilde{\sigma}^2_{\text{KRG}} \exp\left(- \sum_{i=1}^{D} \left(\frac{x_{A,i} - x_{B,i}}{\theta_i} \right)^2 \right) \tag{B.9}$$

where $[\tilde{\sigma}^2_{\text{KRG}}, \theta_i]$ are the model *hyperparameters* which represent the process variance and the length scales of the model variations along each dimension ([89]). The estimation of the hyperparameter values is done during the KRG training process, which is typically accomplished by the maximization of the likelihood as described in [67, 84, 89]. Equations

(B.7) to (B.9) represent the formulation of Ordinary Kriging Models (OKRG) used in this study.

APPENDIX C

ALGORITHM FOR APPROXIMATE VORONOI CORNER IDENTIFICATION

This appendix provides the description of the strategy for the approximate identification of Voronoi corners; as described in sections 5.3 and 5.6, these geometrical details are required by NNAS-D and NNAS batch-mode formulations.

Let's assume that we would like to approximately identify the corners of the VC of a generic point $\mathbf{x}^*$ (VC^*) belonging to a D-dimensional dataset D_A. The approximate procedure described in this section returns the subset of randomly generated points that are located "enough" close to the VC corners. The algorithm proceeds as follow:

1. creation of n_R random points within an hypercube centered in $\mathbf{x}^*$ and edge equal to two times a length L that approximately represents the distance between $\mathbf{x}^*$ and its neighbors (e.g. the exploration metric E_{i^*} for NNAS-B and NNAS-D if a 0-1 normalized design space is considered). As a rule of thumb, n_R should be proportional to the dimensionality of the function, e.g. $n_R = cD$.

2. Identification of the $D + 1$ nearest samples in D_A for each random points. This process can be efficiently completed by using k-nearest-neighbors search (kNN-S) algorithm.

3. Identification of random points that are "enough" close to the VC^* shell. A VC shell is composed by the $(D - 1)$-dimensional hyperplanes that define the boundaries of the VC (figure C.1a). First of all, we filter out all the random points that does not have have $\mathbf{x}^*$ as one of the first two closest neighbors (figure C.1b). Then we identify the points that have the normalized value of difference of distances between them and the first two neighbors in D_A smaller than a given threshold (figure C.1c). More

formally:

$$\frac{|d(\mathbf{x}_{\mathrm{R,i}}, \mathbf{x}_{\mathrm{cl},1}) - d(\mathbf{x}_{\mathrm{R,i}}, \mathbf{x}_{\mathrm{cl},2})|}{L} < c_{\mathrm{shell}} \tag{C.1}$$

where d is the euclidean distance operator, $\mathbf{x}_{\mathrm{R,i}}$ is the ith random point, and the $\mathrm{cl, j}$ indicates the jth closest sample of $\mathbf{x}_{\mathrm{R,i}}$ in D_A. To these points, it is necessary to add the random points that eventually are close to the external shell portion represented by design space bounds. These points are identified by considering the random points that have $\mathbf{x}^*$ as closest D_A neighbor and have a distance from whatever domain bound less than a given threshold (it is possible to use the same value used for eq.(C.1)).

4. Identification of shell random points that are close to Voronoi corners. Those that are close to an internal corner have the normalized maximum absolute value of the difference in distance between them and the consecutive $(D+1)$ D_A neighbors less that a given threshold (figure C.1d), i.e.

$$\frac{\max(|d(\mathbf{x}_{R,i}, \mathbf{x}_{\mathrm{cl},1}) - d(\mathbf{x}_{R,i}, \mathbf{x}_{\mathrm{cl},2})|, \ldots, |d(\mathbf{x}_{R,i}, \mathbf{x}_{\mathrm{cl},D}) - d(\mathbf{x}_{R,i}, \mathbf{x}_{\mathrm{cl},D+1})|}{L} < c_{\mathrm{corners}} \tag{C.2}$$

The corner regions that are close to an external corner (i.e. that rests on one domain bound) are instead found using the eq.(C.2), but considering only the $(D+1-n_{\mathrm{bnd}})$ neighbors, where n_{bnd} is the number of domain bounds where $\mathbf{x}_{R,i}$ is closest to.

The described procedure returns a set of points that are located in the neighborhood of Voronoi corners, and the index of D_A points that rest on the approximate corners. The two-dimensional procedure is illustrated in figure C.1.

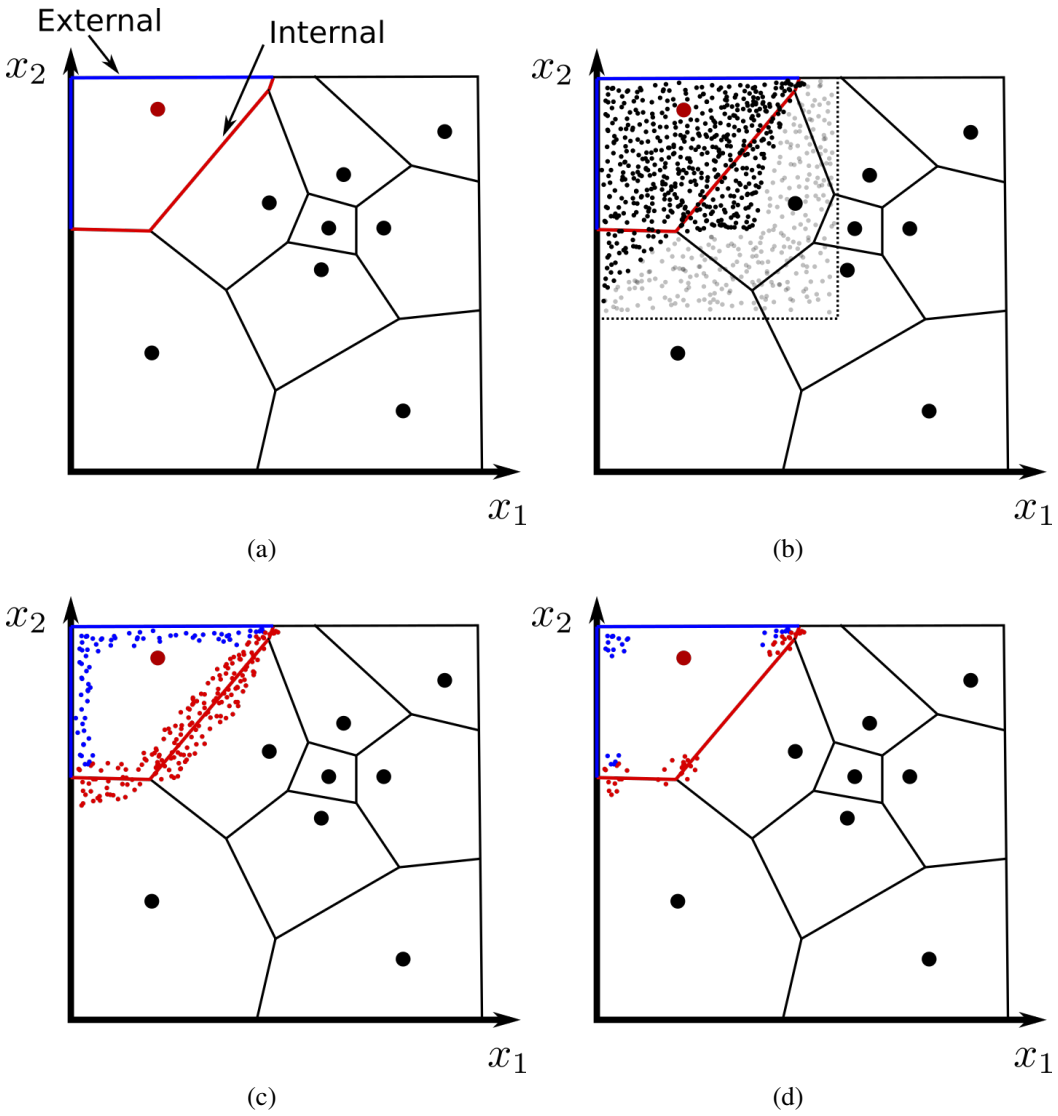


Figure C.1: Example of steps in the procedure to identify approximate Voronoi corners. Figure C.1a shows the external (blu) and internal (red) portions of the Voronoi shell of the cell "owned" by the red dot. Figure C.1b displays the hypercube delimiting the region in which the random points are created. The points that have the red dot as one of the two closest neighbors are in black. Figure C.1c plots the subset of random points that are close "enough" to the Voronoi shell. Finally, figure C.1d shows the random points that have been selected as representative of the Voronoi corners.

APPENDIX D

TEST FUNCTIONS

This section provides the formulations and the ranges of the test function used in this study (table D.1).

Table D.1: List of test functions with relative dimensionality and references

Function name	Dimensionality (D)	Number of responses (n_f)	References
f2DBranin	2	1	[3, 116]
f2DPeaks3	2	1	[20, 31]
f2DPeaks5	2	1	[20, 31]
f2DPeaks8	2	1	[20, 31]
f2DExponential	2	1	[16]
f2DExponential2	2	1	New
f2DNonPolySurf	2	1	[16, 70]
f2DSixHumpCamelBack	2	1	[16, 70]
f2DLNA	2	1	[38, 55]
f2DWitchHat	2	1	[31]
f2DExponential2Split	2	2	New
f5DXrotor	5	1	New

D.1 Branin Function

Eq.(D.1) is a common test function used as optimization algorithm test case; more information is available in [3]. This function is used as test case in [71, 116].

$$f(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10 \qquad \text{(D.1)}$$

The ranges of the two design variables are listed in table D.2.

Table D.2: List of f2DBranin design variable and their ranges

Design variable	Meaning	Ranges
x_1	First design variable	$[-5, 10]$
x_2	Second design variable	$[0, 15]$

D.2 Peaks Function

The Peaks function is available as the built-in MATLAB function `Peaks`, and it is used as test case in [20]. This function is obtained by translating and scaling Gaussian distributions. The same formulation is used for f2DPeaks3, f2DPeaks5, f2DPeaks8 by changing the ranges of the design variables as indicated in table D.3.

Table D.3: List of f2DPeaks design variable and their ranges

Design variable	Meaning	Ranges
x_1	First design variable	$[-5, 5]$
x_2	Second design variable	$[-5, 5]$

D.3 Exponential Function

$$f(x_1, x_2) = x_1 \exp(-x_1^2 - x_2^2) \tag{D.2}$$

The ranges of the two design variables are listed in table D.4.

D.4 Exponential2 Function

$$f(x_1, x_2) = x_1 \exp(-x_1^2 - x_2^2) + 2\left(\tilde{x}_1 \exp(-\tilde{x}_1^2 - \tilde{x}_2^2)\right) \tag{D.3}$$

Table D.4: List of f2DExponential design variables and their ranges

Design variable	Meaning	Ranges
x_1	First design variable	$[-2, 6]$
x_2	Second design variable	$[-2, 6]$

where

$$\tilde{x}_1 = 2\left(x_1 - 5\right) \tag{D.4}$$

$$\tilde{x}_2 = 2\left(x_2 - 5\right) \tag{D.5}$$

The ranges of the two design variables are the same for f2DExponential function and they are listed in table D.4.

D.5 Non Polynomial Surface Function

$$f(x_1, x_2) = \frac{\left(30 + 5x_1 \sin\left(5x_1\right)\right)\left(4 + \exp\left(-5x_2\right)\right) - 100}{6} \tag{D.6}$$

The ranges of the two design variables are listed in table D.5.

Table D.5: List of f2DNonPolySurf design variables and their ranges

Design variable	Meaning	Ranges
x_1	First design variable	$[0, 1]$
x_2	Second design variable	$[0, 1]$

D.6 Six-Hump Camel-Back Function

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right) x_2^2 \tag{D.7}$$

The ranges of the two design variables are listed in table D.6.

Table D.6: List of f2DSixHumpCamelBack design variables and their ranges

Design variable	Meaning	Ranges
x_1	First design variable	$[-2, 2]$
x_2	Second design variable	$[-1, 1]$

D.7 LNA Function

Eq.(D.8) describes the admittance of a Low Noise Amplifier model; more information is available in [55]. This function is used as test case in [38]

$$y_{12} \cong \frac{g_m}{1 - \omega^2 C_{gs}(L_s + L_m) + j\omega L_s g_m} \tag{D.8}$$

$$W = 100 \cdot 10^{-6} \cdot 10^{\overline{W}} \tag{D.9}$$

$$L_s = 0.5 \cdot 10^{-9} \cdot 10^{\overline{L_s}} \tag{D.10}$$

$$f = (11 + 10\overline{f}) \cdot 10^{9} \tag{D.11}$$

$$L = (90 + 30\overline{L}) \cdot 10^{-9} \tag{D.12}$$

$$V_{GT} = 0.275 + 0.2\overline{V}_{GT} \tag{D.13}$$

$$L_m = 1 \cdot 10^{-9} \cdot 10^{\overline{L}_m} \tag{D.14}$$

$$\omega = 2\pi f \tag{D.15}$$

$$g_m = 1 \cdot 10^{-4} \frac{W}{L} V_{GT} \tag{D.16}$$

$$C_{gs} = 0.01 \cdot WL \tag{D.17}$$

The output considered is $|y_{12}|$ as function of $\overline{W}$ and $\overline{L}_s$ for the f2DLNA and of $\overline{W}$, $\overline{L}_s$, $\overline{f}$, $\overline{L}$ and $\overline{V}_{GT}$ for the f5DLNA. The meaning of the variables and the constant parameters are listed in table D.7.

Table D.7: List of f2DLNA and f5DLNA design variables and their ranges

Design variable	Meaning	Ranges f2DLNA	Ranges f5DLNA
$\overline{W}$	Normalized width	$[-1, 1]$	$[-1, 1]$
$\overline{L}_s$	Normalized inductance	$[-1, 1]$	$[-1, 1]$
$\overline{f}$	Normalized frequency	1	$[-1, 1]$
$\overline{L}$	Normalized length	1	$[-1, 1]$
$\overline{V}_{GT}$	Normalized voltage	0	$[-1, 1]$
$\overline{L}_m$	Normalized inductance	0	0

D.8 Witch Hat Function

$$
\begin{aligned}
f(x_1, x_2) =& 0.5 \sin\left(-\frac{\pi}{2} + \frac{\pi}{\sqrt{2}}\left(\cos\left(\frac{\pi}{4}\right) x_1 + \sin\left(\frac{\pi}{4}\right) x_2\right)\right) \\
& - \exp\left(-\frac{(x_1 - 0.5)^2}{2 \cdot 0.1^2} - \frac{(x_2 - 0.5)^2}{2 \cdot 0.1^2}\right)
\end{aligned} \tag{D.18}
$$

The ranges of the two design variables are listed in table D.8.

Table D.8: List of f2DWitchHat design variables and their ranges

Design variable	Meaning	Ranges
x_1	First design variable	$[0, 1]$
x_2	Second design variable	$[0, 1]$

D.9 Exponential2Split Function

f2DExponential2Split is a two-responses function, and the formulation is inspired by the f2DExponential2 test function (D.4).

$$f_1(x_1, x_2) = 20 x_1 \exp(-x_1^2 - x_2^2) \tag{D.19}$$

$$f_2(x_1, x_2) = \left(\tilde{x}_1 \exp(-\tilde{x}_1^2 - \tilde{x}_2^2)\right) \tag{D.20}$$

$$\tag{D.21}$$

where

$$\tilde{x}_1 = 2\,(x_1 - 5) \tag{D.22}$$

$$\tilde{x}_2 = 2\,(x_2 - 5) \tag{D.23}$$

The ranges of the two design variables are the same for f2DExponential function and they are listed in table D.4.

D.10 Estimation of propeller performance by Xrotor solver

f5DXrotor test functions use a black-box solver (XRotor) to estimate the performance of different propeller geometries at different operative conditions. In particular, the single response f5DXrotor compute the propeller shaft power, while three-response test case additionally returns estimations of thrust and efficiency. The spanwise variations of chord length and twist angle are described by two 1D NURBS functions of 3rd and 4th order, respectively. The ranges of the five design variables and the fixed values for f5DXrotor function are listed in table D.9 and table D.10, respectively.

Table D.9: List of f5DXrotor design variables and their ranges

Design variable	Meaning	Ranges
RPM	Propeller rotational speed	$[2500, 3500]$ RPM
v	Air speed	$[5, 40]\ \mathrm{m\,s^{-1}}$
$y^{(\mathrm{c})}_{\mathrm{tw},1}$	y coordinate of the first twist NURBS control point	$[65, 100]\ ^\circ$
$y^{(\mathrm{c})}_{\mathrm{tw},2}/y^{(\mathrm{c})}_{\mathrm{tw},1}$	Ratio of y coordinates of the second and first control points for the twist NURBS	$[0.3, 0.47]$
$y^{(\mathrm{c})}_{\mathrm{tw},3}/y^{(\mathrm{c})}_{\mathrm{tw},2}$	Ratio of y coordinates of the third and second control points for the twist NURBS	$[0.3, 0.47]$

Table D.10: List of f5DXrotor fix parameters

Parameter	Meaning	Value
R_T	Tip radius	0.6 m
h	Altitude	0 m
$\bar{x}^{(\mathrm{c})}_{\mathrm{ch},1}$	x coordinate of the first control point of the chord NURBS normalized by R_T	$1/15$
$\bar{x}^{(\mathrm{c})}_{\mathrm{ch},2}$	x coordinate of the second control point of the chord NURBS normalized by R_T	0.27
$\bar{x}^{(\mathrm{c})}_{\mathrm{ch},3}$	x coordinate of the third control point of the chord NURBS normalized by R_T	0.9
$\bar{x}^{(\mathrm{c})}_{\mathrm{ch},4}$	x coordinate of the fourth control point of the chord NURBS normalized by R_T	0.9993
$\bar{y}^{(\mathrm{c})}_{\mathrm{ch},1}$	y coordinate of the first control point of the chord NURBS normalized by R_T	0.14
$\bar{y}^{(\mathrm{c})}_{\mathrm{ch},2}$	y coordinate of the second control point of the chord NURBS normalized by R_T	0.3
$\bar{y}^{(\mathrm{c})}_{\mathrm{ch},3}$	y coordinate of the third control point of the chord NURBS normalized by R_T	0.11
$\bar{y}^{(\mathrm{c})}_{\mathrm{ch},4}$	y coordinate of the third control point of the chord NURBS normalized by R_T	0.0137
$\bar{x}^{(\mathrm{c})}_{\mathrm{tw},1}$	x coordinate of the first control point of the twist NURBS normalized by R_T	$1/15$
$\bar{x}^{(\mathrm{c})}_{\mathrm{tw},2}$	x coordinate of the second control point of the twist NURBS normalized by R_T	0.4
$\bar{x}^{(\mathrm{c})}_{\mathrm{tw},3}$	x coordinate of the third control point of the twist NURBS normalized by R_T	0.9993
$\alpha_{\mathrm{C_L}=0}$	Zero lift angle	$0\,^\circ$
$dC_\mathrm{L}/d\alpha$	Lift coefficient slope	2π
$C_{\mathrm{L,max}}$	Max lift coefficient	1.5
$C_{\mathrm{L,min}}$	Min lift coefficient	-0.5
$(dC_\mathrm{L}/d\alpha)_{\mathrm{st}}$	Lift coefficient slope at stall	0.1
C_m	Moment coefficient	-0.1
$C_{\mathrm{D,min}}$	Min drag coefficient	$0.13\dot{1}0^{-1}$
$C_\mathrm{L}@C_{\mathrm{D,min}}$	Lift coefficient at minimum drag	0.5
$dC_\mathrm{D}/d(C_\mathrm{L}^2)$	Drag polar coeffiecient	$0.4\dot{1}0^{-2}$

REFERENCES

[1] G. Abate and D. N. Mavris, "Performance analysis of different positions of leading edge tubercles on a wind turbine blade," in *2018 Wind Energy Symposium*, 2018, p. 1494.

[2] G. Abate, D. N. Mavris, and L. N. Sankar, "Performance effects of leading edge tubercles on the nrel phase vi wind turbine blade," *Journal of Energy Resources Technology*, 2019.

[3] E. P. Adorio and U Diliman, "Mvf-multivariate test functions library in C for unconstrained global optimization," 2005.

[4] A. Atkinson, A. Donev, and R. Tobias, *Optimum experimental designs, with SAS*. Oxford University Press, 2007, vol. 34.

[5] F. Aurenhammer, "Voronoi diagrams; a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, Sep. 1991.

[6] V. C. Aute, "Single and multiresponse adaptive design of experiments with application to design optimization of novel heat exchangers," University of Maryland, College Park, Department of Mechanical Engineering, Tech. Rep., 2009.

[7] T. Benamara, P. Breitkopf, I. Lepot, and C. Sainvitu, "Adaptive infill sampling criterion for multi-fidelity optimization based on gappy-pod," *Structural and Multidisciplinary Optimization*, vol. 54, no. 4, pp. 843–855, 2016.

[8] A. J. Booker, J. Dennis Jr, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset, "A rigorous framework for optimization of expensive functions by surrogates," *Structural optimization*, vol. 17, no. 1, pp. 1–13, 1999.

[9] M. A. Bossak, "Simulation based design," *Journal of Materials Processing Technology*, vol. 76, no. 1, pp. 8–11, 1998.

[10] G. E. Box and N. R. Draper, "A basis for the selection of a response surface design," *Journal of the American Statistical Association*, vol. 54, no. 287, pp. 622–654, 1959.

[11] P. Boyle, "Gaussian processes for regression and optimisation," 2007.

[12] H. Bozdogan, "Akaike's information criterion and recent developments in information complexity," *Journal of mathematical psychology*, vol. 44, no. 1, pp. 62–91, 2000.

[13] F. A. C. Viana, V. Pecheny, and R. T. Haftka, "Using cross validation to design conservative surrogates," *Aiaa Journal*, vol. 48, no. 10, pp. 2286–2298, 2010.

[14] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, vol. 4, no. 1, pp. 41–59, 1977.

[15] W. Chen, "A robust concept exploration method for configuring complex systems," PhD thesis, Georgia Institute of Technology, 1995.

[16] M. Chen Quin Lam, "Sequential adaptive designs in computer experiments for response surface model fit," PhD thesis, The Ohio State University, USA, 2008.

[17] R Choudhary, A Malkawi, and P. Papalambros, "Analytic target cascading in simulation-based building design," *Automation in construction*, vol. 14, no. 4, pp. 551–568, 2005.

[18] K Crombecq, I. Couckuyt, D. Gorissen, and T. Dhaene, "Space-filling sequential design strategies for adaptive surrogate modelling," in *The first international conference on soft computing technology in civil, structural and environmental engineering*, 2009.

[19] K. Crombecq, "Surrogate modeling of computer experiments with sequential experimental design," PhD thesis, Ghent University, 2011.

[20] K. Crombecq, D. Gorissen, D. Deschrijver, and T. Dhaene, "A novel hybrid sequential design strategy for global surrogate modeling of computer experiments," *SIAM Journal on Scientific Computing*, vol. 33, no. 4, pp. 1948–1974, 2011.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] R. DeLoach, "Applications of modern experiment design to wind tunnel testing at nasa langley research center," *AIAA paper*, vol. 713, 1998.

[23] M Drela and H Youngren, *Xrotor download page*, 2014.

[24] D. Duvenaud, "Automatic model construction with gaussian processes," PhD thesis, University of Cambridge, UK, 2014.

[25] N. Dyn, D. Levin, and S. Rippa, "Numerical procedures for surface fitting of scattered data by radial functions," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 2, pp. 639–659, 1986.

[26] B. Efron and B Efron, *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982, vol. 38.

[27] M. Eldred and D. Dunlavy, "Formulations for surrogate-based optimization with data fit, multifidelity, and reduced-order models," in *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2006, p. 7117.

[28] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, pp. 1–67, 1991.

[29] A. Garbo and B. J. German, "Comparison of adaptive design space exploration methods applied to S-duct CFD simulation," in *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2016, p. 0416.

[30] ——, "Adaptive sampling with adaptive surrogate model selection for computer experiment applications," in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, p. 4430.

[31] ——, "Performance assessment of a cross-validation sampling strategy with active surrogate model selection," *Structural and Multidisciplinary Optimization*, 2019.

[32] K. Gkaragkounis, E. Papoutsis-Kiachagias, and K. Giannakoglou, "The continuous adjoint method for shape optimization in conjugate heat transfer problems with turbulent incompressible flows," *Applied Thermal Engineering*, vol. 140, pp. 351–362, 2018.

[33] T. Goel, R. T. Hafkta, and W. Shyy, "Comparing error estimation measures for polynomial and kriging approximation of noise-free functions," *Structural and Multidisciplinary Optimization*, vol. 38, no. 5, pp. 429–442, 2009.

[34] T. Goel, R. T. Haftka, W. Shyy, and N. V. Queipo, "Ensemble of surrogates," *Structural and Multidisciplinary Optimization*, vol. 33, no. 3, pp. 199–216, 2007.

[35] T. Goel and N. Stander, "Comparing three error criteria for selecting radial basis function network topology," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 27, pp. 2137–2150, 2009.

[36] D. Gorissen, "Heterogeneous evolution of surrogate models," Master's thesis, Katholieke Universiteit Leuven, Belgium, 2007.

[37] D. Gorissen, K. Crombecq, I. Couckuyt, and T. Dhaene, "Automatic approximation of expensive functions with active learning," in *Foundations of Computational, Intelligence Volume 1*, Springer, 2009, pp. 35–62.

[38] D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene, "Sequential modeling of a low noise amplifier with neural networks and active learning," *Neural Computing and Applications*, vol. 18, no. 5, pp. 485–494, 2009.

[39] A. Gorodetsky and Y. Marzouk, "Mercer kernels and integrated variance experimental design: Connections between Gaussian process regression and polynomial approximation," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 4, no. 1, pp. 796–828, 2016.

[40] L. Gu, "A comparison of polynomial based regression models in vehicle safety analysis," in *ASME Design Engineering Technical Conferences, ASME Paper No.: DETC/DAC-21083*, 2001.

[41] P Hajela and L Berke, "Neural networks in structural analysis and design: An overview," *Computing Systems in Engineering*, vol. 3, no. 1-4, pp. 525–538, 1992.

[42] Z.-H. Han, S. Görtz, and R. Hain, "A variable-fidelity modeling method for aero-loads prediction," in *New results in numerical and experimental fluid mechanics vii*, Springer, 2010, pp. 17–25.

[43] R. L. Hardy, "Multiquadric equations of topography and other irregular surfaces," *Journal of geophysical research*, vol. 76, no. 8, pp. 1905–1915, 1971.

[44] J. van der Herten, I. Couckuyt, D. Deschrijver, and T. Dhaene, "A fuzzy hybrid sequential design strategy for global surrogate modeling of high-dimensional computer experiments," *SIAM Journal on Scientific Computing*, vol. 37, no. 2, A1020–A1039, 2015.

[45] C. Jiang, X. Cai, H. Qiu, L. Gao, and P. Li, "A two-stage support vector regression assisted sequential sampling approach for global metamodeling," *Structural and Multidisciplinary Optimization*, pp. 1–16, 2018.

[46] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 1–13, 2001.

[47] R. Jin, W. Chen, and A. Sudjianto, "On sequential sampling for global metamodeling in engineering design," in *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 2002, pp. 539–548.

[48] M. E. Johnson, L. M. Moore, and D. Ylvisaker, "Minimax and maximin distance designs," *Journal of statistical planning and inference*, vol. 26, no. 2, pp. 131–148, 1990.

[49] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[50] J. Kallrath, *Modeling languages in mathematical optimization*. Springer Science & Business Media, 2013, vol. 88.

[51] V. Klee, "On the complexity ofd-dimensional voronoi diagrams," *Archiv der Mathematik*, vol. 34, no. 1, pp. 75–80, 1980.

[52] J. P. Kleijnen, *Design and analysis of simulation experiments*. Springer, 2008, vol. 20.

[53] J. P. Kleijnen and W. C. Van Beers, "Application-driven sequential designs for simulation experiments: Kriging metamodelling," *Journal of the Operational Research Society*, vol. 55, no. 8, pp. 876–883, 2004.

[54] H.-T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM (JACM)*, vol. 22, no. 4, pp. 469–476, 1975.

[55] T. H. Lee, *The design of CMOS radio-frequency integrated circuits*. University Press, Cambridge, UK, 2003.

[56] G Li, S Azarm, A Farhang-Mehr, and A. Diaz, "Approximation of multiresponse deterministic engineering simulations: A dependent metamodeling approach," *Structural and Multidisciplinary Optimization*, vol. 31, no. 4, pp. 260–269, 2006.

[57] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Structural and Multidisciplinary Optimization*, vol. 35, no. 6, pp. 561–569, 2008.

[58] Y. Lin, "An efficient robust concept exploration method and sequential exploratory experimental design," PhD thesis, Georgia Institute of Technology, 2004.

[59] Y.-c. Lin, B. J. Fregly, R. T. Haftka, and N. V. Queipo, "Surrogate-based contact modeling for efficient dynamic simulation with deformable anatomic joints," 2005.

[60] H. Liu, Y.-S. Ong, and J. Cai, "A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design," *Structural and Multidisciplinary Optimization*, pp. 1–24, 2017.

[61] H. Liu, S. Xu, Y. Ma, X. Chen, and X. Wang, "An adaptive Bayesian sequential sampling approach for global metamodeling," *Journal of Mechanical Design*, vol. 138, no. 1, 2016.

[62] H. Liu, S. Xu, X. Wang, S. Yang, and J. Meng, "A multi-response adaptive sampling approach for global metamodeling," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2016.

[63] J. L. Loeppky, J. Sacks, and W. J. Welch, "Choosing the sample size of a computer experiment: A practical guide," *Technometrics*, vol. 51, no. 4, pp. 366–376, 2009.

[64] A. Lovison and E. Rigoni, "Adaptive sampling with a lipschitz criterion for accurate metamodeling," *Communications in Applied and Industrial Mathematics*, vol. 1, no. 2, pp. 110–126, 2011.

[65] T. Mackman, C. Allen, M. Ghoreyshi, and K. Badcock, "Comparison of adaptive sampling methods for generation of surrogate aerodynamic models," *AIAA journal*, vol. 51, no. 4, pp. 797–808, 2013.

[66] L. Magnier and F. Haghighat, "Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network," *Building and Environment*, vol. 45, no. 3, pp. 739–746, 2010.

[67] P. Marcy, "On the use and utility of gradient information in computer experiments," PhD thesis, University of Wyoming. Department of Statistics, USA, 2014.

[68] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[69] M. Meckesheimer, A. J. Booker, R. R. Barton, and T. W. Simpson, "Computationally inexpensive metamodel assessment strategies," *AIAA journal*, vol. 40, no. 10, pp. 2053–2060, 2002.

[70] E. Mehdad and J. P. Kleijnen, "Stochastic intrinsic kriging for simulation metamodeling," *Applied Stochastic Models in Business and Industry*, vol. 34, no. 3, pp. 322–337, 2018.

[71] A. Mehmani, S. Chowdhury, and A. Messac, "Predictive quantification of surrogate model fidelity based on modal variations with sample density," *Structural and Multidisciplinary Optimization*, vol. 52, no. 2, pp. 353–373, 2015.

[72] A. Mehmani, J. Zhang, S. Chowdhury, and A. Messac, "Surrogate-based design optimization with adaptive sequential sampling," in *53rd AIAA/ASME/ASCE/AHS/ASC*

Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA, 2012, p. 1527.

[73] P. K. Mishra, S. K. Nath, M. K. Sen, and G. E. Fasshauer, "Hybrid gaussian-cubic radial basis functions for scattered data interpolation," *arXiv preprint*, 2015.

[74] E. A. Morelli and R. DeLoach, "Wind tunnel database development using modern experiment design and multivariate orthogonal functions," in *41st AIAA Aerospace Sciences Meeting and Exhibit*, vol. 653, 2003.

[75] M. D. Morris, T. J. Mitchell, and D. Ylvisaker, "Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction," *Technometrics*, vol. 35, no. 3, pp. 243–255, 1993.

[76] P. C. Murphy and D. Landman, "Experiment design for complex vtol aircraft with distributed propulsion and tilt wing," in *AIAA Atmospheric Flight Mechanics Conference*, Description of a large dimensionality DoE on a wind tunnel. It mentions R2, adj-R2, pred-R2, 2015, p. 0017.

[77] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook, *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons, 2016.

[78] M. A. Osborne, S. J. Roberts, A. Rogers, and N. R. Jennings, "Real-time information processing of environmental sensor network data using bayesian gaussian processes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 1, p. 1, 2012.

[79] E. Papoutsis-Kiachagias, S Porziani, C Groth, M. Biancolini, E Costa, and K. Giannakoglou, "Aerodynamic optimization of car shapes using the continuous adjoint method and an rbf morpher," in *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, Springer, 2019, pp. 173–187.

[80] K. Paul and H. James, "Motivated metamodels: Synthesis of cause-effect reasoning and statistical metamodeling," *Santa Monica: RAND*, 2003.

[81] H. Peng and X. Ling, "Optimal design approach for the plate-fin heat exchangers using neural networks cooperated with genetic algorithms," *Applied Thermal Engineering*, vol. 28, no. 5, pp. 642–650, 2008.

[82] R. Perrini, *Pianeta scuola. Dalla A come apprendimento alla V come valutazione*. Armando Editore, 2002.

[83] S. Pierret and R. Van den Braembussche, "Turbomachinery blade design using a navier-stokes solver and artificial neural network," in *ASME 1998 International Gas Turbine and Aeroengine Congress and Exhibition*, American Society of Mechanical Engineers, 1998, V001T01A002–V001T01A002.

[84] M. Plumlee and D. W. Apley, "Lifted brownian kriging models," *Technometrics*, no. just-accepted, 2016.

[85] A. Quan, *Batch Sequencing Methods for Computer Experiments*. The Ohio State University, 2014.

[86] N. V. Queipo, J. V. Goicochea, and S. Pintos, "Surrogate modeling-based optimization of sagd processes," *Journal of Petroleum Science and Engineering*, vol. 35, no. 1, pp. 83–93, 2002.

[87] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[88] J. R. RA Martins, J. J. Alonso, and J. J. Reuther, "High-fidelity aerostructural design optimization of a supersonic business jet," *Journal of Aircraft*, vol. 41, no. 3, pp. 523–530, 2004.

[89] C. E. Rasmussen and C. K. Williams, "Gaussian processes for machine learning," Cambridge, MA, USA, Tech. Rep. ISBN 0-262-18253-X, 2006.

[90] R. Reichart, K. Tomanek, U. Hahn, and A. Rappoport, "Multi-task active learning for linguistic annotations.," in *ACL*, vol. 8, 2008, pp. 861–869.

[91] C. F. Reinhart and J. Wienold, "The daylighting dashboard–a simulation-based design analysis for daylit spaces," *Building and environment*, vol. 46, no. 2, pp. 386–396, 2011.

[92] D. A. Romero, C. H. Amon, and S. Finger, "On adaptive sampling for single and multi-response bayesian surrogate models," in *ASME 2006 international design engineering technical conferences and computers and information in engineering conference*, American Society of Mechanical Engineers, 2006, pp. 393–404.

[93] ——, "Multiresponse metamodeling in simulation-based design applications," *Journal of Mechanical Design*, vol. 134, no. 9, p. 091 001, 2012.

[94] B. Rosenbaum, "Efficient global surrogate models for responses of expensive simulations," *These de doctorat, Universität Trier*, 2013.

[95] D. E. Rumelhart, B. Widrow, and M. A. Lehr, "The basic ideas in neural networks," *Communications of the ACM*, vol. 37, no. 3, pp. 87–93, 1994.

[96] J. Sacks, S. B. Schiller, and W. J. Welch, "Designs for computer experiments," *Technometrics*, vol. 31, no. 1, pp. 41–47, 1989.

[97] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical science*, pp. 409–423, 1989.

[98] T. J. Santner, B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*. Springer Science & Business Media, 2013.

[99] S. Shan and G. G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Structural and Multidisciplinary Optimization*, vol. 41, no. 2, pp. 219–241, 2010.

[100] M. S. Shephard, M. W. Beall, R. M. O'bara, and B. E. Webster, "Toward simulation-based design," *Finite Elements in Analysis and Design*, vol. 40, no. 12, pp. 1575–1598, 2004.

[101] T. W. Simpson, "Development of a design process for realizing open engineering systems," PhD thesis, Georgia Institute of Technology, 1995.

[102] T. W. Simpson, D. K. Lin, and W. Chen, "Sampling strategies for computer experiments: Design and analysis," *International Journal of Reliability and Applications*, vol. 2, no. 3, pp. 209–240, 2001.

[103] T. W. Simpson, J. Poplinski, P. N. Koch, and J. K. Allen, "Metamodels for computer-based engineering design: Survey and recommendations," *Engineering with computers*, vol. 17, no. 2, pp. 129–150, 2001.

[104] M. E. Tarter, "Inverse cumulative approximation and applications," *Biometrika*, vol. 55, no. 1, pp. 29–41, 1968.

[105] F. A. Viana, R. T. Haftka, and V. Steffen Jr, "Multiple surrogates: How cross-validation errors can help us to obtain the best predictor," *Structural and Multidisciplinary Optimization*, vol. 39, no. 4, pp. 439–457, 2009.

[106] B. Wang, P. Hao, G. Li, Y. Fang, X. Wang, and X. Zhang, "Determination of realistic worst imperfection for cylindrical shells using surrogate model," *Structural and Multidisciplinary Optimization*, vol. 48, no. 4, pp. 777–794, 2013.

[107] D. H. Wolpert, "The supervised learning no-free-lunch theorems," in *Soft computing and industry*, Springer, 2002, pp. 25–42.

[108] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[109] D. H. Wolpert, W. G. Macready, *et al.*, "No free lunch theorems for search," Technical Report SFI-TR-95-02-010, Santa Fe Institute, Tech. Rep., 1995.

[110] Y.-T. Wu, Y. Shin, R. Sues, and M. Cesare, "Safety-factor based approach for probability-based design optimization," in *19th AIAA Applied Aerodynamics Conference*, 2001, p. 1522.

[111] S. Xu, H. Liu, X. Wang, and X. Jiang, "A robust error-pursuing sequential sampling approach for global metamodeling based on voronoi diagram and cross validation," *Journal of Mechanical Design*, vol. 136, no. 7, p. 071 009, 2014.

[112] B Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[113] N. Zhang and D. W. Apley, "Fractional Brownian fields for response surface metamodeling," *Journal of Quality Technology*, vol. 46, no. 4, p. 285, 2014.

[114] ——, "Brownian integrated covariance functions for gaussian process modeling: Sigmoidal versus localized basis functions," *Journal of the American Statistical Association*, no. just-accepted, pp. 00–00, 2015.

[115] Y. Zhang, N. H. Kim, and R. T. Haftka, "General surrogate adaptive sampling using interquartile range for design space exploration," in *AIAA Scitech 2019 Forum*, 2019, p. 2213.

[116] Q. Zhou, X. Shao, P. Jiang, Z. Gao, H. Zhou, and L. Shu, "An active learning variable-fidelity metamodelling approach based on ensemble of metamodels and objective-oriented sequential sampling," *Journal of Engineering Design*, vol. 27, no. 4-6, pp. 205–231, 2016.