Chapter 3

Overview of Classic Surrogates and Metamodeling Strategies in Structural Analysis

Chapter details

Chapter DOI:

https://doi.org/10.4322/978-65-86503-88-3.c03

Chapter suggested citation / reference style:

Santos, Rogerio R., et al. (2022). "Overview of Classic Surrogates and Metamodeling Strategies in Structural Analysis". *In* Jorge, Ariosto B., et al. (Eds.) *Uncertainty Modeling: Fundamental Concepts and Models*, Vol. III, UnB, Brasilia, DF, Brazil, pp. 54–88. Book series in Discrete Models, Inverse Methods, & Uncertainty Modeling in Structural Integrity.

P.S.: DOI may be included at the end of citation, for completeness.

Book details

Book: Uncertainty Modeling: Fundamental Concepts and Models

Edited by: Jorge, Ariosto B., Anflor, Carla T. M., Gomes, Guilherme F., & Carneiro, Sergio H. S.

Volume III of Book Series in:

Discrete Models, Inverse Methods, & Uncertainty Modeling in Structural Integrity

Published by: UnB City: Brasilia, DF, Brazil Year: 2022

DOI: https://doi.org/10.4322/978-65-86503-88-3

Overview of Classic Surrogates and Metamodeling Strategies in Structural Analysis

Rogerio R. Santos[1*], Guilherme F. Gomes[2], and Ariosto B. Jorge[3]

[1]Division of Mechanical Engineering, Technological Institute of Aeronautics - ITA, Brazil. E-mail: rsantos9@gmail.com
[2]Mechanical Engineering Institute, Federal University of Itajuba - UNIFEI, Brazil. E-mail: guilhermefergom@unifei.edu.br
[3]Post-Graduate Program - Integrity of Engineering Materials, University of Brasilia, Brazil. E-mail: ariosto.b.jorge@gmail.com

*Corresponding author

Abstract

In some situations, the need for multiple evaluations of a computational model is not feasible due to the large cost in terms of budget or time required. The dominant practice consists in the creation of statistical approximations for the response of the model and performing additional analysis on the meta-model. The present contribution discusses some classic alternatives for the establishment of experimental designs. Next, the Least-Squares and Kernel methods are outlined as alternatives to approximate models. Subsequently, characteristics of the Parallel, Sequential, and Multi-level ensembling methods are presented. The current work is concluded with considerations on the goodness-of-fit assessment.

Keywords: Surrogate modeling, Metamodeling, Design of Experiments, Machine Learning.

1 Introduction

Contemporary projects in the field of structural analysis have to deal with a lot of data. The need to translate raw numbers into intelligence that drives real-world projects requires the ability to classify and organize information that arises from multiple and sometimes conflicting requirements.

In this context, how to break down and structure complex problems and data sets is of paramount importance. Design of Experiments (DoE) is concerned with designing tasks capable of describing and explaining the variation of information under specific hypothesized conditions. It is associated with experiments in which the design introduces conditions that directly affect the variation/response.

Inferring the system's behavior from a reduced set of data can contribute to reducing the computational budget of a project and increase the probability of identifying unforeseen scenarios. In the present contribution, classical and contemporary techniques are presented and their

suitability for structural analysis is discussed. It starts with the concepts of Design of Experiments, Metamodeling and Surrogate alternatives for the establishment of experimental designs. Next, the Least-Squares and Kernel methods are outlined as alternatives of approximate models. Subsequently, characteristics of the Parallel, Sequential and Multi-level ensembling methods are presented. Considerations on the goodness-of-fit appraisal will concludes the present contribution.

2 Design of Experiments, Metamodeling and Surrogates

In certain circumstances, it is not feasible or practical to calculate exact values for the functions $F(\{X\}), G_j(\{X\})$ and $H_k(\{X\})$ pertaining to the optimization problem formulation seen in chapters "An overview of Linear and Non-linear Programming methods for Structural Optimization" (Choze et al. [2022b]) and "Overview of Traditional and Recent Heuristic Optimization Methods" (Choze et al. [2022a]). One such possibility is whenever the computational cost is too burdensome, and even the specialized structural synthesis approximations (Choze et al. [2022b]) cannot provide an adequate solution. Another situation, in itself or combined with excessive computational cost, is when the calculation engines responsible for providing the values of the many quantities involved belong to computational domains that are disparate enough to jeopardize a direct integration, and hence a common frame of reference has to be obtained to reconcile them.

In either case, it is mainstream practice to create statistical approximations of these quantities, henceforth denoted as $\hat{F}(\{X\}), \hat{G}_j(\{X\})$ and $\hat{H}_k(\{X\})$, with the " $\hat{}$ " notation used to highlight their non-exact nature. As approximated entities, there are mechanisms to drive the computational burden down and, both as closed-form equations or represented through standardized database functions, their computational compatibility can be assured to the extent that they are operated together within the same optimization procedure. The overall procedure for obtaining these approximations is explained after its introduction in Table 1 (Chihara and Hesterberg [2018]).

From left to right, the first step consists of observing enough values of decision variables combinations ($\{X\}$) and the corresponding objective/constraint values that they yield. Explicitly, the yielded values, which are collectively denoted as $\{Y\}$, are dependent upon $\{X\}$ which, in statistical nomenclature, are the independent variables (within reason, freely chosen to map the decision space). Depending on the specific purposes of each study, one or more suitable design of experiments (first column in Table 1) will determine specifically which setups of independent variables are observed to form a matrix where, in a given layout, each row corresponds to an observation (as determined by the applicable/available experimental design) and each column maps into the corresponding element of the vector of decision variables $\{X\}$. A statistical surrogate will map an approximation to describe the underlying relationship between independent and dependent variables, pairing each row in $[X]$ (now in plain matrix notation) with the corresponding one in $\{Y\}$ (still in vector notation, focusing on a single response at a time). One or more such methods are chosen from the second column in Table 1. Although the number of choice possibilities is potentially high, reflecting a myriad of different approaches, the fundamental operating principles underpinning all of them are common, relating to the optimization of a goodness-of-fit metric. Finally, and most importantly, even though a suitable goodness of fit may be reached once executing of steps 2 and 3 within Table 1 (with strictly training data, as more usual, or with a mix of training and validation data), additional appraisal, corresponding to step 4, is strongly advised for a final and complete performance assessment of the surrogate's capabilities.

Specific techniques and additional comments relative to the steps outlined above are presented in the subsequent sections. Several of these instances are supplemented with their corresponding computer codes implemented with the R statistical programming platform (R Core Team [2020]), indicated as "Code". For this purpose, the command `install.packages` is the main tool to add new packages to the R software. It takes a vector of names and a destination library, downloads

Table 1: Standardized steps sequence for building statistical approximations, also known as meta-models or surrogates.

Experimental Design	Model Choice	Model Fitting		Quality Verification
(Fractional) Factorial	Polynomial	Least Squares Regression	Response Surface	Analysis of Residuals
Central Composite		Weighted Least Squares Regression		ANOVA
Box-Behnken				⋮
Alphabet-Optimal	Splines	Best Linear Unbiased Predictor	Bayesian Models	
Orthogonal	Frequency Domain	Best Linear Predictor		
Plackett-Burman	Kernel Smoothing			
Hexagon	Radial Basis Functions			
Hybrid				
Latin Hypercube				
Enumerative	Neural Networks	Back-Propagation	Neural Networks	Residual Error
	Rule based or Decision Tree	Entropy	Inductive Learning	
Random				

the packages from the repositories (from a local folder or remote web server) and installs them in the R environment. To install the packages needed to run the codes in this chapter, for example, enter the following:

```
install.packages('FrF2','rsm','lhs','glmnet','DiceKriging',
'RSNNS','randomForest')
```

Help is available at the command prompt via `?install.packages` command.

3 Experimental Design

From a statistical theory perspective, a more exhaustive list of possible experimental design approaches is richly documented in the literature, such as in references Myers et al. [2016], Box et al. [2005], and Khuri and Cornell [2019]. The following sections are extracts of this broader picture that more closely correspond to the state-of-the-art and practice for computer analysis and optimization of structural systems. Nevertheless, active research is ongoing in the direction of broadening the scope and possibilities within the field, and the interested reader is referred to Garud and Kraft [2017a], Garud and Kraft [2017b], Bhosekar and Ierapetritou [2018], Boukouvala and Floudas [2017], and Pronzato and Müller [2012].

3.1 Full and fractional factorial designs

Initial exploration and understanding of decision spaces in m variables tends to be undertaken by screening experimental designs, whose purpose is to filter out variables that do not influence the outcome(s). Among the various options for screening designs, the simplest ones are factorial designs at two levels, which serve to identify linear relationships between inputs and outputs. For $m = 5$ and variable ranges codified between -1 for lower bound and $+1$ for upper bound, the resulting combination would be as appears on Table 2.

The 32 total combinations arise from 2^m, for two levels (at the lower and upper bounds for each variable) over $m = 5$ variables, and define an hypercube (each edge corresponding to one dimension or variable) with the experimental points at the vertices. The exponential growth with respect to m indicates that, in practice, this may even be a simple approach, but is only feasible for screening purposes with moderate values of m and/or low cost for performing each individual run. This limitation may be better perceived if one considers $m = 10$ variables yielding 1024 total runs just for screening on a linear relationship basis. For completeness of this discussion, consider also the inclusion of simple curvature with base 3 would result in $3^{10} = 59049$ runs, proving the point that straight factorial combination may only work in very particular cases and may lead to experimental inefficiencies in exchange for its simplicity. For the purposes of computational coverage, creation of full factorial designs in R (R Core Team [2020]) is illustrated in Code 1 for 3 factors (A, B and C).

As an initial step to trade-off complexity and efficiency, the introduction of fractional factorial designs leverages the idea of incurring into the partial cost of factorial designs while reducing the clarity of the conclusions to be obtained. The multiple trade-off points within this balancing act are measured by different resolutions, which become an attribute of the experimental design similarly to its base and number of variables. In formal terms, resolution is defined as the minimum word length in the defining relation, excluding 1. Whereas Myers et al. [2016] and Box et al. [2005] present complete theoretical discussions about the defining relations of different experimental designs and, consequently, their resolutions, the practical aspect to be emphasized is that higher resolutions are desired, with the following recommendations: a minimum of 3 (typically noted as III) for variable screening, a minimum of 4 (typically noted as IV) for model fitting and, finally, a minimum of 5 (typically noted as V) for optimization.

To illustrate these concepts, consider alternatives for fractional designs that can be derived from Table 2, utilizing only part of the 5 total factors and confounding the remaining k ones, that

Table 2: Full factorial experimental design at two-levels with $m = 5$ independent / input variables, whose ranges are codified between -1 (lower bound) and $+1$ (upper bound).

Run	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5
1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	1
3	-1	-1	-1	1	-1
4	-1	-1	-1	1	1
5	-1	-1	1	-1	-1
6	-1	-1	1	-1	1
7	-1	-1	1	1	-1
8	-1	-1	1	1	1
9	-1	1	-1	-1	-1
10	-1	1	-1	-1	1
11	-1	1	-1	1	-1
12	-1	1	-1	1	1
13	-1	1	1	-1	-1
14	-1	1	1	-1	1
15	-1	1	1	1	-1
16	-1	1	1	1	1
17	1	-1	-1	-1	-1
18	1	-1	-1	-1	1
19	1	-1	-1	1	-1
20	1	-1	-1	1	1
21	1	-1	1	-1	-1
22	1	-1	1	-1	1
23	1	-1	1	1	-1
24	1	-1	1	1	1
25	1	1	-1	-1	-1
26	1	1	-1	-1	1
27	1	1	-1	1	-1
28	1	1	-1	1	1
29	1	1	1	-1	-1
30	1	1	1	-1	1
31	1	1	1	1	-1
32	1	1	1	1	1

is, 2^{5-k}. The resulting possibilities are detailed in Table 3, which is generalized for a large number of variables/fractions/resolutions of practical applicability in NIS.

Code 1: Full Factorial Design for 3 factors (A B and C)

```
expand.grid(A = c("-1", "+1"), B = c("-1", "+1"),
C = c("-1", "+1"))
A  B  C
1 -1 -1 -1
2  1 -1 -1
3 -1  1 -1
4  1  1 -1
5 -1 -1  1
6  1 -1  1
7 -1  1  1
8  1  1  1
```

Code 2 brings the R counterpart of the $k = 1, 2^{5-1}_{R=V}$ design also listed in Table 3. Please note the difference in notation, where the factors are alluded to as capital letters A, B, C, D and E instead of numbers, both versions being prevalent (and conceptually equivalent) in the literature.

Code 2: Regular Fractional Factorial 2-level design

```
FrF2(16,5, generators='ABCD')
    A  B  C  D  E
 1  1 -1  1 -1  1
 2 -1  1 -1 -1 -1
 3 -1 -1  1  1  1
 4  1  1  1 -1 -1
 5  1 -1  1  1 -1
 6  1  1  1  1  1
 7  1 -1 -1  1  1
 8  1  1 -1 -1  1
 9 -1  1  1  1 -1
10  1  1 -1  1 -1
11 -1  1 -1  1  1
12 -1 -1 -1  1 -1
13 -1 -1  1 -1 -1
14 -1  1  1 -1  1
15 -1 -1 -1 -1  1
16  1 -1 -1 -1 -1
class=design, type= FrF2.generators
```

Even with the introduction of the fractional strategies and the consequent reduction in the number of runs, some situations result in non-affordable experimental effort. This is often the case when non-linearity prevail and/or the experimental investigation actually targets some level of prediction, beyond pure screening.

Such situations require alternative approaches, and the most popular ones for computational experiments, especially when dealing with structural mechanics, are described in the next two subsections.

Table 3: Fractional factorial experimental design options with various resolutions out of the $m = 5$ independent/input variables listed in Table 2. * confounded variable/Factor.

<table>
<tr><th>Fraction (k) and naming</th><th>Runs</th><th>Resolution</th><th>Confounding Structure</th></tr>
<tr><td>$k = 1$,
$2^{5-1}_{R=V}$</td><td>
<table>
<tr><th>Run</th><th>$F1$</th><th>$F2$</th><th>$F3$</th><th>$F4$</th><th>$F5^*$</th></tr>
<tr><td>1</td><td>-1</td><td>-1</td><td>-1</td><td>-1</td><td>1</td></tr>
<tr><td>2</td><td>-1</td><td>-1</td><td>-1</td><td>1</td><td>-1</td></tr>
<tr><td>3</td><td>-1</td><td>-1</td><td>1</td><td>-1</td><td>-1</td></tr>
<tr><td>4</td><td>-1</td><td>-1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>5</td><td>-1</td><td>1</td><td>-1</td><td>-1</td><td>-1</td></tr>
<tr><td>6</td><td>-1</td><td>1</td><td>-1</td><td>1</td><td>1</td></tr>
<tr><td>7</td><td>-1</td><td>1</td><td>1</td><td>-1</td><td>1</td></tr>
<tr><td>8</td><td>-1</td><td>1</td><td>1</td><td>1</td><td>-1</td></tr>
<tr><td>9</td><td>1</td><td>-1</td><td>-1</td><td>-1</td><td>-1</td></tr>
<tr><td>10</td><td>1</td><td>-1</td><td>-1</td><td>1</td><td>1</td></tr>
<tr><td>11</td><td>1</td><td>-1</td><td>1</td><td>-1</td><td>1</td></tr>
<tr><td>12</td><td>1</td><td>-1</td><td>1</td><td>1</td><td>-1</td></tr>
<tr><td>13</td><td>1</td><td>1</td><td>-1</td><td>-1</td><td>1</td></tr>
<tr><td>14</td><td>1</td><td>1</td><td>-1</td><td>1</td><td>-1</td></tr>
<tr><td>15</td><td>1</td><td>1</td><td>1</td><td>-1</td><td>-1</td></tr>
<tr><td>16</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>
</td><td>V</td><td>DEFINING RELATION:
I = 12345

CONFOUNDED VARIABLE(S):
$F5 = F1 \times F2 \times F3 \times F4$

CONFOUNDING STRUCTURE:
$F1 = F1 + F2345$
$F2 = F2 + F1345$
$F3 = F3 + F1245$
$F4 = F4 + F1235$
$F5 = F5 + F1234$

Note: each variable/Factor confounded with interactions of order 4, which are negligible.</td></tr>
<tr><td>$k = 2$,
$2^{5-2}_{R=III}$</td><td>
<table>
<tr><th>Run</th><th>$F1$</th><th>$F2$</th><th>$F3$</th><th>$F4^*$</th><th>$F5^*$</th></tr>
<tr><td>1</td><td>-1</td><td>-1</td><td>-1</td><td>1</td><td>1</td></tr>
<tr><td>2</td><td>-1</td><td>-1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>3</td><td>-1</td><td>1</td><td>-1</td><td>-1</td><td>1</td></tr>
<tr><td>4</td><td>-1</td><td>1</td><td>1</td><td>-1</td><td>-1</td></tr>
<tr><td>5</td><td>1</td><td>-1</td><td>-1</td><td>-1</td><td>-1</td></tr>
<tr><td>6</td><td>1</td><td>-1</td><td>1</td><td>-1</td><td>1</td></tr>
<tr><td>7</td><td>1</td><td>1</td><td>-1</td><td>1</td><td>-1</td></tr>
<tr><td>8</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>
</td><td>III</td><td>DEFINING RELATION:
I = 124 = 135 = 2345

CONFOUNDED VARIABLE(S):
$F4 = F1 \times F2$
$F5 = F1 \times F3$

CONFOUNDING STRUCTURE:
$F1 = F1{+}F24{+}F35{+}F12345$
$F2 = F2{+}F14{+}F345{+}F1235$
$F3 = F3{+}F15{+}F245{+}F1234$
$F4 = F4{+}F12{+}F235{+}F1345$
$F5 = F5{+}F13{+}F234{+}F1245$

Note: each variable/Factor confounded with interactions of order 4, which are negligible.</td></tr>
</table>

3.2 Central composite designs

Central Composite designs are suitable for situations of moderate dimensionality (when some form of fractional or even full factorial design fits the experimental budget) but it is important to consider non-linear effects that will certainly not be captured via two-level combinations. In this context, consider the initial scenario with 3 independent variables and the fractional form 2^{3-1} resulting in 4 total linear experimental runs, in the portion denoted as "cube" (signifying edges with equal lengths). Adding a total of 5 additional runs (one at the "center", and the remaining at the far edges of a "star") would cover the non-linear effects as represented schematically in Figure 1, which is mirrored in Table 4.

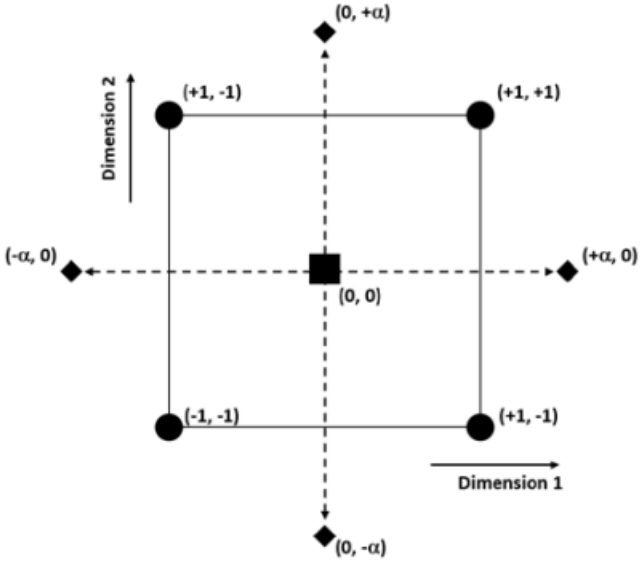


Figure 1: Diagram of a Central Composite Design with linear and non-linear portions represented in two-dimensions.

Table 4: Central Composite Design corresponding to Figure 1, with 2 factors covered at 5 total levels.

Run #	F1	F2
1 ("cube")	-1	-1
2 ("cube")	-1	1
3 ("cube")	1	-1
4 ("cube")	1	1
5 ("center")	0	0
6 ("star")	$-\alpha$	0
7 ("star")	$+\alpha$	0
8 ("star")	0	$-\alpha$
9 ("star")	0	$+\alpha$

The square vertices are the factorial points spanning the range between the linear lower bound (-1) and the linear upper bound (+1), and the qualifier "linear" is deliberate to indicate that the overall ranges have to be examined and equalized so that no experimental points are configured outside of any feasible experimental range. The reason, which also appears on Figure 1, is that the

non-linear effects are captured by the rhombuses α units afar from the center (square icon), which usually assume absolute values larger than unity. Determination of the values of α depend upon a number of factors, both formal/mathematical and also practical:

- α for rotation: ensures that the resulting design yields predictive variances (i.e., uncertainties) that, at any point x, depend only on the distance of x from the design center point. A design with this desirable property can be rotated around its center point without changing the prediction variance at x, hence its nomenclature;
- α for orthogonality: guarantees that the effect of one factor or interaction can be estimated separately from the effect of any other factor or interaction in the model, which counters the factor confusion arising from fractional factorial designs and their associated (lower) resolution;
- Special cases and arbitrary values of α: practical constraints may limit the choice of values for α in such a way that neither rotation or orthogonality is obtained, requiring additional and more careful assessment of goodness-of-fit prior to utilizing the model resulting from the experiment. One possibility is that, due to lack of experimental bandwidth, the values or α are not allowed to go beyond the linear range [-1, +1], in which case the resulting designs are called face centered composite designs.

Once again, the R code counterpart is offered in Code 3, highlighting configuration parameters necessary for the creation of a Central Composite Design.

Code 3: Central Composite Design example

```
library(rsm)
n_var = 3
n_center_points_cube = 4
n_center_points_star = 6
ccd(n_var, n0 = c(n_center_points_cube,
  n_center_points_star),
  alpha = "orthogonal")

  run.order std.order x1.as.is x2.as.is x3.as.is Block
1 1 5 -1 -1 1 1
2 2 9 0 0 0 1
3 3 2 1 -1 -1 1
4 4 3 -1 1 -1 1
5 5 11 0 0 0 1
6 6 4 1 1 -1 1
7 7 8 1 1 1 1
8 8 1 -1 -1 -1 1
9 9 6 1 -1 1 1
10 10 12 0 0 0 1
11 11 10 0 0 0 1
12 12 7 -1 1 1 1
13 1 10 0 0 0 2
14 2 7 0 0 0 2
15 3 9 0 0 0 2
16 4 2 2 0 0 2
17 5 8 0 0 0 2
18 6 3 0 -2 0 2
19 7 4 0 2 0 2
20 8 5 0 0 -2 2
21 9 1 -2 0 0 2
22 10 12 0 0 0 2
23 11 11 0 0 0 2
24 12 6 0 0 2 2
```

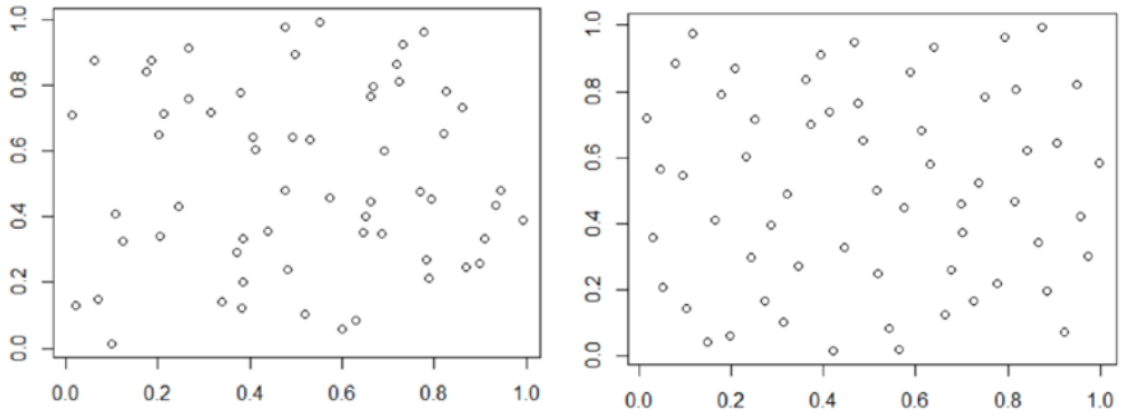
3.3 Latin Hypercube (computational) designs

Since their inception in the seminal paper by Sacks et al, 1989, computer experimental designs have gained popularity due to their inherent suitability regarding phenomena that are immune to experimental variation. In this sense, any of the experimental setups discussed in sections 3.1 and 3.2 above would in principle require replications at each point whenever feasible, if they are undertaken in the actual physical world. In computers however, barring minor differences due to platforms (chipset, operating systems, etc.), such replicates would yield the same results at any given experimental run.

At any measure, with or without replications, application scenarios with relatively high dimensionality and any degree of non-linearity may push beyond experimental budget feasibility even for Central Composite Designs. In such scenarios, the ideal combination of attributes for the experimental design would encompass:

1. Low, and ideally fixed/predetermined experimental cost (total number of runs);
2. Ability to explore various levels and capture non-linearity;
3. Ability to execute item 2 minimizing redundancy of information aggregated by each individual experimental run.

Whereas a random sample with N points over m dimensions would comply with these 3 requirements, it would offer low to no control over coverage of the space, with the enduring possibility that some areas are over-sampled at the expense of voids in other locations (Figure 2a). On the other hand, a true space filling experimental design would look like Figure 2b and, as implied by the name, experimental samples would be distributed as to satisfy the uniform filling of the available space, such as minimizing the maximum distance or, conversely, maximizing the minimum distance between any 2 given points in m dimensional space.



(a) Random. (b) Optimal Latin Hyper Cube.

Figure 2: Random vs. Space filling Design with 60 points over 2 dimensions between 0 and 1. Generated in R with random seed = 1 for Figure (2a) and function optimumLHS for Figure (2b).

Code 4 contains the necessary R syntax to create designs such as that of Figure 2b.

Code 4: Optimum Latin Hypercube sample

```
library(lhs)
n_var = 3
n_samples = 16
optimumLHS(n = n_samples, k = n_var)
       [,1]       [,2]       [,3]
 [1,] 0.77986305 0.25962784 0.98113240
 [2,] 0.73374286 0.89983958 0.65652518
 [3,] 0.08106826 0.38917024 0.80616234
 [4,] 0.67524880 0.03049537 0.36286886
 [5,] 0.34750631 0.78054941 0.84859144
 [6,] 0.12511944 0.17081607 0.10528484
 [7,] 0.52469292 0.60694614 0.04280036
 [8,] 0.41165871 0.54098857 0.50797994
 [9,] 0.98482138 0.36344077 0.73544520
[10,] 0.84308190 0.18906118 0.20958842
[11,] 0.02456373 0.64927328 0.25217138
[12,] 0.61763298 0.46786939 0.89379079
[13,] 0.47930504 0.98500063 0.15556912
[14,] 0.22676730 0.86768002 0.48733852
[15,] 0.26289577 0.06276729 0.58312047
[16,] 0.92962543 0.69242589 0.43707276
```

The ability to cover multidimensional spaces more uniformly than pure random numbers and the amount of control relative to overall experimental cost have converted Latin Hypercube Sampling (LHS) designs into the *de facto* standard of computational design and analysis of experiments.

4 Choice and Fitting of Approximate Models

4.1 Least-Square based regression methods

The most fundamental method of empirical surrogates building is the Ordinary Least Squares (OLS for brevity) regression, an average filter that creates an approximation linear in the coefficients that assumes a form along the lines of the one depicted in Equation 1:

$$\begin{aligned} \hat{y} \quad = \quad & a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \\ & + b_1 x_1^2 + b_2 x_2^2 + \cdots + b_n x_n^2 \\ & c_1 x_1 x_2 + \cdots + c_m x_{n-1} x_n \end{aligned} \tag{1}$$

which can also be represented through the compact linear combination/dot product form in Equation 2, of coefficients β over t model terms x, each one bearing its respective degree of nonlinearity at the exponent:

$$\hat{y} = \sum_{i=1}^{t} \beta_i x_i \tag{2}$$

Specific for the historically usual case of a quadratic response surface with interactions (Barthelemy and Haftka [1993], Kaufman et al. [1996], Sobieszczanski-Sobieski and Haftka [1997]), the true response y (the function that yields all of the observations $\{Y\}$) is approximated by $\hat{y}$ accounting

for all of the independent variables belonging to the n-dimensional vector $\{X\}$ in their pure linear forms (weighted by the coefficients $a_i, i = 1, \ldots, n$), their quadratic forms, as the simplest possible representation of potential curvature/non-linearity (pondered by $b_i, i = 1, \ldots, n$) and also the interactions, in which all pairwise multiplications of the terms in $\{X\}$ are exhausted and pondered by $c_i, i = 1, \ldots, m$ with $m = C_{n,2}$ (combination of the n terms within $\{X\}$, two at-a-time). Optionally, the model also contains a free term or intercept, which approximates the average of the data generating process responsible for creating all possible values of y.

Collecting all the $2n+m+1$ coefficients into the vector of coefficients $\{\beta\}$, the OLS regression method applies the deterministic linear algebra set of operations described in Equation 3, which satisfy the least-squares criterion in Equation 4, in which the sum of residuals (between true and estimated response values) squared is minimized:

$$\{\beta\} = ([X]^T \cdot [X])^{-1} \cdot [X]^T \cdot \{Y\} \tag{3}$$

$$\{\beta\} = \text{argmin} L(\{\beta\}) = \text{argmin} \frac{1}{2N} \sum_N (\{\beta\} \cdot [X] - \{Y\})^2 \tag{4}$$

Noticing that the OLS criterion is indeed an optimization problem in itself (minimizing the loss function $L(\{\beta\})$, with the objective function being a slightly modified version of the sum of squares of residuals, exhibiting the following convenient features:

- Its quadratic nature is amenable to optimization methods;
- Once derived, as part of the optimization procedure, the number 2 in both the exponent and the denominator are mutually canceled;
- The whole expression is normalized by the number of available samples N, avoiding artificial inflation of the sum of squares of residuals for larger sample sizes.

The computational counterpart of Equations 3 and 4 in the R environment is shown in Code 5.

Code 5: Least-square regression

```
library(stats)

x = data.frame(matrix(rnorm(200), nrow=100, ncol=2))
colnames(x) = c('x1', 'x2')
noise = rnorm(100)
y = 2*x[,1] + (0.01*x[,2])^2 + 0.5*noise

model1 = lm(y ~ x1 + x2, data = data.frame(x,y))

z = data.frame(matrix(c(1,1), nrow=1, ncol=2))
colnames(z) = c('x1', 'x2')
fc1 = predict(model1, z)

cat('R2: ', summary(model1)\$r.squared,
  '\nPrediction: ', fc1)

R2: 0.942392
Prediction: 1.896089
```

Equation 4, which makes the underlying optimization procedure explicit, offers important levers for improving the quality of the model, specifically with respect to model/term selection. Based on the principle of parsimony, the effective number of terms in the model will be narrowed down to only the most significance, which improve the bias-variance trade-off in the fashion schematically represented in Figure 3.

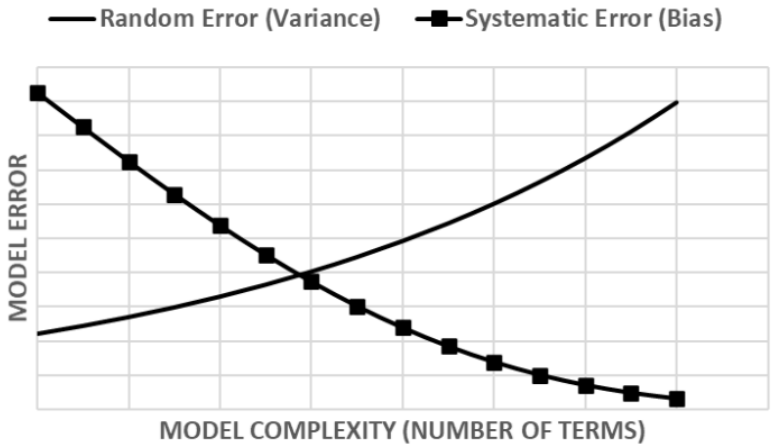


Figure 3: Schematic representation of the bias-variance trade-off for model selection.

Mathematically, the optimal trade-off illustrated in Figure 3 comes out of an extension of Equation 4, also known as regularized regression

$$\min\{L(\beta) + \lambda\,[\alpha\|\{\beta\}\|_1 + (1-\alpha)\|\{\beta\}\|_2]\} \tag{5}$$

where the minimization of the loss-funtion in $\{\beta\}$ becomes a constrained procedure via penalization, which is mediated by the regularization parameter λ and, in its most general form, is applied onto two components, the first $||\cdot||_1$ and second $||\cdot||_2$ norms of the vector of coefficients.

Both $||\cdot||_1$ and $||\cdot||_2$ are minimized when the coefficients with smaller values vanish, which inherently leads to model/term selection. When α equals 1, the penalization is maximized, since the first norm summing up all the terms of $\{\beta\}$ (both the significant ones and those tending to zero) is added to the objective function. When α equals 0, on the other hand, the penalization is not as severe, with the overall summation of coefficients yielding a smaller number, since those with values tending to zero become even smaller once they are squared. These two particular cases of Equation 5 correspond to LASSO and Ridge Regression, respectively, with the acronym LASSO (Least Absolute Shrinkage and Selection Operator) reflecting the most severe penalization case. In practice, it is most usual that $0 \leq \alpha \leq 1$, and the flexibility to pick the best possible value within the unit interval is reflected by the model taking the name of "Elastic Net". The presence λ and α introduce the possibility of regularization (i.e., inherent model selection by dropping the less significant terms) at the expense of increased complexity for the loss function to be minimized, now dependent on these two additional parameters.

Reflecting the enhancement/generalization of Equation 6 over the formulation in Equations 3-4, Code 6 contains the R implementation of the regularized regression approach, with the family of Generalized Linear Models.

Code 6: Generalized linear model with "elasticnet" regularization

```
library(glmnet)

x = matrix(rnorm(200), nrow=100, ncol=2)
noise = rnorm(100)
y = 2*x[,1] + (0.01*x[,2])^2 + 0.5*noise

model2 = glmnet(x, y, beta='elnet')

z = matrix(c(1,1), nrow=1, ncol=2)
fc2 = predict(model2, z)
cat('R2: ', tail(model2\$dev.ratio, 1),
  '\nPrediction: ', tail(fc2[1,], 1))

R2: 0.9405139
Prediction: 1.989429
```

Up to this point, improvements in the model performance is sought quantitatively, that is, deciding how many terms remain in the model. For models of polynomial form, such as those considered far, each term is then a monomial, whose limited flexibility resides mostly on its degree of non-linearity (i.e., exponent) and, even so, typically at the expense of demanding significant additional data. The fundamental limitations of monomials, however, is their stiffness, as illustrated by the Runge phenomenon in Figure 4.

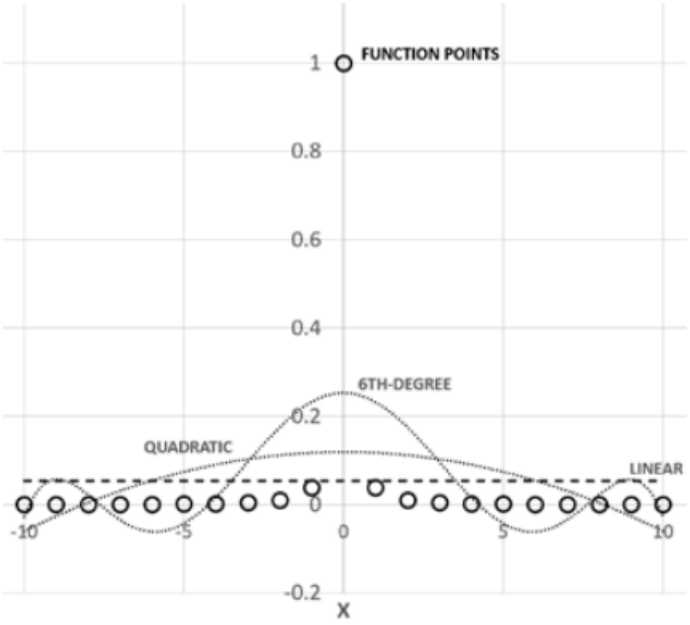


Figure 4: Illustration of the Runge phenomenon, whereby monomial stiffness leads to oscillations when model order is increased in an attempt to capture underlying non-linearity driven by the outlier at 0.

The Runge phenomenon is illustrated by the Runge function described in Equation 6, with the outlier in 0 causing local inflation of the sum of squares of residuals and peripheral oscillations, hence severely limiting the accuracy of a monomial based model which operates as an average

filter (recall that the outlier strongly deviates from that average), not prone to be improved by regularization or, at least, by regularization alone.

$$R(x) = \frac{1}{1 + 25x^2} \tag{6}$$

Several efforts of applied and theoretical nature have been implemented to circumvent these limitations associated with monomial stiffness within the realm of fundamental regression, and the interested reader is encouraged to pursue non-linear regression (Frank [1995]), robust regression (in the sense of immunity to outliers, as detailed in Andersen [2008]) and Generalized Additive models (Hastie [2017]).

However, setting the stage of methods that would become collectively known (and quite successful) as Machine Learning approximations (Hastie et al. [2009]), there is an operating principle that replaces monomials by kernels. Therefore, they are also known as kernel methods, which display the following salient features:

- Parametrization, which result in flexibility (through configuration/optimization of the parameters values) that counters the monomial's inherent stiffness;
- Combination of linear approach (dot product) and non-linear representation (the kernel function);
- In the extreme case, monomials are the simplest possible configuration, where the kernels are just the variables themselves.

4.2 Kernel methods

In its simplest, yet widely used forms, the kernels belong to the family of Radial Basis functions, and is applied for modeling purposes as follows:

1. Collect a sample of size N, comprised of independent variables $[X]$ and dependent variables $\{Y\}$, using the same notation applied for regression methods in Equations 1 to 6;
2. As in Equation 7, calculate the D, which is symmetric and square of order N, containing the Euclidean distances (second norms) between all possible pairwise combinations of independent variables within the sample matrix $[X]$ collected in Step 1. It is important to highlight that the terms in the main diagonal all default to 0 (distances between each point in the sample and themselves) and that, by way of this step, the term-based logic for the model is replaced by a location-based logic, with locations mapped in terms of the distances and, moreover, the distances themselves default to simple scalars irrespective of the original dimensionality of the problem (number of input or independent variables);

$$D = \begin{bmatrix} \|[x]_{1,N}, [X]_{1,N}\| & \|[x]_{2,N}, [X]_{1,N}\| & \cdots & \|[x]_{N,1}, [X]_{1,N}\| \\ \|[x]_{1,N}, [X]_{2,N}\| & \|[x]_{2,N}, [X]_{2,N}\| & \cdots & \|[x]_{N,2}, [X]_{2,N}\| \\ \vdots & \vdots & \ddots & \vdots \\ \|[x]_{1,N}, [X]_{N,N}\| & \|[x]_{2,N}, [X]_{N,N}\| & \cdots & \|[x]_{N,N}, [X]_{N,N}\| \end{bmatrix} \tag{7}$$

3. Transform the distances matrix D by calculating the radial-basis function value for each of its terms $D_{i,j}$. These radial-basis functions, as suggested by their name, assign weights/penalizations to each distance as they depart radially from the points where samples are

available. A number of usual radial-basis functions is listed in Equations 8-13:

$$f_{rb}(D_{i,j}) = D_{i,j} \tag{8}$$
$$f_{rb}(D_{i,j}) = (D_{i,j})^3 \tag{9}$$
$$f_{rb}(D_{i,j}) = (D_{i,j})^2 \cdot \ln(D_{i,j}) \tag{10}$$
$$f_{rb}(\sigma, D_{i,j}) = \exp\left[\frac{-(D_{i,j})^2}{2\sigma^2}\right] \tag{11}$$
$$f_{rb}(\sigma, D_{i,j}) = \sqrt{r^2 + \sigma^2} \tag{12}$$
$$f_{rb}(\sigma, D_{i,j}) = 1/\sqrt{r^2 + \sigma^2} \tag{13}$$

In the order in which they are listed, the radial basis functions in Equations 8 to 13 are the kernels with linear, cubic, spline, gaussian, multi-quadric and inverse multi-quadric shapes, corresponding to different penalization strategies as a function of location distance. For nomenclature purposes, this step is also known as the "kernelization" of the distances in the input space;

4. This step is involved with estimation of the response value at a new point $\{X_{new}\}$ not belonging to the sample $[X]$ collected in Step 1. Since the method is distance based, a total of N Euclidean distances are to be calculated, each of them turned into a scalar that captures a distance which is represented within a space with the same dimensionality of the functional relationship between inputs and output. Equation 14 represents this idea mathematically:

$$D_{X_{new}} = \begin{Bmatrix} \|\{X\}_1, \{X\}_{new}\| \\ \|\{X\}_2, \{X\}_{new}\| \\ \vdots \\ \|\{X\}_N, \{X\}_{new}\| \end{Bmatrix} \tag{14}$$

5. "Kernelize" the distances vector determined in Equation 14, analogously to the application of Step 2 over the matrix in Equation 7;

6. Pre-multiply the "kernelized" distances matrix emerging out of Step 2 by the vector of sampled outputs (dependent variable) $\{Y\}$, observing that they both possess the same number of rows N. This operation yields the so-called weights of the model;

7. Perform the dot product (or linear combination) between the vector of weights obtained in Step 6 and the "kernelized" vector of distances between the inference and the sample points, obtained in Step 5. At this point, because of the linear combination operator applied over "kernelized" entities (in this case, distances within the input space), the philosophies of linear combination (proper of the regression methods discussed between Equations 1 and 6 and kernels are merged together, resulting in a model with a general form that is still linear in its parameters, but flexible to any arbitrary non-linear kernel within each component, as opposed to the stiff nature of the monomials. This combination of advantages has resulted in wide and successful adoption of these models within the realm of Machine Learning, and is summarized by Equation 15 below:

$$Y = \sum_N w_i \cdot f_{rb}(D) \tag{15}$$

Please note that despite Equation 15 representing an inference (i.e., an estimated value), the notation Y is used instead of $\hat{Y}$ to indicate that, in its original form, the family of radial basis

function methods is interpolative, with perfect recall of the response values Y at the N points used training (a condition to be relaxed at advanced variations of the approach to be discussed in the sequence).

Specifically, Equations 12 to 13 correspond to functional forms that depend on additional parameters (namely, σ), whose value(s) are determined either by deterministic linear algebra type of operations (Steps 1 to 7 detailed above), or through an optimization method aimed at minimizing a loss function associated with model accuracy. The latter is the logic underpinning the two methods to be discussed next (Kriging and Gaussian Processes), that derive as enhancements of the fundamental kernel idea.

Hence, introducing a more sophisticated version of the gaussian kernel of Equation 12, a full covariance map is established as a function of a "nugget" (or "jitter") C_0, a "sill" $C_0 + C_1$ and a "range" a. The terms between double quotes reflect the nomenclature of the Variogram based Kriging (also known as Variographic method), in which the smoothness of the input space is mapped according to:

- A discontinuity around each known point (C_0), to account for the possibility of noisy data records. Even in the case of computer experiments, which are deterministic by nature, the introduction of a minor noise level may be beneficial (and in extreme cases, necessary) to reduce the stiffness of matrix inversion operations to be introduced in the sequence;
- A limit for the variance at infinity ($C_0 + C_1$), whereby prediction bounds that measure estimate uncertainties will not grow indefinitely even in the case of extrapolation beyond the input ranges used to train the model. This is a fundamental difference with respect to least-squares based regression and, whereas extrapolation should always be practiced with caution, there is a level of robustness inherent to Kriging;
- A distance (a) beyond which the variance stabilizes as described in the item above

$$C(D) = \begin{cases} C_0 + C_1 & \text{if } |D| = 0 \\ C_1 \cdot \exp\left(\frac{-3D}{a}\right) & \text{if } |D| > 0 \end{cases} \tag{16}$$

Procedurally, the Variographic Kriging method applies as follows:

1. The variographic kernel of Equation 16 is applied over the distance matrix D, generating its "kernelized" version analogously to Step 2 of the Radial Basis approach discussed earlier. The "kernelized" distance matrix is the covariance matrix itself, in which geometrical distances are converted into statistical distances, speaking for the smoothness of the input space. Smoothness is an important consideration because, among the fundamental assumptions of the method, it is considered that points that are statistically close in the input space give rise to points that will also be statistically close in the output space;
2. Involving prediction, this step requires calculation of the input space distance vector between a point whose output is to be predicted, with respect to all known points (the sample used to train the model). This is equivalent to Step 4 of the Radial Basis procedure;
3. Analogously to Step 5 in the Radial Basis discussion, "kernelize" the distance vector obtained in the previous step, converting it into the covariance vector between training and prediction points;
4. Invert the covariance matrix obtained in Step 1. Numerical difficulties in this inversion step may be circumvented by adding a small random noise, which is the "nugget" or "jitter" term of the variographic kernel introduced in Equation 16;

5. Calculate a weights vector by pre-multiplying the inverse covariance matrix obtained in Step 4 by the covariance vector between training and prediction points, calculated in Step 3;

6. Obtain the mean (in the average sense) predictions by multiplying the weights vector by the vector of known outputs from the training sample and summing over the components. This means that the mean predicted value is some form of a linear combination (dot product) of the known outputs, mediated by their respective covariance-driven weights;

7. Repeat Step 6, replacing the vector of known outputs by the covariance vector between training and prediction points obtained in Step 3, to obtain the predictive variance. This is a distinctive feature of the Kriging method in terms of uncertainty quantification. Uncertainty will be higher in regions of the input space with sparse sampling, but will be bounded to a finite value as per the discussion of the "sill" and "range" terms in Equation 16.

Application of Steps 1 to 7 as described indicates two distinctive features that are worth highlighting:

a) while the final estimation step (for both predictive mean and predictive variance) is a linear combination, the kernel is flexible enough to capture a wide gamut of non-linear behaviors, similarly to the discussed via Equation 14 for the Radial Basis approach;

b) a linear algebra based procedure but, given the additional parameters C_0, C_1 and a, their values should be simultaneously optimized to satisfy criteria pertinent to the context, as for example (and very usually) reducing predictive variance.

As far as the optimization of aspect "b" above goes, re-framing the Kriging formulation in terms of its counterpart method known as Gaussian Process (GP) offers additional options in terms of flexibility and performance. Regardless, given the similarity between Kriging and GP, they are often used and named interchangeably, and the following formulation may allude to both:

$$C(D) = \sigma_f^2 \cdot \exp\left(-\frac{D}{2l^2}\right) + \sigma_n^2 \cdot \delta(D) \tag{17}$$

where the covariance structure depends on a functional variance σ_f^2 and a noise term σ_n^2, the variance parameters of two multi-variate gaussian distributions (hence the name of the method) that represent the estimates and the error, respectively. The additional parameter l is the length scale and, overall, the presence of the negative exponent that defaults to 1.0 for distances equal to zero carries over from Equation 12. Lastly, the δ operator introduces the noise term σ_n^2 only outside of the main diagonal of the resulting covariance matrix, obtained with the use of Equation 17 to "kernelize" the geometric distance matrix D, indicating that the method interpolates exactly over the points known from the training sample.

With the above considerations, the task of fitting the model (i.e., identifying the values of the parameters σ_f^2, σ_n^2 and l) morphs into finding the multivariate gaussian distributions whose variances should be equal to σ_f^2 and σ_n^2. In classical statistical practice, the determination of probability distributions that best fit to data is known as Maximum Likelihood Estimation (MLE for short), expressing the procedure of finding the probability distribution(s) that most likely give rise to the observed data (Millar [2011]). For this purpose, a new procedural combination of matrix algebra and parameter optimization ensues as follows:

1. Calculate the covariance matrix by "kernelizing" the geometrical distance matrix D according to Equation 17;

2. Invert the covariance matrix obtained in Step 1;

3. Again involving prediction, use Equation 17 to "kernelize" the vector(s) of distances between known training points and unknown one(s), converting it into the covariance vector between training and prediction points;

4. To obtain the weights, pre-multiply the covariance vector between training and prediction points (obtained in Step 3) by the inverse covariance matrix from Step 2;

5. To obtain the predictive mean, perform the dot product between the weights arising from Step 4 and the output training vector;

6. Multiply the covariance vector between training and prediction points obtained in Step 3 by its transpose (equivalent to the dot product of this vector by itself, hence resulting in a scalar);

7. Subtract the value obtained in Step 6 from any of the equal values that belong to the main diagonal of the covariance matrix obtained in Step 1. The variances go along this main diagonal, and this subtraction is an adjustment that leads into the predictive variance at the inference point.

Once again, matrix algebra operations provide both the predictive mean and variance, that is, an estimate and the associated uncertainty. However, since the entire procedure is influenced by the values of σ_f^2, σ_n^2 and l, they need to be estimated in the MLE sense via Equation 18:

$$\max\left(\log\left(P(y|x,\{\theta\})\right)\right) = -\frac{1}{2}\{y\}^T[C]^{-1}\{y\} - \frac{1}{2}\log(|C|) - \frac{N}{2}\log(2\pi) \tag{18}$$

which indicates that the negative log-likelihood of observing outputs y given inputs x and model parameters $\theta = \{\sigma_f^2, \sigma_n^2, l^2\}$ is maximum for values within θ that influence the 3 terms in the right-hand side of the equation, which depend only on the known training outputs y, the covariance matrix C (where the parameters $\theta = \{\sigma_f^2, \sigma_n^2, l^2\}$ matter) and the sample size N.

Code 7 implements the rather equivalent Kriging/Gaussian Process approaches in R.

Code 7: Kriging model

```
library('DiceKriging')

x = data.frame(matrix(rnorm(200), nrow=100, ncol=2))
colnames(x) = c('x1', 'x2')
noise = rnorm(100)
y = 2*x[,1] + (0.01*x[,2])^2 + 0.5*noise

model3 = km(design = x, response = y)

z = data.frame(matrix(c(1,1), nrow=1, ncol=2))
colnames(z) = c('x1', 'x2')
fc = predict.km(object = model3, newdata = z, type = 'SK')
cat('Prediction: ', fc\$mean)

Prediction: 1.118517
```

Further exploration of the idea of kernels, with model fitting procedures that combine linear algebra matrix manipulations with parameter optimization, allows the opportunity to consider yet another method: neural networks. Although there are numerous variations of the fundamental concept, and significant recent expansion within the specialized application field of Deep Learning (Goodfellow et al. [2016]), the focus on the aforementioned operating principles pivots towards many, and typically simpler kernels, combined with minimization of the predictive error via a mechanism known as back-propagation. Providing materiality to these concepts, consider the (shallow) neural network topology in Figure 5.

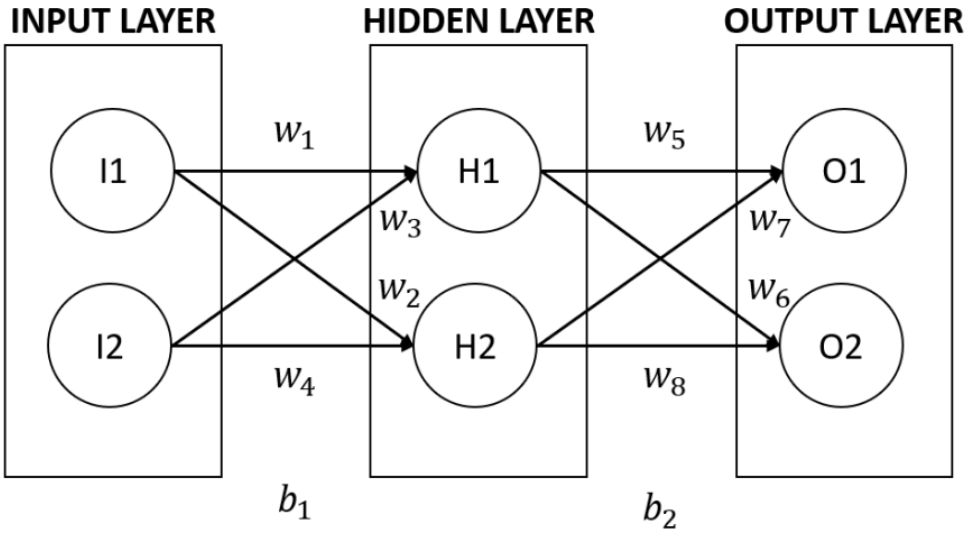


Figure 5: Shallow neural network topology, with input and output layers surrounding one, and only one hidden layer (hence, shallow), all of them containing 2 neurons (kernels) each.

The two input kernels/neurons $I1$ and $I2$ receive the values from the input variables, so that there are as many neurons in the input variables as the dimensionality of the predictive problem itself. The inputs are then transmitted into the hidden layer(s), whereas a single one is capable to map relationships between inputs and outputs at any arbitrary degree of non-linearity.

This transmission step occurs over weighted channels w_1, w_2, w_3 and w_4 such that, in the general case, the actual number of channels corresponds to the number of neurons in the input layer (i.e., the number of input variables) multiplied by the number of neurons in the hidden layer (arbitrarily chosen by the modeler, aiming at balancing bias and variance type of errors).

Over the course of this transmission, an optional bias term b_1 can be added to offset the linear combination of the input values and the transmission weights, as indicated in Equation 19:

$$\begin{aligned} H1_{input} &= w_1 I1 + w_3 I2 + b_1 \\ H2_{input} &= w_2 I1 + w_4 I2 + b_1 \end{aligned} \tag{19}$$

Differently than in the input layer, the neurons/kernels in the hidden layer process their inputs before transmitting them forward, utilizing some non-linear activation function mostly intended to saturate a potentially wide range of variation of the input values into a narrow and normalized (between 0 and 1, or -1 and 1) band. A prevalent choice for this role is the sigmoid function recalled by Equation 20. At this point, the kernel principle whereby a linear combination undergoes

a non-linear transformation is repeated.

$$H1_{output} = K(H1_{input}) = \frac{1}{1 + \exp(-H1_{input})} \tag{20}$$
$$H2_{output} = K(H2_{input}) = \frac{1}{1 + \exp(-H2_{input})}$$

After this non-linear transformation via the kernel K, the same process taking place between the input and the hidden layer takes place between the hidden layer and the output one, with generation of the corresponding input via Equation 19, but utilizing weights and bias w_5, w_6, w_7, w_8 and bias b_2 as indicated in Figure 5, and output via Equation 20.

The final output thus produced is compared to the true output values y_1 and y_2 at the 2 arbitrarily chosen neurons/kernels at the output layer, generating an error signal that triggers the core of the neural networks model building: the backpropagation method. The naming is owed to the fact that the error is back-propagated into the previous layers, in reverse order, and their respective weights and biases are re-adjusted so that in the next forward pass (again involving Equations 19 and 20 across all layers, as described above), the final error is reduced. The process repeats until convergence, and the following detailed backpropagation steps apply for a network with the topology as in Figure 5 and a neuron/kernel activation function as in Equation 20:

1. Calculate the total error in the differentiable form of the quadratic loss function in Equation 21, accumulating over the i neurons in the output layer:

$$E_{total} = \frac{1}{2} \sum_i (y_i - O_{output,i})^2 \tag{21}$$

2. Express the sensitivity of the total error calculated in Equation 21 with respect to the weights of the output layer, applying the chain rule of derivation as in Equation 22,

$$\frac{\partial E_{total}}{\partial w} = \frac{\partial E_{total}}{\partial O_{output}} \cdot \frac{\partial O_{output}}{\partial K} \cdot \frac{\partial K}{\partial w} \tag{22}$$

   where, in this specific sequence: the total error depends on the outcome of the output layer, which depends on the kernel transformation at the same layer, which depends on the input it received based on the application of the weights from the previous (hidden) layer.

3. Calculate the 3 derivative terms of the chain rule in Equation 22 which, for a sigmoid activation/kernel function as in Equation 20, results in Equations 23-25 below:

$$\frac{\partial E_{total}}{\partial O_{output}} = -(y_i - O_{output,i}) \tag{23}$$
$$\frac{\partial O_{output}}{\partial K} = O_{output,i} \cdot (1 - O_{output,i}) \tag{24}$$
$$\frac{\partial K}{\partial w} = H_{output,i} \tag{25}$$

4. Multiply the 3 terms of the chain derivative expressed in Equations 23-25 to calculate the expression stated in Step 1;

5. Update the weights by using any of the optimization methods discussed in chapter "An overview of Linear and Non-linear Programming methods for Structural Optimization" (Choze et al. [2022b]), given the availability of its total amount (Equation 21, as calculated in Step 1) and its sensitivities in analytical form (Equations 23-25, corresponding to

Step 3). Although any optimization method can be used, the simplest line search in the direction of the gradient with step size/learning rate α, as expressed in Equation 26,

$$\Delta w = -\alpha \frac{\partial E_{total}}{\partial w} \tag{26}$$

is the most usual, given the convenience of the derivatives availability (some implementations will make the value of α adaptive for efficient convergence). On the other end of the spectrum, Boltzmann Machines is the usual nomenclature for neural network models fitted when genetic algorithms are used to perform this step.

6. Run the backpropagation of errors one further layer backwards, that is, repeat the procedure between Equations 21 and 26 for the connection between the input and the hidden layer. Because the total error actually carries over from the output layer, an extra implicit derivative must be accounted for, and the term corresponding to Equation 23 is further unpacked as shown in Equation 27, for every neuron j in the hidden layer and i in the output layer:

$$\frac{\partial E_{total}}{\partial H_{output,j}} = \sum_{j} \frac{\partial E_{output,i}}{\partial H_{output,j}} \tag{27}$$

and each term in the summation is a repetition of Equation 20, implicit within its first term (Equation 23). The two remaining terms are calculated as before, per Equations 24 and 25, giving rise to the sensitivity of the total error with respect to the weights;

7. Repeat Step 5 with the sensitivity of error versus weights obtained in Step 6, finalizing the backward pass and getting ready to initiate the next forward pass (Equations 19 and 20) if the method is to proceed for an additional iteration.

Requirements for the amount of training data and number of iterations until successful convergence vary on a case-dependent basis, but tend to be high as compared to other methods. Nevertheless, the wall clock time for training neural networks is reduced due to the simplicity of each of the individual operations.

As a concluding comment about neural networks, the arbitrary number of neurons in the hidden layer is equivalent to the number of terms in polynomial least-squares regression models (Equations 1 to 5), and hence should be chosen to balance between over-fitting (excessive number of terms leading to increased variance driven predictive errors) and under-fitting (insufficient number of terms leading to bias driven predictive errors). The practitioner is especially cautioned relative to a natural tendency for over-fitting, barring extreme low number of hidden neurons, and appropriate metrics to gauge this phenomenon are offered in the next section about goodness-of-fit. Many variations of the basic Neural Network method are available in the R statistical computing platform, with one of them illustrated in Code 8.

Code 8: Radial Basis Function (RBF) network

```
library(RSNNS)

x = matrix(rnorm(200), nrow=100, ncol=2)
noise = rnorm(100)
y = 2*x[,1] + (0.01*x[,2])^2 + 0.5*noise

model4 = rbf(x, y, size=40, maxit=1000)

z = matrix(c(1,1), nrow=1, ncol=2)
predict(model4, z)

1.864962
```

5 Ensemble Methods

Whereas each one of the modeling approaches described in section 4.1-4.2 individually possess their benefits and drawbacks, the practice of building model ensembles assumes that any individual models are "weak predictors", which can be enhanced by some combination strategy capable of collectively downplay the shortcomings and amplify the strengths.

Operationalization of models ensembling strongly relies upon the theory of statistical resampling (Chihara and Hesterberg [2018]) whose primary techniques, the jackknife (deterministic sub setting, performed without replacement) and the bootstrap (random sub setting, performed with replacement) both leverage the most fundamental topic of sampling distributions (Devire [2015]). In particular, bootstrapping is the preferred method for mainstream model ensembling due to its ability to balance sample size (recalling that shortage of data is more likely than not), introduction of variability for model diversity (otherwise the very purpose of ensembling is defeated) and, depending upon specific circumstances, independence of the sub-samples. Within this construct, ensemble modeling techniques fall within the following categories.

5.1 Parallel ensembling

The main steps are:

1. Perform the bootstrapping, extracting S sub-samples out of N total available points;
2. Fit S independent (weak) learners, utilizing each of the individual subsets obtained in Step 1;
3. Summarize the S independent (weak) learners into a combination, which is typically and average for predictive models whose output is a continuous random variable and either an average (soft combination) or a majority vote (hard combination) if the output is categorical.

When performing Step 1, the goal is to generate as many subsets as possible, provided that each one is still large enough to allow for reasonable individual models to be produced, and at the same time minimizing the correlation (i.e., repetition of data that gets resampled with replacement) across samples. Certainly, satisfactory fulfillment of all these conflicting requirements depends upon a large enough N to start with.

For Step 2, it is usual to choose each independent learner to be of the same type before they are "bootstrap aggregated" or "bagged" in Step 3. This homogeneity does not mean that the intended diversity of the ensembling is compromised, not only because each individual learner arises from a different (and ideally independent) data set, but also due to parameter choices that can add to achievement of this goal. For example, it is common to balance individual learners that overfit (residing at the right hand end of the spectrum in Figure 3) and underfit (residing at the left hand end of the spectrum in Figure 3) so that the resulting ensemble best balances the bias-variance trade-off. Moreover, it is also common practice to fit the individual weak learners with different feature sets (i.e., different versions of the matrix of independent variables $[X]$) so that additional diversity is obtained.

Last but not least, the most important aspect of Step 3 is finding the optimal weighting strategy for each individual weak learner, which is possible by applying variations of Equations 4 and 5 when regularization is or is not considered, respectively. In either case, the weights are decision variables for minimizing a suitable loss function, ideally out of an independent data set used exclusively for testing purposes (additional details on candidate metrics presented in section 6 about goodness-of-fit appraisal).

5.2 Sequential ensembling

Steps 1 and 2 for sequential ensembling occur as in the parallel methods, and the fundamental difference in Step 3 is that, instead of solving a single instance of weights optimization, many of them are resolved in sequence as in Equation 28:

$$EM_i = EM_{i-1} + w_i \cdot WM_i \tag{28}$$

where the ensemble model EM at the i-th iteration is updated from the previous one by adding a weighted contribution of one of the possible weak learners WM, such that the loss function is minimized. Because the addition of each further learner "boosts" the previous one(s), and this is done adaptively at each iteration, this approach is often referred to as "adaptive boosting" or "adaboost" for short, and can be generalized as in Equation 29 for a loss function L:

$$\text{argmin}_{w,WM} \left(\sum_{i=1}^{N} L \left[\hat{Y}, EM_{i-1}(X) \right] + w_i \cdot WM_i \right) \tag{29}$$

When the steepest descent (Vanderplaats [1998]) optimization method is applied to solving Equation 29, relying solely on information about the gradient of the loss function from any iteration to the next, the so-called gradient boosting method (GBM, for short) is configured, and Equation 30 represents it accordingly:

$$EM_i = EM_{i-1} - w_i \cdot \nabla E_{i-1} \tag{30}$$

where the weights w_i function as the steps to be taken along the line search following the direction of the gradient of the error E in the previous ensemble configuration. It is assumed that the loss function is continuously differentiable and, when it incorporates regularization terms, along the lines of Equation 5, the nomenclature xgboost (for "extreme" gradient boosting) is applied.

5.3 Multi-level ensembling

A common thread between parallel and sequential ensembling strategies is that 1) both processes are guided through minimization of a loss function and 2) it is typical, although not mandatory,

that models of the same type are combined together. Conversely, multi-level ensembling techniques will 1) fit a meta-model, at another level (hence the nomenclature), combining all existing learners to match given response values and 2) typically, although not mandatory, combine models of different types in the process. Equation 31 represents the idea in matrix notation, with the solution of the system of k simultaneous equations yields the coefficients/weights w that combine alternative weak learners to best match the ground truth $\{Y\}$.

$$\begin{bmatrix} w_1 \cdot \hat{Y}_1^{WM_1} & w_2 \cdot \hat{Y}_1^{WM_2} & \cdots & w_k \cdot \hat{Y}_k^{WM_k} \\ w_1 \cdot \hat{Y}_2^{WM_1} & w_2 \cdot \hat{Y}_2^{WM_2} & \cdots & w_k \cdot \hat{Y}_k^{WM_k} \\ \vdots & \vdots & \ddots & \vdots \\ w_1 \cdot \hat{Y}_k^{WM_1} & w_2 \cdot \hat{Y}_k^{WM_2} & \cdots & w_k \cdot \hat{Y}_k^{WM_k} \end{bmatrix} = \begin{Bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_k \end{Bmatrix} \tag{31}$$

Reflecting the increased popularity of ensemble surrogates, the R statistical platform incorporates many of the methods, with Code 9 containing the Random Forest in particular:

Code 9: Random Forest algorithm

```
library(randomForest)

x = matrix(rnorm(200), nrow=100, ncol=2)
noise = rnorm(100)
y = 2*x[,1] + (0.01*x[,2])^2 + 0.5*noise

model5 = randomForest(x, y, ntree = 500)

z = matrix(c(1,1), nrow=1, ncol=2)
predict(model5, z)

1.905261
```

6 Goodness-of-Fit Appraisal

Complete appraisal of goodness-of-fit depends on two complementary angles: model recall (its ability to accurately estimate the outputs in training data) and generalization (its ability to accurately estimate the outputs in independent test data, not utilized during training). Whereas generalization is evidently more important for a final determination of the model qualities, recall analysis should be also considered as a tool for building understanding, helpful to improve the model overall. For this purpose, the foremost recall metric is the coefficient of multiple determination, shown in Equation 32:

$$R^2 = \frac{SS_R}{SS_Y} = 1 - \frac{SS_E}{SS_Y} \tag{32}$$

It defines the percent ratio between the explained variance (by the model) and the total variance, and is calculated as follows:

1. Extract the difference between all known training output Y and their average. Square these differences and add them together to obtain SS_Y;

2. Repeat Step 1, but for the differences between actual and estimated (by the model) Y. These model errors (E) are the residuals, and this step obtains their sum of squares;

3. Using the principle of variance additivity, obtain the sum of squares of regression (R) by subtracting the term in Step 2 from that in Step 1;

4. Possessing all terms in Equation 32, apply it to calculate the coefficient of multiple determination.

The adjusted coefficient of multiple determination, on its hand, places effort on recognizing the principle of parsimony, and penalizes the metric in Equation 32 as the model acquires additional terms (i.e., operates on higher dimension input space). Denoting the number of model terms as p, it is possible to use this adjusted form of the coefficient of multiple determination to understand how it balances with respect to the available number of training samples N:

$$R_a^2 = 1 - \left(\frac{N-1}{N-p}\right) \cdot \left(1 - R^2\right) \tag{33}$$

The structure of Equation 33 is such that the need to fit additional parameters from a fixed training sample size is penalized, and the effect is similar to that from Equation 5 which regularizes the regression by eliminating less or non-essential terms.

Supplementing Equations 32 and 33, the Analysis of Variance (ANOVA for short) offers an additional way into the sources of error from random variation and also terms in the model/dimensionality of input space, as shown in Table 5.

Table 5: Analysis of Variance (ANOVA) for measuring goodness-of-fit.

Source of Variability	Statistical degrees-of-freedom (dof)	Sums of Squares (SS)	Mean Squares (MS)	F-Statistic
Model	$p-1$	Step 3, R^2 calculation		
Residuals	$N-p$	Step 1, R^2 calculation	$\frac{SS}{dof}$	$\frac{MS_{model}}{MS_{residual}}$
Total	$N-1$	Step 2, R^2 calculation		

It should be noted that Mean Squares are actually variances, and the F-Statistic, as a ratio between variances, follows the Snedecor probability distribution and, in this context, measures the probability that the model is only able to explain an amount of the total variance which is statistically equivalent to the residuals variance. In other words, the F-Statistic measures the probability that the model can only explain as much of the data as the error itself, and is hence useless. With this interpretation, adequate goodness-of-fit requires this number to be as low as possible, contrary to the coefficient of multiple determination and its adjusted form.

The possibility of calculating model generalization metrics assumes that there are independent observations, not consumed for the purposes of model training, that can be spared. This tends to be a strong assumption, considering the considerations made in section 3 on experimental design and the challenges of balancing data availability and model quality. For this reason, traditional partitions such as sparing some percentage (typically 20% - 30%) of the overall data for testing will only perform when there is abundance of data, usually not the case with high fidelity simulations of structures. The solution then resides in Cross-validation, which operates according to the following algorithm:

1. Create a fold, by sparing a small percentage of the available data for testing, the rest being dedicated to training;

2. Train the model with the training data portion (majority) of the fold determined in Step 1;

3. Test the model with the testing data portion (minority) of the fold determined in Step 1 and calculate performance metrics;

4. Repeat Steps 2 and 3 until all possible folds are exhausted;

5. Analyze the distribution of the performance metrics calculated in Step 3 over the many folds.

This approach assumes that there will be no replacement of the data across the folds and, at the limit, the LOOCV (Leave-One-Out-Cross-Validation) will happen when only one data point is retained for testing within each fold. This is the most time-consuming option, but the one that yields the most complete possible version for the distribution of whatever performance metric is chosen in Step 3. As for the metrics themselves, the RMSE (Root Mean Squared Error) is a typical choice, and it is defined in Equation 34 as the square-root of the average quadratic error between true values (Y) and their model estimates ($\hat{Y}$):

$$RMSE = \sqrt{\frac{\sum_N \left(\hat{Y} - Y\right)^2}{N}} \tag{34}$$

It should be noted that RMSE is not the only metric applicable to appraise model generalization, albeit it is by and large the most used one, more so in the context of cross-validation described herein. Other alternatives consist of calculating variance explained type of metrics, similar to those in Equations 32 and 33, but using data that was not directly used to build the model in the first place. Hence, it is conceivable to calculate the sum of squares of test residuals (i.e., those arising from independent data) as inputs to such equations for estimating the portion of the overall variance not explained by the model. Caution should be exercised though with the basis for comparison, which should remain standardized so that such metrics are meaningful. One of the approaches to address this concern is to use the "out-of-sample" (OOS) version of Equation 35, which is defined as follows:

$$R^2_{OOS} = 1 - \frac{RMSE_m}{RMSE_b} \tag{35}$$

and depends upon $RMSE_m$, calculated for the model using Equation 34, and an ideal (possibly arbitrarily set) benchmark $RMSE_b$ for a real or hypothetical model. It should be noted that, contrary to the $RMSE$ defined in Equation 34, $-\infty \leq R^2_{OOS} \leq 1$ and a value of 0 represents a tie between the model being appraised and the benchmark. Conversely, positive/negative values signify that the model outperforms/underperforms the benchmark.

A totally different perspective, which is more solid but difficult to automate, counts on on understanding the error structure of the models in detail rather than relying solely upon summary metrics such as any of those offered in Equations 32 to 35 and Table 5. Models that have exhausted all possible information contained in the data yield errors that behave as random numbers, ideally centered around zero (indicating absence of bias) and with small spread (indicating absence of variance). These two features allude to the error components depicted in Figure 3 and it expected that one can only be improved at the expense of the other, forcing a trade-off. Nevertheless, the two characteristics can be combined to some extent when the errors are Normally distributed (i.e., behave as random numbers that follow the Normal probability distribution with mean 0 and some variance), and then the exercise of appraising goodness of fit defaults to measuring deviation/adherence of the modeling errors with respect to the Normal reference.

Following this rationale, specific techniques may be more visual, such as those illustrated in Figures 6, 7 and 8, or rather parametric (and easier to automate), such as the tests of Normality summarized in Table 6.

Figure 6 shows distinct scenarios in which the residuals distributions vary from closely adhering to random (Figure 6a) to exhibiting multiple symptoms of Normality violation (Figure 6b). In the former, uniformly scattered points representing the residuals spread symmetrically around zero, without revealing any relevant pattern indicative of non-randomness. In the latter, on the contrary, multiple patterns arise by way of repeated ascending/descending patterns, as well as lack of symmetry and outliers, indicating that multiple processes are manifesting themselves in tandem, rather than a single random distribution representative of a good model fit. The particular case explored in Figure 6c represent the case in which the model lacks complexity to represent a linear trend, and is hence under-fitted. Generalizations of this concept could show very visible and identifiable patterns such as parabolas or other shapes, which would be associated with even more severe under-fitting other than just missing a linear trend. Finally, the scenario depicted in Figure 6d is the most subtle and potentially damaging one, because an apparently random residual distribution is actually strongly biased towards positive. Realistic applications will pose the additional challenging of combining all these types of archetypal behaviors into a single residual distribution, making it very complex to issue an appraisal about goodness of fit.

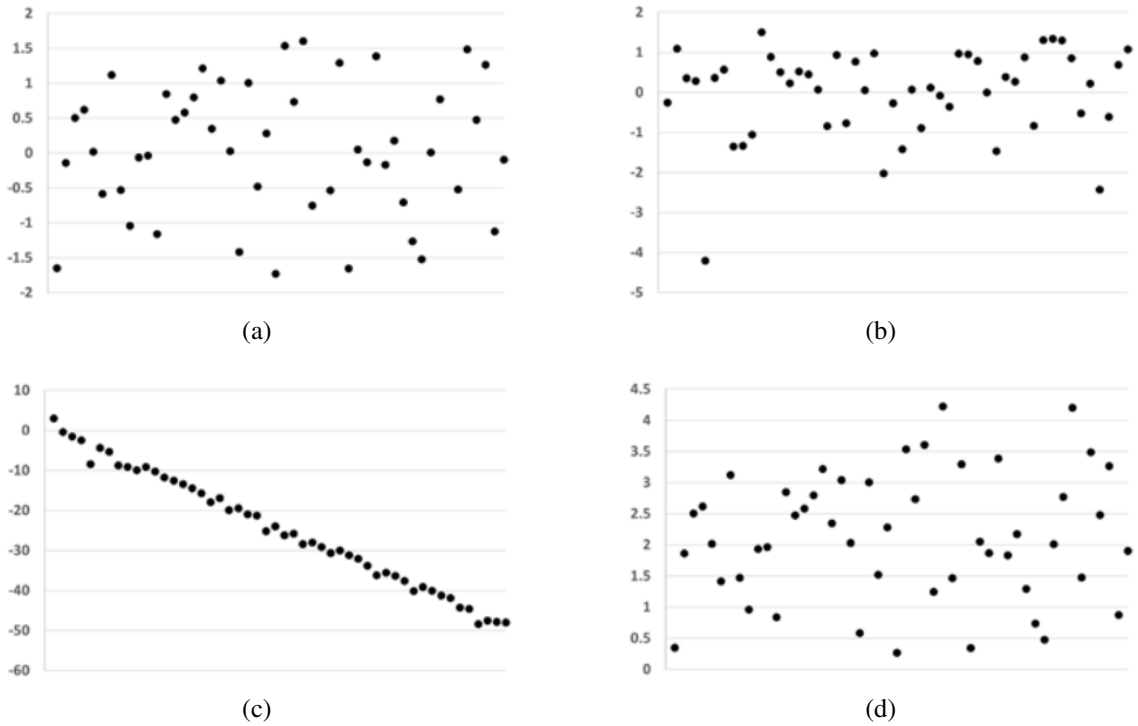
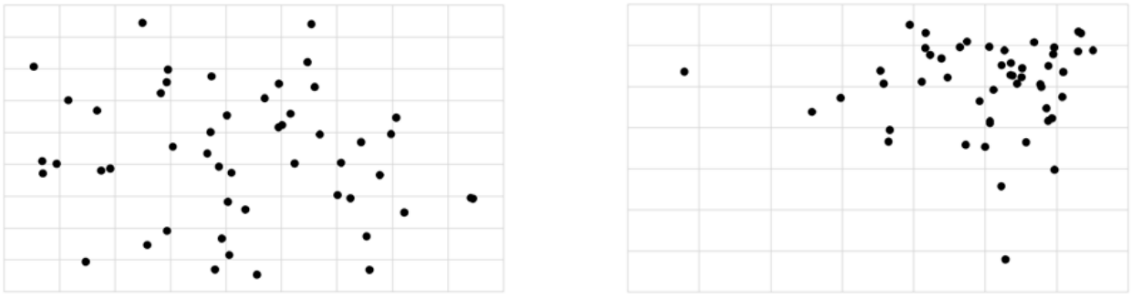


Figure 6: Different patterns of residuals for goodness of fit analysis.

Panels (6a) and (6b) in Figure 6 are in opposite ends of the spectrum with respect to residuals that closely adhere to adequate versus poor goodness of fit, respectively. Since such verdicts are seldom this binary, Figure 7 offers an additional appraisal instrument by plotting the residuals versus lagged versions of themselves. With $lag = 1$ in these particular plots, it is possible to confirm the absence of randomness in panel (7b), which corresponds to the same panel in Figure 6 and, at the same time, certain patterns arise in panel (7a), again homologous to the same in Figure 6. For Figure 7, the lagged plot is powerful enough to reveal that the data from Figure 7a is actually a random draw from a Normal distribution with mean 0 and variance 1 ($\mathbb{N}(0, 1)$) distribution with only 50 points, which are still not sufficient to display full blown randomness as it would have been the case with a much larger sample size, but still satisfactorily random/Normal on a relative

scale.



(a) (b)

Figure 7: Plot of lagged residuals for goodness of fit analysis.

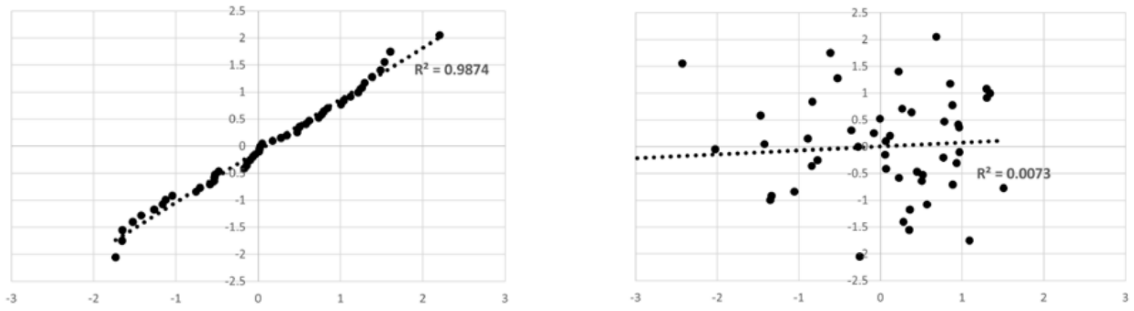
On a related note, the lagging period can be generalized to other values and, at the limit, their collection at several values gives rise to auto-correlation and partial auto-correlation plots, which are paramount in time-series analysis (Shumway and Stoffer [2017]) and, in the current context, would reveal additional details about the residuals structure for the purposes of model appraisal.

Figure 8 again looks at the comparison between panels (a) and (b) from Figures 6 and 7, however analyzing the very same data through a different instrument: the (Normal) probability plot, which are generated as follows:

1. Sort the residuals in ascending order;
2. Estimate the probability of each residual value based on its rank order by dividing the position in the sorted sequence by the total number of residuals N. Per this method, the estimated rank order probability for the n-th residual in the sequence is n/N, and the last value in the series results in 1 (100% of the residuals covered, producing an empirical cumulative probability distribution);
3. Calculate the z-scores (distances from the mean normalized by the standard deviation) using the probabilities obtained in Step 2 and the inverse cumulative version of the distribution of reference. If the inverse Normal cumulative is chosen, the empirical Normal z-scores are obtained;
4. Calculate the adherence of the cloud of z-scores obtained in Step 3 to the theoretical distribution (Normal, in the current context) by fitting a straight line and measuring the goodness of fit through the R^2 (as in Equation 32).

Specifically, in Figure 8, panel (a) confirms a highly satisfactory adherence to normality (R^2 nearing 100%), whereas the various violations of Normality observed in panel (b) in Figures 6 and 7 are confirmed by a very low R^2.

The way of measuring adherence to Normality through a metric, like the R^2 in Figure 8, is extending the concept through formal Hypothesis testing, whereby a statistic is calculated and compared to a critical value to determine if the "null hypothesis" (residual data follows the reference distribution, to be assumed as Normal) can be accepted and at which confidence level. Several versions of this principle are summarized in Table 6, which lists the most commonly used tests of Normality, applicable to the specific purpose of measuring goodness-of-fit through the $i = 1, \ldots, N$ available values of residuals denoted as ε.



(a) (b)

Figure 8: Normal Probability Plot, also known as quantiles plot, for goodness of fit analysis.

In every case, F and C are the reference (i.e., standard Normal with zero mean and unit variance) and empirical (i.e., from the residual data itself) cumulative probability distribution functions (again, their comparison relates to the procedure used to produce Figure 8).

To conclude this section on goodness-of-fit, residuals analysis is presented as the ultimate ground truth in appraising if a model is or is not suited for a particular application. Even for model types such as Kriging and Gaussian Processes, which possess inherent uncertainty quantification features as displayed at their respective descriptions (Equations 16 to 18 and associated content), the application of the metrics shown in this section, and more emphatically cross-validation and residual analysis, are thoroughly recommended to the practitioner.

Table 6: Collection of usual tests of hypothesis to assess goodness of fit through Normality.

Anderson-Darling Can be specified to reference distributions other than Normal by generalizing F:

$$A^2 = -N - S$$

$$S = \sum_{i=1}^{N} \frac{(2i-1)}{N} \cdot [\ln F(\varepsilon_i) + \ln(1 - F(\varepsilon_N + 1 - i))]$$

Liliefors Utilizes the supremum ($\sup$) operator, which finds the lowest upper bound of a set of values:

$$L = \sup_{\varepsilon} |F(\varepsilon) - C(\varepsilon)|$$

Kolmogorov-Smirnov Can be specified to reference distributions other than Normal by generalizing F:

$$D = \max_{1 \leq i \leq N} \left(F(\varepsilon_i) - \frac{i-1}{N}, \frac{i}{N} - F(\varepsilon_i) \right)$$

Kulbach-Leibler Divergence In this formula, already particularized for the Normal case, but also possible to be generalized depending on the density function chosen for pre-multiplying the natural logarithm and appearing in its argument's denominator. The lower case $c(\varepsilon)$ in the natural logarithm argument's numerator is the empirical density:

$$D_k(P|\mathbb{N}) = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{\varepsilon-\mu}{\sigma}\right)^2} \cdot \ln \left[c(\varepsilon) \cdot \left\{ \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{\varepsilon-\mu}{\sigma}\right)^2} \right\}^{-1} \right] d\varepsilon$$

Jarque-Bera Strictly for Normality test, does not depend on a reference distribution, but rather on the residuals skewness S and kurtosis k:

$$JB = \frac{n}{6} \left[S^2 + \frac{1}{4}(k-3)^2 \right]$$

Chi-Squared The observed (O_i) and expected (E_i) terms correspond to the number of residuals counted within a certain bin or range i (expected are per the Normal distribution for using residuals to appraise goodness of fit, but may also be generalized to other distributions). The χ^2 itself is a random variable arising from the sums of squares of Normals, as implied by the formula:

$$\chi^2 = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{E_i}$$

References

NIST/SEMATECH e-Handbook of Statistical Methods. URL https://www.itl.nist.gov/div898/handbook/pri/section3/pri3347.htm.

R. Andersen. *Modern methods for robust regression*. Number 152. Sage, 2008.

J.-F. Barthelemy and R. T. Haftka. Approximation concepts for optimum structural design - a review. *Structural optimization*, 5(3):129–144, 1993.

A. Bhosekar and M. Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers and Chemical Engineering*, 108:250–267, 2018.

F. Boukouvala and C. A. Floudas. ARGONAUT: AlgoRithms for Global Optimization of coNstrAined grey-box compUTational problems. *Optimization Letters*, 11(5):895–913, 2017.

G. E. P. Box, J. S. Hunter, and W. G. Hunter. *Statistics for Experimenters: Design, Innovation, and Discovery*. Wiley-Interscience, 2nd edition, 2005.

L. M. Chihara and T. C. Hesterberg. *Mathematical Statistics with Resampling and R*. Wiley, 2nd edition, 2018.

S. B. Choze, R. R. Santos, and G. F. Gomes. Overview of traditional and recent heuristic optimization methods. In *Model-Based and Signal-Based Inverse Methods*, volume 1 of *Discrete Models, Inverse Methods & Uncertainty Modeling in Structural Integrity*, chapter 4, pages 107–142. University of Brasilia, 2022a. URL https://doi.org/10.4322/978-65-86503-71-5.c04.

S. B. Choze, R. R. Santos, A. B. Jorge, and G. F. Gomes. An overview of linear and non-linear programming methods for structural optimization. In *Model-Based and Signal-Based Inverse Methods*, volume 1 of *Discrete Models, Inverse Methods & Uncertainty Modeling in Structural Integrity*, chapter 3, pages 65–106. University of Brasilia, 2022b. URL https://doi.org/10.4322/978-65-86503-71-5.c03.

J. L. Devire. *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 9th edition, 2015.

I. E. Frank. Modern nonlinear regression methods. *Chemometrics and intelligent laboratory systems*, 27(1):1–19, 1995.

K. I. A. Garud, S. S. and M. Kraft. Design of computer experiments: A review. *Computers and Chemical Engineering*, 106:71–95, 2017a.

K. I. A. Garud, S. S. and M. Kraft. Smart sampling algorithm for surrogate model development. *Computers and Chemical Engineering*, 96:103–114, 2017b.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. The MIT Press, illustrated edition, 2016.

T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

T. J. Hastie. Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge, 2017.

M. Kaufman, V. Balabanov, A. A. Giunta, B. Grossman, W. H. Mason, S. L. Burgee, R. T. Haftka, and L. T. Watson. Variable-complexity response surface approximations for wing structural weight in hsct design. *Computational Mechanics*, 18(2):112–126, 1996.

A. I. Khuri and J. A. Cornell. *Response Surfaces: Designs and Analyses*. CRC Press, 2nd edition, 2019.

R. B. Millar. *Maximum Likelihood Estimation and Inference: With Examples in R, SAS and ADMB*. Wiley, 1st edition, 2011.

R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley Series in Probability and Statistics, 4th edition, 2016.

L. Pronzato and W. G. Müller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, 2012.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL https://www.R-project.org/.

R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer International Publishing, 4th edition, 2017. URL https://doi.org/10.1007/978-3-319-52452-8.

J. Sobieszczanski-Sobieski and R. T. Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, 14(1):1–23, 1997.

G. N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design*. Vanderpaats Research and Development, 1998.