Solving inverse problems in thermal engineering using a surrogate model

SARRA FAKHFAKH
Solving inverse problems in thermal engineering using a surrogate model

SARRA FAKHFAKHAH
"No man can reveal to you aught but that which already lies half asleep in the dawning of our knowledge.

The teacher who walks in the shadow of the temple, among his followers, gives not of his wisdom but rather of his faith and his lovingness.

If he is indeed wise he does not bid you enter the house of wisdom, but rather leads you to the threshold of your own mind.

The astronomer may speak to you of his understanding of space, but he cannot give you his understanding.

The musician may sing to you of the rhythm which is in all space, but he cannot give you the ear which arrests the rhythm nor the voice that echoes it.

And he who is versed in the science of numbers can tell of the regions of weight and measure, but he cannot conduct you thither.

For the vision of one man lends not its wings to another man.

And even as each one of you stands alone in God’s knowledge, so must each one of you be alone in his knowledge of God and in his understanding of the earth."

- Khalil Gibran, The Prophet
Through either measured or computed experimental data, inverse problems aim to determine parameters not straightforwardly given by measure. An inverse problem results in an optimization problem that requires many simulations of the direct problem which computations can be costly. One approach is to replace the reference model by a reduced model. A reduced model or surrogate is built by a statistical learning method (a theory on how to characterize the behaviour of a function based on observed data). In this case, uncertainties have a bigger effect on the problem and the errors introduced by the surrogate can significantly alter the convergence process. Furthermore, it is well known that the determination of input parameters via observed ones is an ill conditioned problem. As a result, the slightest measuring errors can engender tremendous gaps in the values of the reconstructed parameters, thus ruining their use. We can remedy that by adding a penalty in the optimization problem which would ensure a better stability during its resolution.

Identification problems are here solved using a surrogate model. The issues of this approach on the resolution of the reference model are evaluated. Other specific substitution models known to be less reliable but better suited to inverse problems due to a regularization expression are constructed.

Classical optimization methods including a penalty for the resolution of identification problems are implemented. The impact of the reliability of the surrogate model on the robustness and accuracy of the resolution is then carried out. In order to improve the surrogate model fidelity, sequential enrichment of the design of experiments is applied. Finally, the methodology is tested on a simplified thermal engineering example: the one dimensional heat conduction problem.
LÖSNING AV ETT OMVÄNT PROBLEM INOM TERMISK TEKNIK MED HJÄLP AV EN SURROGATMODELL

Abstrakt


 Identifieringsproblemet lösas här med hjälp av en surrogatmodell. Frågan om denna strategi påverkar lösningsprocessen av referensmodellen utvärderas. Andra specifika substitutions modeller, som är kända för att vara mindre tillförlitliga men bättre lämpade för inversa problem på grund av ett regulariseringsuttryck, är konstruerade.


Acknowledgements

I would like to express all my gratitude to my supervisor Dr. Patricia Klotz for giving me the opportunity to work on this challenging master thesis subject and for guiding me through it.

I would also like to thank my co-supervisor Dr. Vincent Mouysset for his valuable advice and constructive explanations all along this internship.

And of course, I thank my university supervisor Prof. Dr. Per Enqvist for his help in writing this report, for his proofreading and for his help with the administrative work at KTH.

Last but not least, I would like to thank all the PhD students with whom I had lunches and spent break time with for their support. I especially thank Guillaume Davy for his help with the software Python and Kevin Delmas for making the internship constraints bearable.
Contents

Abstract ii

Acknowledgements iv

List of Figures vii

List of Tables viii

Abbreviations ix

1 Introduction 1
  1.1 General context and goals ................................................. 1
  1.2 Related research work .................................................. 2
  1.3 Outline ................................................................. 3

2 Inverse problem resolution 5
  2.1 Problem Statement ......................................................... 5
    2.1.1 Writing an inverse problem ....................................... 5
    2.1.2 The heat conduction forward problem ............................. 6
    2.1.3 Test case .......................................................... 7
    2.1.4 Gradient-based optimization for non regularized formulation ... 8
  2.2 Regularized problems ...................................................... 10
    2.2.1 Inverse problem formulation with regularization ................. 11
    2.2.2 Gradient-based optimization for regularized formulations .......... 11
    2.2.3 Derivative-free method for regularized problems .................. 19
  2.3 Results discussion ....................................................... 22

3 Kriging models construction and kriging based optimization 26
  3.1 Building a surrogate: Kriging model ................................... 26
    3.1.1 Construction of the Kriging model ................................ 27
    3.1.2 Kriging prediction ............................................... 30
  3.2 Construction of a DOE by sequential enrichment ........................ 32
    3.2.1 Method description .............................................. 32
    3.2.2 Test functions and results ....................................... 34
  3.3 Efficient Global Optimization (EGO) ................................... 37
    3.3.1 Algorithm presentation .......................................... 37
    3.3.2 Test functions results ........................................... 38

4 Inverse problem resolution using surrogate models 40
  4.1 N dimensional surrogate-based optimization ............................ 40
    4.1.1 Problem statement and method description ........................ 40
    4.1.2 Results discussion ............................................... 42
  4.2 N+1 dimensional surrogate-based optimization .......................... 42
    4.2.1 Problem statement and method description ........................ 42
4.2.2 Results and discussion ........................................... 43
4.3 Surrogate-based optimization with DOE enrichment ........ 45
  4.3.1 Sequential use of EGO algorithm ............................. 45
  4.3.2 Method results ................................................. 46

5 Improvements and discussion ..................................... 49
  5.1 Discussing surrogate models .................................... 49
  5.2 Considering the 3 dimensional heat conduction inverse problem ........ 50
  5.3 Discussing inverse problems resolution via surrogate models ...... 50

6 Conclusions .......................................................... 52

Bibliography .......................................................... 53
# List of Figures

2.1 Gauss-Newton methods and measurement errors ($tol = 10^{-4}$) ....... 10  
2.2 Flowchart: Regularized Gauss-Newton based methods ............... 12  
2.3 $u$ function measurements with an average error of 2% ............... 15  
2.4 IRGN method ($tol_S = 10^{-5}, tol_U = 10^{-5}$) .................. 15  
2.5 TVRGN method ($tol_S = 4 \times 10^{-5}, tol_U = 10^{-3}, tol_S = 10^{-2}$) ........... 18  
2.6 Illustration of three iterations of the DIRECT algorithm ............ 20  
2.7 DIRECT $L^2$ method ($tol_f = 10^{-8}, tol_x = 10^{-4}, tol_S = 4 \times 10^{-5}, tol_U = 10^{-5}$) .......................................................... 22  
2.8 DIRECT $TV$ method ($tol_f = 10^{-8}, tol_x = 10^{-4}, tol_S = 4 \times 10^{-5}, tol_U = 10^{-3}$) .......................................................... 22  
2.9 Boxplots: DIRECT methods .............................................. 25  
2.10 Boxplots: IRGN and TVRGN methods ................................... 25  
3.1 Example of a Kriging surrogate [1] .................................... 31  
3.2 Example of LHS sampling .............................................. 33  
3.3 DOE Space filling: LHS .............................................. 33  
3.4 Sequential Enrichment flowchart ..................................... 34  
3.5 2D test functions .................................................. 35  
3.6 Expanding an initial DOE: 2D Rosenbrock Function ............... 36  
3.7 Expanding an initial DOE: Camel Back Function ................. 36  
3.8 Schematic representation of $EI$ for a 1D example case [1] ....... 38  
4.1 Surrogate based resolution results ................................... 44  
4.2 Sequential use of EGO algorithm .................................... 46  
4.3 Sequential use of EGO on the $L^2$ regularized problem .......... 47  
4.4 Sequential use of EGO on the $TV$ regularized problem .......... 48  
1 $u$ function measurements with an average error of 10% ............. 57  
2 IRGN method ...................................................... 57  
3 TVRGN method ...................................................... 57  
4 DIRECT $L^2$ method .................................................. 58  
5 DIRECT $TV$ method .................................................. 58  
6 $u$ function measurements with an average error of 2% ............. 59  
7 IRGN method ...................................................... 59  
8 TVRGN method ...................................................... 59  
9 DIRECT $L^2$ method .................................................. 60  
10 DIRECT $TV$ method .................................................. 60
List of Tables

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>End Conditions used in different algorithms</td>
</tr>
<tr>
<td>2.2</td>
<td>Algorithms performance</td>
</tr>
<tr>
<td>2.3</td>
<td>Algorithms performance for increased measurement errors</td>
</tr>
<tr>
<td>2.4</td>
<td>Algorithms performance for a higher dimension</td>
</tr>
<tr>
<td>3.1</td>
<td>RMSE, $err_{\text{max}}$ and $err_r$ for each test function</td>
</tr>
<tr>
<td>3.2</td>
<td>EGO algorithm results</td>
</tr>
</tbody>
</table>
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>LHS</td>
<td>Latin Hypercube Sampling</td>
</tr>
<tr>
<td>SRS</td>
<td>Simple Random Sampling</td>
</tr>
<tr>
<td>DOE</td>
<td>Design Of Experiment</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
<tr>
<td>GN</td>
<td>Gauss Newton</td>
</tr>
<tr>
<td>IRGN</td>
<td>Iteratively Regularized GaussNewton</td>
</tr>
<tr>
<td>TV</td>
<td>Total Variation</td>
</tr>
<tr>
<td>DE</td>
<td>Differential Evolution</td>
</tr>
<tr>
<td>DIRECT</td>
<td>DIViding RECTangles</td>
</tr>
<tr>
<td>EGO</td>
<td>Efficient Global Optimization</td>
</tr>
<tr>
<td>EI</td>
<td>Expected Improvement</td>
</tr>
</tbody>
</table>
1. Introduction

This Chapter gives the general context of the present master thesis in order to better understand its goals and future use. A state of art on the related research work is then conducted. Finally, the contribution of the current study and a document outline are provided.

1.1 General context and goals

This master thesis project has been fulfilled at the French Aerospace Lab ONERA, Toulouse, France. ONERA, created in 1946, is now the leading French player in aeronautics, space and defence research with 25% of the national research effort. One of its most important missions is providing industry with high level expertise through various collaborations.

In this context, my supervisor Dr. Patricia Klotz, a research engineer at the department of Modeling and Information Processing DTIM, formerly collaborated with a private European research company specialized in thermics 'Epsilon Engineering' by supervising PhD works. As an example, Dr. Jonathan Guerra [1] worked on the resolution of multi-objective problems under uncertainties applied to transient thermal engineering phenomena.

The collaboration between ONERA and Epsilon engineering continues by elaborating a future PhD subject. Thus, this master thesis aim to investigate inverse problems resolution using surrogate models which might find a large variety of applications in aerospace thermal engineering problems. As a starting point, the heat conduction problem is then the one considered in this report.

The final goal of this master thesis is to use surrogate models in solving an inverse problem. First, the studied inverse problem is identified and the used assumptions stated. Second, surrogate models are studied and their construction is implemented and controlled. An investigation on different methods used in solving inverse problems is carried out. Chosen algorithms are implemented and their results compared. Finally, an approach is suggested on how surrogate models can be used in solving inverse problems in order to speed up the optimization process.

My supervisor Dr. Patricia Klotz who formerly worked on surrogate modeling [1, 2] along with my co-supervisor Dr. Vincent Mouysset who worked on inverse problems and identification problems [3] advised me and guided me throughout this project.
1.2 Related research work

An inverse problem is such that starting from a set of observed data, missing parameters are determined. By definition an inverse problem is an ill-posed problem highly sensitive to measurement errors on observed data. Consequently, the solution to an inverse problem, if it exists, is not guaranteed to be either stable or unique [4]. Regularization techniques are widely used to treat ill posed problem. In this study, we focus among others on the Tikhonov regularization [5].

An inverse problem is written as an optimization problem. Various optimization algorithms have been developed to reach a reliable solution. Among those algorithms, we can distinguish gradient-based methods and derivative-free methods.

In [6], the gradient based Levenberg-Marquardt method is used to solve a heat and mass transfer in a capillary porous media inverse problem. The same method is also used in [7] to solve an inverse radiation-conduction problem. L.H. Liu in [8] uses the conjugate gradient method and a two dimensional network searching method in solving an inverse radiation problem.

As derivative-free methods, a grid based continuous Ant Colony Optimization (ACO) algorithm solves a coupled radiation and conduction heat transfer inverse problem in [9]. In [9], the Basic Ant Colony Optimization (BACO) algorithm is also improved into the Homogenous Ant Colony Optimization (HACO) algorithm in order to avoid convergence to local optima. In [10], Biao Zhang applies the Quantum Particle Swarm Optimization (QPSO) algorithm to solve a radiative heat transfer inverse problem. An Improved Quantum Particle Swarm Optimization (IQPSO) [10] is developed and proven to be more effective and robust even with noisy data.

If the derivative-free algorithms have the advantage of requiring the knowledge of the objective function only, their main downside is a much higher computational cost [11]. The objective function is estimated many times (thousands up to millions of forward simulations [12]) which can be very expensive especially as the problem dimension gets larger.

According to the literature, researchers managed to speed up the convergence process of different derivative-free optimization algorithms. From the literature we can mention M. Frangos. In [12], Frangos investigates different methods used to simplify and accelerate inverse problems resolution namely reducing the dimension of the input space, using an efficient sampling and finally including reduced models in the inverse problems resolution.

We also want to mention physics-based surrogates. This option consists in using simplified physical laws or numerical discretization which permits to build a computationally
faster reduced model preserving the physical meaning. However, the former simplification methods are not necessary sufficient and can compromise the physics and behaviour of a given system.

Another way to build a reduced model is to construct a surrogate knowing a sample of costly evaluations. These surrogates are independent from the problem physics but are built by statistical learning techniques. The main advantages of such surrogates are their reliability and adaptability to the studied problem especially as they give an analytical fast computed expression of the approximated function.

In the present report, we are interested in using the latter surrogate models to evaluate the direct problem. Surrogate models have been used in different areas of inverse problems. In [13, 14], surrogate models and more specifically Kriging models are used to solve inverse problems in the field of electromagnetics. N.Lioni [15] uses Bayesian surrogate for the resolution of an identification problem in thermal engineering (Wearable Computers). In [16], Kriging models are used in the estimation of the distribution of a non observed random variable. [16] discusses the relevance of the Design Of Experiments (DOE). S.Vakili and M.S.Gadala [11] use surrogate models to improve their Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) algorithm in solving the inverse heat conduction problem.

1.3 Outline

The heat conduction problem is considered in this report. In order to simplify the study, only the one spatial dimensional (1D) case is discussed. Moreover, variations over time are not covered. The inverse problem description is done in Chapter 2.

The DIRECT algorithm [17, 18] and the Gauss-Newton methods (GN, IRGN ...) [3, 19, 20] are here the basis optimization algorithms in solving the inverse problem. Both methods are described and their results discussed in Chapters 2.2.2 and 2.2.3. Explanations and different implementation steps are also given. The methods are verified and validated.

Different surrogate model types can be used in solving inverse problems for example: Polynomial models [21], artificial neural networks (ANN) [21] and Kriging models [2, 22]. We use the Kriging models detailed in Chapter 3.

In Chapter 4, we investigate how Kriging models can be used to solve the inverse problem considered in this report. Moreover, the approach is outlined.

Chapter 5 is dedicated to discussion and presents possible improvements.

A summary and a conclusion are given in Chapter 6.
All the source codes are written in Python version 2.7.
All the simulations are done on a PC with an Intel(R) Core(TM) i5-5200U 2.20 GHz CPU, with 8 GB of Ram and running Microsoft Windows 10.
2. Inverse problem resolution

In this chapter, the solution to the full (non-reduced) model is considered along with the studied test case stated in Section 2.1. In Section 2.1, the need of regularization techniques is demonstrated through the test case results. Section 2.2 is dedicated to solving the regularized problems using both gradient-based optimization methods and derivative-free optimization methods. Finally, a discussion of the test case results is provided in Section 2.3.

2.1 Problem Statement

After giving the general formulation of an inverse problem in Section 2.1.1, the general 3 dimensional forward heat conduction problem is described in Section 2.1.2. The considered test case: the 1 dimensional heat conduction problem is finally given in Section 2.1.3.

2.1.1 Writing an inverse problem

An inverse problem is the process of estimating unknown parameters not accessible by measures (such as causal factors) based on a set of observed data. Let’s consider a function $u$:

$$u : \mathbb{R}^d \rightarrow \mathbb{R}$$

$$x \rightarrow u(x)$$

The function $u$ is dependant of unknown parameters for example an initial state function or cause function $U_0 \in \mathbb{R}^N$. With $U_0 = (u_0^1, u_0^2, \ldots, u_0^N)$

We write the unknown parameters $u_j^0$, $j \in [1..N]$, $N$ being the number of the unknown parameters.

Consider now an approximation $u_k$ of the function $u$ that could be among others an analytical or a numerical approximation...

$$u_k : \mathbb{R}^d \times \mathbb{R}^N \rightarrow \mathbb{R}$$

$$(x, U_0) \rightarrow u_k(x, U_0)$$

Assume we dispose of a set of observed data on $u$, i.e., a set of $M$ $u$-measurements $(u(x_1), u(x_2), \ldots, u(x_M)) = U$. Therefore, the inverse problem is seen as minimizing the distance between observed data $(u_i = u(x_i), i \in [1..M])$ and numerically estimated...
values \((u_k(x_i), i \in [1..M])\). The resulting optimization problem is written:

\[
\min_{U_0 \in \mathbb{R}^N} \sum_{i=1}^{M} \|u(x_i) - u_k(x_i, U_0)\|^2
\]

Which can also be written in a matrix form as:

\[
\min_{U_0 \in \mathbb{R}^N} \|U - U_k(U_0)\|^2 = S(U_0)
\]

Where \(U_0, U_k(U_0)\) and \(U\) are vectors whose elements are respectively \(u^j_0, j \in [1..N]\), \((u_k(x_i))\) and \(u_i = u(x_i), i \in [1..M]\).

By default, an inverse problem is an ill posed problem. The uniqueness and existence of a solution are not guaranteed \([4]\). Moreover, the solution might not be continuously dependant on the data.

After defining the studied test case in section 2.1.3, methods for inverse problems resolution are presented and detailed. Thereafter, the results are compared and discussed in sections 2.2.2 and 2.2.3.

### 2.1.2 The heat conduction forward problem

In the Cartesian coordinate system, the function \(u(x,y,z)\) refers to the temperature in \(^\circ\text{C}\). The heat conduction problem is then described by the 3 dimensional equation 2.1.

\[
\frac{\partial}{\partial x}(k_x \frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(k_y \frac{\partial u}{\partial y}) + \frac{\partial}{\partial z}(k_z \frac{\partial u}{\partial z}) + q = c_p \rho \frac{\partial u}{\partial t} \tag{2.1}
\]

Where \(q\) is the heat generation per unit volume \(W/m^3\). \(k_x, k_y\) and \(k_z\) are the conductivities in each direction \((W/m^\circ\text{C})\). \(c_p\) is the specific heat \((J/kg^\circ\text{C})\) and \(\rho\) the density \((kg/m^3)\).

An initial state \(u_0(x,y,z)\) can be considered. In addition, various boundary conditions can be taken into account such as a prescribed temperature, a prescribed heat flux or a radiative heat exchange and/or a convective heat exchange.

A convective heat exchange is a heat transfer between a solid and a moving fluid. The convective heat exchange boundary condition is written as:

\[
-(k_x \frac{\partial u}{\partial x} + k_y \frac{\partial u}{\partial y} + k_z \frac{\partial u}{\partial z}) = h(u_s - u_f) \tag{2.2}
\]

with \(h\) the convective heat transfer coefficient, \(u_s\) the prescribed temperature at the wall and \(u_f\) the ambient temperature or the fluid temperature.

The radiative heat exchange is a heat transfer through electromagnetic radiation. The
Inverse problem resolution

7

radiative heat exchange boundary condition is written as:

\[-(k_x \frac{\partial u}{\partial x} + k_y \frac{\partial u}{\partial y} + k_z \frac{\partial u}{\partial z}) = \epsilon \sigma (u_s^4 - u_f^4)\] (2.3)

Where \(\epsilon\) the emissivity, \(\sigma\) the Stefan Boltzmann constant. \(u_s\) the prescribed temperature at the wall and \(u_f\) the ambient temperature.

2.1.3 Test case

We consider a highly simplified heat conduction problem which is used to test different optimization techniques. A heat conduction problem [5] in 1 dimension is taken here as an example. However, all the methods can be adapted to the former 3 dimensional problem.

The direct heat conduction problem can be written as:

\[
\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \\
u_0(x) = \exp\left(-\frac{(x-x_0)^2}{\sigma^2}\right)
\]

Which \(u = u(x)\) and \(x \in [x_0 - \sigma, x_0 + \sigma]\).

Taking \(\alpha = 1\) and \(\sigma = 1\), the solution to this problem is written in equation 2.4,

\[u(x,t) = \int_{x_0-1}^{x_0+1} K(t,x-y)u_0(y)dy\] (2.4)

Where \(K(t,x)\) is the heat kernel: \(K(t,x) = \frac{1}{(4\pi t)^{\frac{3}{2}}} \exp(-\frac{|x|^2}{4t}), x \in [x_0 - 1, x_0 + 1]\).

At a given time \(t=1\):

\[u(x) = \int_{x_0-1}^{x_0+1} K(x-y)u_0(y)dy\]

If the direct problem is finding \(u(x)\) satisfying the heat equation and an initial state \(u_0(x)\), the inverse problem is finding the initial state \(u_0(x)\) knowing a set of observed data \((u_i = u(x_i), i \in [1..M])\).

Practically, a discretization is applied in order to numerically solve this inverse problem. The function \(u_0(x)\) is approximated by a step function \(u_{0,k}(x)\).

\[u_{0,k}(x) = \sum_{j=1}^{N} u_0^j \chi_j(x)\]

\[u_0^j \in \mathbb{R}, \quad \forall j \in [1..N]\]
Inverse problem resolution

Where \( \chi_{I_j}, j \in [1..N] \) are indicator functions and \([x_0 - 1, x_0 + 1] \subset [a, b] = \cup I_j\). Consequently, \( u_k(x) \) approaches \( u(x) \) and can be written as following:

\[
 u_k(x) = \sum_{j=1}^N u_0^j \int_{I_j} K(x - y)dy
\]

The function \( u(x) \) is then approached by \( u_k(x) \), a linear function in \( u_0^j \forall j \in [1..N] \).

The inverse problem is then written as a linear least square problem:

\[
 \min_{U_0 \in \mathbb{R}^N} S(U_0) = \|U - U_k(U_0)\|^2
\]

### 2.1.4 Gradient-based optimization for non regularized formulation

In order to solve the test case inverse problem in its non-regularized formulation described in 2.1.3, a gradient-based method and more precisely the Gauss-Newton method (GN) is used.

**Method description**

The convex problem to be solved here is simply written as:

\[
 \min_{U_0 \in \mathbb{R}^N} S(U_0)
\]

The Gauss-Newton Method (Algorithm 1) is used to find a local optimizer of a least squares problem. It is an iterative process which starting from \( U_{0(p)} \) at the stage \( p \) of the optimization algorithm determines the next stage variables vector \( U_{0(p+1)} \) that minimizes the objective function.

The present algorithm generates a sequence \( (U_{0(p)}) \) which converges to the optimal vector of variables \( (U_{0optimal}) \).

Let \( U_{0(p+1)} \) be written as:

\[
 U_{0(p+1)} = U_{0(p)} + dU_{0p}
\]

Where \( dU_{0p} \) is a descent direction bringing \( U_{0(p)} \) closer to an optimizer \( U_{0optimal} \).

Applying the first order necessary optimality condition to \( U_{0(p+1)} = U_{0(p)} + dU_{0p} \), namely setting the first derivative of the objective function in \( U_{0(p+1)} \) to zero \( (S(U_{0(p+1)}) = 0) \), \( dU_{0p} \) is given by the equation 2.5 [3]:

\[
 dU_{0p} = \left( DU_k(U_{0(p)})^T DU_k(U_{0(p)}) \right)^{-1} \left( DU_k(U_{0(p)})^T (U_k(U_{0(p)}) - U) \right)
\]

(2.5)
Inverse problem resolution

Where $DU_k(U_0(p))$ is the Jacobian matrix of $U_k$ evaluated in $U_0(p)$.

Equation 2.5 is obtained under the assumption that the second derivative of $U_k$ is negligible.

The termination condition is taken here as:

$$\|DU_k(U_0(p))^T(U - U_k(U_0(p)))\| \leq tol$$

Where the tolerance $tol$ is chosen by the user.

An optimal step length $\alpha^*$ can also be computed to improve the optimization process:

$$\text{minimize } S(U_0 + \alpha dU_0)$$

$$\text{s.t } \alpha \in \mathbb{R}$$

The discretization parameter $N$ is in this case a regularization parameter. As the dimension $N$ increases, the problem gets ill conditioned and leads to a loss of information. The solution to the problem $U_0^{optimal}$ is highly sensitive to the variations in $(u_i, i \in [1..M])$ values measurements. In fact, the observed data can be issued from precise calculations but commonly come from experimental results or modeling results. $(u_i, i \in [1..M])$ are then affected by errors.

**Algorithm 1** Gauss-Newton (GN)

**input**: Initial solution Guess $U_{0,init}$

Functions $U_k$ and $DU_k$

**output**: $U_0^{optimal}$

$p \leftarrow 0$

$p \leftarrow U_{0,init}$

$cond \leftarrow 1$

**while** $cond > tol$ **do**

Evaluate $U_k(U_0(p))$ and $DU_k(U_0(p))$

$cond \leftarrow \|DU_k(U_0(p))^T(U - U_k(U_0(p)))\|$

Compute $dU_0(p)$ using equation 2.5

Determine the optimal step length $\alpha^*$

$U_0(p+1) \leftarrow U_0(p) + \alpha^*dU_0(p)$

$p \leftarrow p + 1$

**end**

Return $U_0(p+1) = U_0^{optimal}$

$p$ the number of iterations
Test case results

The Jacobian matrix is written as equation 2.6:

\[
DU_k(U_{0(p)}) = \begin{bmatrix}
\int_{I_1} K(x_1 - y)dy & \ldots & \int_{I_N} K(x_1 - y)dy \\
\vdots & \ddots & \vdots \\
\int_{I_1} K(x_M - y)dy & \ldots & \int_{I_N} K(x_M - y)dy
\end{bmatrix}
\] (2.6)

The results given by the Gauss-Newton method is given in 2.1.

![Figure 2.1: Gauss-Newton methods and measurement errors (tol = 10^{-4})](image)

(a) Result from exact random measures  
(b) Result from non-exact measures

We show here the need of regularization when solving an inverse problem. The non regularised algorithm \(GN\) is used to solve the inverse problem related to exact data (figure 2.1a) and then related to data with ‘measurement’ errors of an order of magnitude of ±2% (figure 2.1b).

In figures 2.1, we can see that measurement errors induce a non valid inverse problem solution when the gradient based algorithm \((GN)\) is used. Regularization techniques are presented in the following section 2.2.1.

### 2.2 Regularized problems

Section 2.2.1 sets the used regularization techniques. The regularized problems are then solved using gradient-based algorithms namely the Iteratively Regularized Gauss-Newton method 2.2.2 and the Total Variation Regularized Gauss-Newton method 2.2.2. A derivative-free optimization algorithms DIRECT is also presented 2.2.3 as an alternative when the derivative of a given function is unknown.
2.2.1 Inverse problem formulation with regularization

Regularizing this ill posed problem consists in building a converging sequence of solutions to well-posed problems [2]. Generally, a regularization term is added to the initial objective function. However, searching for an optimizer in a finite dimension space instead of an infinite dimension space is already a problem regularization. [5].

The Tikhonov [5] regularization term is one of the most commonly used ones. It is written in 2.7:

$$R_{L^2} = \|U_0 - U_{0ref}\|^2$$  \hspace{1cm} (2.7)

Where $U_{0ref}$ is a reference point. The corresponding objective function is hence the following:

$$S_{L^2}(U_0) = \|U - U_k(U_0)\|^2 + \lambda\|U_0 - U_{0ref}\|^2$$  \hspace{1cm} (2.8)

Another regularization term could be 2.9:

$$R_{TV}(U_0) = \|\nabla U_0\|_{L^1}$$  \hspace{1cm} (2.9)

This regularisation term (equation 2.9) is based on the total variation of the vector $U_0$ elements. It is often referred as the Bounded Variation semi-norm [3, 19, 20, 23].

The new function to be minimized is given by equation 2.10:

$$S_{TV}(U_0) = \|U - U_k(U_0)\|^2 + \lambda\|\nabla U_0\|_{L^1}$$  \hspace{1cm} (2.10)

In equations 2.8 and 2.10, $\lambda$ is a regularization parameter.

The next sessions are dedicated to the minimization of the former functions using both gradient-based methods and derivative-free based methods.

2.2.2 Gradient-based optimization for regularized formulations

Applying the Gauss-Newton method to the initial non-regularized problem is proven non-effective in section 2.1.4 [3]. In order to determine the most suited regularization technique to the studied test case, two algorithms are considered. The Gauss-Newton method is first applied to the Tikhonov regularized problem 2.7 which is known as the Iteratively Regularized Gauss-Newton method (IRGN) [3]. Second, the method is applied to the TV-regularized problem 2.9 [19, 23] [20].

Whatever the used optimization algorithm is, it is applied sequentially. The optimization algorithms are respectively called various times in order to get to the optimal vector.
The general stopping criteria being:

\[
\|S(U_{0(p+1)}; \lambda_{p+1}) - S(U_{0(p)}; \lambda_p)\| \leq tol_S \\
\|U_{0(p+1)} - U_{0(p)}\| \leq tol_U
\]

**The Iteratively Regularized Gauss-Newton Method IRGN**

As it is done in section 2.1.4, we start by describing the optimization method and then the test case are given.

**Method description**

In the present report, the zero vector is taken as a reference point \(U_{0\text{ref}}\).

The problem to be solved is:

\[
\min_{U_0 \in \mathbb{R}^N} S_{L^2}(U_0) = \|U - U_k(U_0)\|^2 + \lambda\|U_0\|^2
\]
Where $\lambda$ is a regularisation parameter that must be positive ($\lambda \geq 0$).

Starting from $U_{0(p)}$ at iteration $p$, the vector $U_{0(p+1)}$ bringing the objective function $S_{L^2}$ closer to its minimum value is written as:

$$U_{0(p+1)} = U_{0(p)} + dU_{0p}$$

where [3]:

$$dU_{0p} = (DU_k(U_{0(p)})^T DU_k(U_{0(p)}) + \lambda I)^{-1}(DU_k(U_{0(p)})^T (U_k(U_{0(p)} - U) + \lambda U_{0(p)}))$$ (2.11)

To improve the optimization process, an optimal step length $\alpha^*$ is determined.

$$\min_{\alpha \in \mathbb{R}} S_{L^2}(U_{0p} + \alpha dU_{0p})$$

The iterative process is similar to the one described in section 2.1.4.

When using the present regularization, taking the first derivative of the objective function as part of the termination condition can be very time consuming if not unreachable [3, 23]. The termination condition is then taken differently. The iterations stop when the objective function variations and/or the vector $U_0$ variations are lower than given limits $tol_S$ and $tol_U$ respectively.

$$|S_{L^2}(U_{0(p+1)}) - S_{L^2}(U_{0(p)})| \leq tol_S$$
$$\|U_{0(p+1)} - U_{0(p)}\| \leq tol_U$$

Assuming that the order of magnitude of errors made on the observed data is $0.01$, one cannot expect the final value of the objective function to be more accurate than $0.01^2 = 10^{-4}$. $tol_S$ is then chosen accordingly.

Different steps of the IRGN algorithm can be summed up in Algorithm 2.
**Algorithm 2** Iteratively Regularized Gauss-Newton (IRGN)

**input**: Initial solution Guess $U_{0\text{init}}$

Functions $U_k$ and $DU_k$

**output**: $U_0^{\text{optimal}}$

1. $p \leftarrow 0$
2. $U_{0(p)} \leftarrow U_{0\text{init}}$
3. $dS \leftarrow 1$
4. $dU \leftarrow 1$

**while** $dS > \text{tol}_S$ and $du > \text{tol}_U$ **do**

- Evaluate $U_k(U_{0(p)})$ and $DU_k(U_{0(p)})$
- Compute $dU_{0(p)}$ using equation 2.11
- Determine the maximum step length $\alpha^*$

| $U_{0(p+1)} \leftarrow U_{0(p)} + \alpha^* dU_{0(p)}$ |
| $du \leftarrow |S_{L^2}(U_{0(p+1)}) - S_{L^2}(U_{0(p)})|$ |
| $dS \leftarrow \|U_{0(p+1)} - U_{0(p)}\|$ |
| $p \leftarrow p + 1$ |

**end**

Return $U_{0(p+1)} = U_0^{\text{optimal}}$

$p$ the number of iterations

For a fixed $\lambda$, the Algorithm 2 is applied to determine $U_0^* = \arg\min_{U_0 \in \mathbb{R}^N} S(U_0; \lambda)$. The regularization parameter $\lambda$ is then decreased ($\lambda \leftarrow \frac{\lambda}{2}$) and Algorithm 2 reapplied (see flowchart 2.2).

**Test case results**

Starting from the measurements in figure 2.3, the iterative use 2.2 of the IRGN algorithm is applied.

Note that the used Jacobian matrix $DU_k$ is defined in equation 2.6.

Using regularized algorithm (IRGN) is shown to be effective in figures 2.4.
Inverse problem resolution

Figure 2.3: $u$ function measurements with an average error of 2%

Figure 2.4: IRGN method ($tol_S = 10^{-5}, tol_U = 10^{-5}$)

It can be seen from figure 2.4b that using the Iteratively Regularized Gauss-Newton method implies around 20 iterations in the iterative process which might slow down the global optimization process. The TV regularization technique is then used in section 2.2.2 and its results presented.

Total Variation Regularized Gauss-Newton Method

The same methodology is applied in this section. After describing the Total Variation Regularized Gauss-Newton method, the studied test case results are discussed.

Method description

In this particular case, the goal is to reduce the variation of $dU_0$ instead of reducing the variation of $U_0$.

The TV regularization is applied to a linearized problem. The objective function is then

$$
\tilde{S}_{TV}(dU_0) = \|U - U_k(U_{0(p)}) - DU_k(U_{0(p)})dU_0\|^2 + \lambda\|\nabla dU_0\|_{L^1}
$$

(2.12)
Inverse problem resolution

$p$ referring to the iteration number.

In order to be able to apply the previous method (Gauss-Newton Method 2.1.4) on the new objective function $S_{TV}(dU_0)$, the regularization term needs to be differentiable which is obviously not the case.

Practically, the regularization term is replaced by a new regularization term.

$$R_{TV}(U_0) = \|\sqrt{\nabla|U_0|^2 + \beta}\|_{L^1} \quad (2.13)$$

Where $\beta$ is a positive real number. The value of $\beta$ must be kept under $\|dU_0\|$.

A similar recursive process to the ones used for the non-regularized Gauss-Newton Method section 2.1.4 and the Iterative Regularized Gauss-Newton Method section 2.2.2 is used [3, 23]. It means that starting from the vector $U_0(p)$ at iteration $p$, the vector $U_0(p+1)$ bringing the objective function closer to its minimum value is calculated as follows:

$$U_{0(p+1)} = U_{0(p)} + dU_0$$

With:

$$[DU_k(U_{0(p)})^T DU_k(U_{0(p)}) + \lambda_p R^T DG(\delta_s)](\delta_{s-1} - \delta_s) = DU_k(U_{0(p)})^T DU_k(U_{0(p)})(\delta_s) - DU_k(U_{0(p)})^T (U - U_k(U_{0(p)})) + \lambda_p R^T G(\delta_s)$$

Where:

For $x \in \mathbb{R}^N$, $G(x)$ is a $\mathbb{R}^{N+1}$ vector defined as:

$$G(x) = \begin{bmatrix} x_1 \\
\sqrt{\|x_1\|^2 + \beta} \\
\sqrt{\|x_2-x_1\|^2 + \beta} \\
\vdots \\
\sqrt{\|x_N-x_{N-1}\|^2 + \beta} \\
\sqrt{\|x_N\|^2 + \beta} \end{bmatrix} \quad (2.14)$$

$DG(x)$ is a $\mathbb{R}^{N \times N+1}$ matrix defined by:

$$DG(x) = \begin{bmatrix} \beta/((\|x_1\|^2 + \beta)^{\frac{3}{2}}) & -\beta/((\|x_2-x_1\|^2 + \beta)^{\frac{3}{2}}) & \beta/((\|x_3-x_2\|^2 + \beta)^{\frac{3}{2}}) & \cdots & -\beta/((\|x_N-x_{N-1}\|^2 + \beta)^{\frac{3}{2}}) \\
-\beta/((\|x_2-x_1\|^2 + \beta)^{\frac{3}{2}}) & \beta/((\|x_3-x_2\|^2 + \beta)^{\frac{3}{2}}) & \cdots & -\beta/((\|x_N-x_{N-2}\|^2 + \beta)^{\frac{3}{2}}) \\
\beta/((\|x_3-x_2\|^2 + \beta)^{\frac{3}{2}}) & -\beta/((\|x_4-x_3\|^2 + \beta)^{\frac{3}{2}}) & \cdots & -\beta/((\|x_N-x_{N-2}\|^2 + \beta)^{\frac{3}{2}}) \\
\vdots & \vdots & \ddots & \ddots & \ddots \\
-\beta/((\|x_N-x_{N-1}\|^2 + \beta)^{\frac{3}{2}}) & -\beta/((\|x_N-x_{N-2}\|^2 + \beta)^{\frac{3}{2}}) & \cdots & \beta/((\|x_N\|^2 + \beta)^{\frac{3}{2}}) \end{bmatrix} \quad (2.15)$$
And \( \mathbf{R} \) is the matrix corresponding to the application \( R \) [23]:

\[
R : \mathbb{R}^N \rightarrow \mathbb{R}^{N+1} \\
\delta \rightarrow \delta_+ - \delta_-
\]

\[
\mathbf{R} = \begin{bmatrix}
1 & 1 & \cdots \\
-1 & 1 & \cdots \\
& \cdots & \cdots \\
& -1 & 1
\end{bmatrix}
\tag{2.16}
\]

The sequence \((\delta_s)_{s=1,2,...}\) converges toward \(dU_{0(p)}\).

The corresponding stopping criteria can be written as:

\[\|\delta_{s+1} - \delta_s\| \leq tol_\delta\]

\(tol_\delta\) cannot be lower than \(\beta\).

Once the vector \(dU_{0(p)}\) corresponding to the vector \(U_{0(p)}\) is determined, an optimal step length \(\alpha_p\) can be evaluated.

The termination conditions on \(U_{0(p)}\) taken in this case are similar to what has been used for the IRGN method 2.2.2. The stopping conditions are rewritten as:

\[\|S_{TV}(U_{0(p+1)}) - S_{TV}(U_{0(p)})\| \leq tol_S\]
\[\|U_{0(p+1)} - U_{0(p)}\| \leq tol_U\]

It should be noticed here as well that \(tol_S\) cannot be chosen regardless of errors rate made on the observed data.

Algorithm 3 illustrates the \(TV\) regularised \(GN\) method.
Algorithm 3 TV Regularized Gauss-Newton (TVRGN)

**input**: Initial solution \( U_{0_{init}} \)
Functions \( U_k \) and \( D U_k \)

**output**: \( U_{0_{optimal}} \)

\[
p \leftarrow 0
\]
\[
U_{0(p)} \leftarrow U_{0_{init}}
\]
\[
dS \leftarrow 1
\]
\[
dU \leftarrow 1
\]

while \( dS > tol_S \) and \( du > tol_U \) do

Evaluate \( U_k(U_{0(p)}) \) and \( DU_k(U_{0(p)}) \)

Compute \( dU_{0(p)} \) as the limit of the sequence \( (\delta_s)_{s=1,2,...} \)

Determine the maximum step length \( \alpha^* \)

\[
U_{0(p+1)} \leftarrow U_{0(p)} + \alpha^* dU_{0(p)}
\]
\[
du \leftarrow |S_{TV}(U_{0(p+1)}) - S_{TV}(U_{0(p)})|
\]
\[
dS \leftarrow \|U_{0(p+1)} - U_{0(p)}\| \quad p \leftarrow p + 1
\]

end

Return \( U_{0(p+1)} = U_{0_{optimal}} \)

\( p \) the number of iterations

For a fixed \( \lambda \), the Algorithm 3 is applied to determine \( U_0^* = \arg\min_{U_0 \in \mathbb{R}^N} S(U_0; \lambda) \). The regularization parameter \( \lambda \) is then decreased (\( \lambda \leftarrow \frac{\lambda}{2} \)) and Algorithm 3 reapplied (see flowchart 2.2).

**Test case results**

Based on the same observed data figure 2.3 The results of an iterative use of the TVRGN algorithm are presented in figures 2.5.

![Figure 2.5: TVRGN method (\( tol_S = 4 \times 10^{-5}, tol_U = 10^{-3}, tol_\delta = 10^{-2} \))]
The TVRGN is also shown to be effective especially as it reaches the desired result (the function $u_{0k}$ approaches $u_0$) is around 3 iterations instead of the 20 iterations needed for the IRGN based iterative process convergence.

Regularizing the ill posed inverse problem 2.1.4 is an effective technique in solving an inverse problem. However, using a gradient based algorithm either the Iteratively Regularized Gauss-Newton (IRGN) method 2.2.2 or the Total Variation Regularized Gauss-Newton (TVRGN) method 2.2.2 needs to dispose of a analytical expression of the Jacobian matrix $DU_k$ (in the test case equation 2.6) which is not necessary accessible.

In the following section, a derivative-free algorithm DIRECT is used instead of the gradient based algorithms on the regularized problems presented section 2.2.1.

### 2.2.3 Derivative-free method for regularized problems

When the function derivative is unknown, the gradient-based method can not be used to find a function minimum (or maximum) value. In this section, a derivative-free method is presented and applied to the test case problem.

**Description of the DIRECT algorithm**

The DIRECT optimization algorithm was introduced by D.R. Janes in [17]. It solves difficult global optimization problems namely non linear, non convex objective functions which gradient is unknown.

The algorithm only requires the objective function to be real-valued and the search domain to be bounded. This method is based on sampling the domain, evaluating the objective function on sampled elements and depending on the former results deciding where to search next. More specifically, the DIRECT algorithm divide the entire definition domain into a set of hypercubes, evaluate the objective function at the center of each hypercube then redivide the hypercubes where the objective function was small.

The DIRECT algorithm converges towards the global optimal value of the objective function. Its main disadvantage is requiring a great number of function evaluations by exploring thinly the domain.

The DIRECT algorithms steps can be summed up as following:

- Transforming the domain into a single hypercube and evaluate the function at its center $c_1$. 
• Dividing the first hypercube by evaluation the function at the points $c_i \pm \delta e_i$
  $i = 1...N$ where N is the departure domain dimension, $e_i$ the $i^{th}$ unit vector, and
  $\delta$ one-third the side-length of the hypercube.

• Evaluating the objective function at each new center and dividing the hypercube
  corresponding the the minimal value.

• The process is repeated until the end criteria are met.

Details on dividing potentially optimal hypercubes can be found in [17, 24]. An illustration of three iterations of the algorithm is though given in figure 2.6.
At each iteration, the function is evaluated at the center of each rectangle (black dots). Optimal rectangles (light blue) are such that the function is optimal at the center of the rectangles in question. Optimal rectangles are identified at each level of division (see iteration 3). The optimal rectangle is then divided into smaller rectangles. Searching for the optimal value of a function using the DIRECT algorithm then covers globally the search space while investigating locally potential optimal regions of the search space.

![Figure 2.6: Illustration of three iterations of the DIRECT algorithm](image-url)
The termination criteria could be criteria on the relative objective function variation, the relative optimizer variation, a number of function evaluations and a maximum computational time.

\[
\frac{f(x_{p+1})-f(x_p)}{f(x_p)} < tol_f \\
\frac{x_{p+1}-x_p}{x_p} < tol_x
\]

The DIRECT algorithm convergence is studied in [25].

The NLOPT library [26] available on python has been used during this project [27] [18].

**DIRECT algorithm used for regularised formulations**

The regularised inverse problems presented in 2.2.1 are solved using the DIRECT algorithm in an iterative process (as it is done for the gradient based algorithms 2.2) as following:

**Algorithm 4** Sequential use of the DIRECT algorithm

**input**: The regularized objective function \( S \) that could be either \( S_{L^2} \) or \( S_{TV} \)

**output**: Optimizer \( U_0 \)

Initialize upper and lower bounds: \( Low_{bounds}, Up_{bounds} \)

Initialize the regularization parameter: \( \lambda \leftarrow 1 \)

**while** termination conditions **do**

| Perform the DIRECT optimisation to get \( U_0^{optimal} \) |
| \( Low_{bounds} \leftarrow 0.5 \times U_0^{optimal}, Up_{bounds} \leftarrow 1.5 \times U_0^{optimal} \) |
| Reset the regularization parameter: \( \lambda \leftarrow \frac{\lambda}{2} \) |

**end**

Return \( U_0^{optimal} \)

The termination conditions at an iteration \( p + 1 \) for the sequential use of the DIRECT algorithm are:

\[
\|S_X(U_{0(p+1)}) - S_X(U_{0(p)})\| \leq tol_S \\
\|U_{0(p+1)} - U_{0(p)}\| \leq tol_U
\]

Where \( X \) refers to the regularization technique (\( L^2 \) or \( TV \)).

**Test case results**

Based on the same measurement as in figure 2.3, figures 2.7 and 2.8 are the results obtained using the DIRECT algorithm in an iterative process on both the \( L^2 \) regularized
Inverse problem resolution

problem and the TV regularized problem 2.2.1. The final results seems to be similar to the results obtained using gradient based algorithms.

\[ U_0 \text{ through iterations} \quad \text{(b) Final } U_0 \quad \text{(c) Objective function} \]

**Figure 2.7:** DIRECT $L^2$ method ($\text{tol}_f = 10^{-8}$, $\text{tol}_x = 10^{-4}$, $\text{tol}_S = 4 \times 10^{-5}$, $\text{tol}_U = 10^{-5}$)

\[ U_0 \text{ through iterations} \quad \text{(b) Final } U_0 \quad \text{(c) Objective function} \]

**Figure 2.8:** DIRECT TV method ($\text{tol}_f = 10^{-8}$, $\text{tol}_x = 10^{-4}$, $\text{tol}_S = 4 \times 10^{-5}$, $\text{tol}_U = 10^{-3}$)

### 2.3 Results discussion

In this section, different resolution methods are compared in terms of precision, robustness and calculation speed. Not only the regularization techniques: the $L^2$ regularization and the TV regularization are compared but also and most importantly the gradient based algorithms ($ITRGN$, $TVRGN$) and the derivative-free algorithm (DIRECT) are compared. First the stopping criteria used in different algorithms are reminded in table 2.1, $\epsilon$ being the order of magnitude of measurement errors. The given values are used when applying the algorithms in different examples considered in this section 2.3.
Table 2.1: End Conditions used in different algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>End conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>GN</td>
<td>(tol = 10^{-4})</td>
</tr>
<tr>
<td>IRGN</td>
<td>(tol_S = 10^{-5}) (tol_U = 10^{-5})</td>
</tr>
<tr>
<td>TVRGN</td>
<td>(tol_S = 2\epsilon \times 10^{-3}) (tol_U = 10^{-3}) (tol_S = 10^{-2})</td>
</tr>
<tr>
<td>DIRECT</td>
<td>(tol_f = 10^{-8}) (tol_x = 10^{-4})</td>
</tr>
<tr>
<td>DIRECT (L^2)</td>
<td>(tol_S = 2\epsilon \times 10^{-3}) (tol_U = 10^{-5})</td>
</tr>
<tr>
<td>DIRECT (TV)</td>
<td>(tol_S = 2\epsilon \times 10^{-3}) (tol_U = 10^{-3})</td>
</tr>
</tbody>
</table>

Table 2.2: Algorithms performance

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>(S(U_{\text{final}}))</th>
<th>(dU_{0,\text{final}})</th>
<th>(dS_{\text{final}})</th>
<th>(\lambda_{\text{final}})</th>
<th>(nb_{\text{it}})</th>
<th>(T(s))</th>
<th>(eval_{\text{f}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRGN</td>
<td>1.4</td>
<td>0.22</td>
<td>(3.5 \times 10^{-4})</td>
<td>(9.54 \times 10^{-7})</td>
<td>21</td>
<td>24.3</td>
<td>-</td>
</tr>
<tr>
<td>TVRGN</td>
<td>13.14</td>
<td>(1.3 \times 10^{-2})</td>
<td>(4.55 \times 10^{-5})</td>
<td>0.5</td>
<td>2</td>
<td>1.72</td>
<td>-</td>
</tr>
<tr>
<td>DIRECT (L^2)</td>
<td>(12.3 \times 10^{-5})</td>
<td>(1.6 \times 10^{-7})</td>
<td>(13.3 \times 10^{-5})</td>
<td>(1.9 \times 10^{-6})</td>
<td>20</td>
<td>3518.1</td>
<td>25172</td>
</tr>
<tr>
<td>DIRECT (TV)</td>
<td>6.6</td>
<td>(6.7 \times 10^{-4})</td>
<td>13.2</td>
<td>0.25</td>
<td>3</td>
<td>378.3</td>
<td>1265</td>
</tr>
</tbody>
</table>

Regularized algorithms (\(IRGN, TVRGN, DIRECT \(L^2\)\) and \(DIRECT \(TV\)\)) are shown to be effective in figures 2.4, 2.5, 2.7 and 2.8 respectively. However, one can notice different performances depending on the used algorithm as well as the used regularization technique. In table 2.2, we give number of iterations \(nb_{\text{it}}\), the execution time \(T(s)\) and the number of function evaluation \(eval_{\text{f}}\) when using the DIRECT algorithms. All else beings equal, the number of iterations and the calculation times are larger when using the \(L^2\) regularization. If \(TV\) regularized problems reach the optimize in a very few iterations (2 to 5 iterations), \(L^2\) regularized problems need around 20 iterations. Moreover, the end conditions set for the \(TV\) regularized problems are more robust than the end conditions set for the \(L^2\) regularized problems. The risk is here to perform additional and useless iterations that tend to take the result away from the optimizer as seen in figure 2.4a.
The measurement errors are increased up to ±20% and more in order to test each regularisation technique robustness. The dimension $N$ of the inverse problem is also increased. In table 2.3, example results for ±10% errors are presented and their relative figures are in Appendix 6. Table 2.4 gives example results for a dimension $N = 20$ for a set of 40 measurements (table 2.4).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$S(U_{final})$</th>
<th>$dU_{0_{final}}$</th>
<th>$dS_{final}$</th>
<th>$\lambda_{final}$</th>
<th>nb$_it$</th>
<th>$T(s)$</th>
<th>eval$_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRGN</td>
<td>$1.32 \times 10^{-2}$</td>
<td>0.48</td>
<td>$1.55 \times 10^{-4}$</td>
<td>$3.05 \times 10^{-5}$</td>
<td>16</td>
<td>19.58</td>
<td>-</td>
</tr>
<tr>
<td>TVRGN</td>
<td>6.58</td>
<td>$1.36 \times 10^{-2}$</td>
<td>$1.03 \times 10^{-4}$</td>
<td>3</td>
<td>0.25</td>
<td>4.67</td>
<td>-</td>
</tr>
<tr>
<td>DIRECT $L^2$</td>
<td>$1.32 \times 10^{-2}$</td>
<td>5.68 $\times 10^{-6}$</td>
<td>$1.32 \times 10^{-2}$</td>
<td>$1.90 \times 10^{-6}$</td>
<td>20</td>
<td>2818.97</td>
<td>16224</td>
</tr>
<tr>
<td>DIRECT TV</td>
<td>6.55</td>
<td>$9.01 \times 10^{-4}$</td>
<td>13.09</td>
<td>0.25</td>
<td>3</td>
<td>381.35</td>
<td>1279</td>
</tr>
</tbody>
</table>

Table 2.3: Algorithms performance for increased measurement errors

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$S(U_{final})$</th>
<th>$dU_{0_{final}}$</th>
<th>$dS_{final}$</th>
<th>$\lambda_{final}$</th>
<th>nb$_it$</th>
<th>$T(s)$</th>
<th>eval$_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRGN</td>
<td>$7.4 \times 10^{-4}$</td>
<td>0.13</td>
<td>$2.9 \times 10^{-5}$</td>
<td>$7.63 \times 10^{-6}$</td>
<td>18</td>
<td>30.42</td>
<td>-</td>
</tr>
<tr>
<td>TVRGN</td>
<td>11.53</td>
<td>$7.7 \times 10^{-3}$</td>
<td>$2.44 \times 10^{-6}$</td>
<td>0.25</td>
<td>3</td>
<td>2.85</td>
<td>-</td>
</tr>
<tr>
<td>DIRECT $L^2$</td>
<td>$9 \times 10^{-4}$</td>
<td>5.68 $\times 10^{-6}$</td>
<td>$1.1 \times 10^{-3}$</td>
<td>17</td>
<td>1.5 $\times 10^{-5}$</td>
<td>12461.97</td>
<td>22467</td>
</tr>
<tr>
<td>DIRECT TV</td>
<td>2.9</td>
<td>$1.23 \times 10^{-4}$</td>
<td>5.8</td>
<td>0.0625</td>
<td>5</td>
<td>5296.76</td>
<td>2863</td>
</tr>
</tbody>
</table>

Table 2.4: Algorithms performance for a higher dimension

Overall, the results show a better performance of the Total Variation TV regularisation when considering precision and robustness.

Execution time is also a criteria to compare both regularisation methods. The speed of convergence in terms of number of iterations and computation time is more costly when using $L^2$ regularisation.

Calculations are repeated up to 100 times for different sets of measurements. We present here (figures 2.9 and 2.10) the resulting box-plots that give an idea on how robust each algorithm is. 50 sets of measurements are done. For $M = 20$ measurements, the errors are taken here at an order of magnitude of ±2%. The inverse problem dimension $N$ is equal to 10.

For the TV regularized problem, the variance obtained when using the DIRECT algorithm is larger than the variance obtained using the TVGN algorithm. However, we notice more bias when using the DIRECT algorithm (figures 2.9b and 2.10b).
Gradient based algorithms and derivative-free algorithms are then compared. If robustness and precision are similar for the same inverse problem, computation times are not. As expected the derivative-free algorithm DIRECT is way slower than gradient based algorithm (Gauss-Newton). A large number of function evaluations is also needed to fulfill DIRECT based methods. Given that the considered problem is rather simple, a more complicated case will need far more function evaluations.

Derivative-free algorithms remain a very interesting way to solve inverse problems as only the function evaluations are needed. We can remedy the computational cost of such methods by using surrogate models presented in chapter 3 where we explain how a Kriging model is constructed and how it can be used. In chapter 4, surrogate models are utilized to solve the test case inverse problem. A surrogate model can be called instead of the costly objective function when applying optimization algorithms or optimization algorithms based on surrogate models can be used.
3. **Kriging models construction and kriging based optimization**

In this chapter, the background material and the basics of Kriging models construction are reviewed in section 3.1. A way to build a surrogate with a reduced number of learning samples is presented in section 3.2. Finally, a surrogate based optimization algorithm is introduced in section 3.3.

### 3.1 Building a surrogate: Kriging model

Surrogate models, also called metamodels, are used to approximate functions whose evaluation is expensive. Metamodels are especially used when these functions are called various times in an optimization process for example.

Based on few numerical simulations, or few realizations of experiments, a surrogate is able to ‘quickly’ estimate the related function. A surrogate model approximates a costly function without knowing about its characteristics or particularities other than its values on a set of observed data [21, 22, 28].

Different surrogate models can be built depending on their basis function and the learning techniques they use. As an example, a surrogate model can describe more or less a function non-linearity. We could mention polynomial models, neural networks models or Kriging models [1, 2, 22, 28].

Regardless of the chosen method, building a surrogate model can follow the following stages [11]:

- **Choosing the Design Of Experiments (DOE):** An evenly distributed sampling over the design space is a way to select the experimental design. A Latin Hypercube Sampling LHS is usually used for its space-filling performances. The DOE is split into: learning samples and test samples which use is detailed in the following stages.

- **Evaluating the objective function:** The expensive objective function is evaluated on both learning samples and test samples.

- **Picking out the right surrogate model:** A model is selected upon its ability to describe, explain and predict the given function [21, 22]. Taking into account robustness, precision and execution speed, we chose a surrogate model. The selection criteria is problem dependant.

- **Building the surrogate:** The model hyper-parameters are computed.

- **Validating the model:** The prediction capability of the model is tested using the
Root Mean Squared Error (RMSE) on the test samples.

During the internship, we focused on Kriging models construction [1, 22, 28] as an analytical expression of their errors is provided. The Design Of Experiments is generated by a Latin Hypercube Sampling detailed in section 3.2. The Kriging model characteristics (Regression and Correlation) are chosen by the evaluation of the RMSE on the test samples (the best (regression, correlation) combination minimises the RMSE). The last two stages in building a Kriging model are the subject of sections 3.1.1 and 3.1.2.

### 3.1.1 Construction of the Kriging model

Consider a function $f$ that converts an input vector $\mathbf{x}$ into an output scalar $y$, $f(\mathbf{x}) = y$.

$$
\begin{align*}
\begin{array}{c}
\mathbf{f} : & \mathbb{R}^d \longrightarrow \mathbb{R} \\
\mathbf{x} & \longrightarrow f(\mathbf{x}) = y
\end{array}
\end{align*}
$$

The idea is to give the best guess of $f$. For this purpose, a set of inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots, \mathbf{x}^{(m)}$ whose respective outputs are $y^{(1)}, y^{(2)}, y^{(3)}, \ldots, y^{(m)}$ is used. $m$ is the number of learning samples.

Based on these known observations, an estimation $\hat{f}$ of $f$ is constructed. In the present report, the Kriging method is used to construct $\hat{f}$. The surrogate built by Kriging interpolates the data [1, 22] i.e:

$$
\hat{f}(\mathbf{x}^{(i)}; \mathbf{\theta}) = f(\mathbf{x}^{(i)}) \quad \forall (\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)})) \in \text{DOE}
$$

$\mathbf{\theta}$ being the surrogate hyper-parameters.

Writing $\mathbf{X} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots, \mathbf{x}^{(m)})$, and $\mathbf{Y} = (y^{(1)}, y^{(2)}, y^{(3)}, \ldots, y^{(m)})$, approximating a function by a kriging model implies [1, 2]:

$$
\mathbf{Y}(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{Z}(\mathbf{x})
$$

In equation 3.1, $\mu(\mathbf{x})$ and $\mathbf{Z}$ are a deterministic term and a random term respectively.

→ The deterministic term $\mu$

We can distinguish 3 different ways to describe $\mu$:

- **Ordinary Kriging** where $\mu(\mathbf{x})$ is an unknown constant
- **Simple Kriging** where $\mu(\mathbf{x})$ is a known constant that could be zero
• **Universal Kriging** where $\mu(x) = \sum_{j=1}^{p} w_j f_j(x)$

With $(f_1, \ldots, f_p)$ is a set of regression functions most commonly polynomial functions and $w_1, \ldots, w_p$ their respective weights.

→ The random term $Z(x)$ is a stationary Gaussian process whose mean value or first moment is null $E(Z(x)) = 0$ and whose variance is $\sigma^2_K$.

The learning samples are correlated using the basis function $\psi$ deduced by:

$$\psi^{(i,l)} = corr[x^{(i)}, x^{(l)}],$$

(3.2)

with $i, l \in [1..m].$

The correlation can be exponential and written as following [1, 22]:

$$\psi^{(i,l)} = \exp(-\sum_{j=1}^{d} \theta_j |x^{(i)}_j - x^{(l)}_j|^{p_j}).$$

(3.3)

Where $d$ is the dimension of an input vector $x$, $(p_j, \theta_j, j \in [1..d])$ are real parameters and $i, l \in [1..m]$. According to [22], the parameters $p_j, j \in [1..d]$ describes the 'smoothness' of the function $f$ and the parameters $\theta_j, j \in [1..d]$ indicates how 'active' the function $f$ is (See [22] for further details).

If each component of the vector $\theta = (\theta_1, \theta_2, \ldots, \theta_d)^T$ is equal to $\frac{1}{\sigma^2}$ and $p_j = 2, \forall j \in [1..d]$, we fall back on a Gaussian correlation kernel.

• $p_j = 1$ for $j \in [1..d]$ in (3.3) corresponds to an absolute exponential correlation

• $p_j = 2$ for $j \in [1..d]$ in (3.3) corresponds to a squared exponential correlation

• $p_j \in [0, 2]$ for $j \in [1..d]$ in (3.3) corresponds to a generalised exponential correlation

Other correlations can also be considered [28]. Equation 3.4 and equation 3.5 corresponds to linear and cubic correlations respectively.

$$\psi^{(i,l)} = \prod_{j=1}^{d} max\{0, 1 - \theta_j |x^{(i)}_j - x^{(l)}_j|\}$$

(3.4)

$$\psi^{(i)} = \prod_{j=1}^{d} 1 - 3\xi_j^2 + 2\xi_j^3$$

(3.5)
Where $\xi_j = \min\{1, \theta_j | x^{(i)}_j - x^{(l)}_j| \}$

The associated $m \times m$ correlation matrix is then $\Psi$ defined in matrix 3.6.

$$\Psi = \begin{bmatrix} \psi(1,1) & \ldots & \psi(1,m) \\ \vdots & \ddots & \vdots \\ \psi(m,1) & \ldots & \psi(1,1) \end{bmatrix}$$ (3.6)

Building the surrogate depends on the determination of the parameters $\theta_1, \theta_2, \ldots, \theta_d$ whichever the used correlation function is 3.4 or 3.5. For fixed $p_{(1,2,...,d)} = 2$ in equation 3.3, building a surrogate is also only dependent of determining the parameters $\theta_1, \theta_2, \ldots, \theta_d$.

Assuming that the likelihood [29, 30] of the set of parameters $\theta_j, j \in [1..d]$ given the learning samples $((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}))$ is written as equation 3.7.

$$L(y^{(1)}, \ldots, y^{(m)} | \mu, \sigma_K) = \left( \frac{1}{2\pi\sigma_K^2} \right)^{\frac{m}{2}} \exp\left[ -\frac{\sum (y^{(i)} - \mu)^2}{2\sigma_K^2} \right]$$ (3.7)

Which can also be written as a function of the correlation matrix 3.6 [22].

$$L = \left( \frac{1}{(2\pi\sigma_K^2)^{(m/2)}|\Psi|^{1/2}} \right) \exp\left[ -\frac{(Y - 1\mu)^T \Psi^{-1} (Y - 1\mu)}{2\sigma_K^2} \right]$$ (3.8)

$$\ln(L) = -\frac{m}{2} \ln(2\pi) - \frac{m}{2} \ln(\sigma_K^2) - \frac{1}{2} \ln |\Psi| - \frac{(Y - 1\mu)^T \Psi^{-1} (Y - 1\mu)}{2\sigma_K^2}$$ (3.9)

A first step will be to give a suitable estimations for both $\mu$ and $\sigma_K$ by setting the first derivative of the likelihood function, or more commonly its natural algorithm, to zero. By doing so, the resulting $\hat{\mu}$ and $\hat{\sigma}_K$ are given in equation 3.10 and equation 3.11 [22].

$$\hat{\mu} = \frac{1^T \Psi^{-1} Y}{1^T \Psi^{-1} 1}$$ (3.10)

$$\hat{\sigma}_K^2 = \frac{(Y - 1\mu)^T \Psi^{-1} (Y - 1\mu)}{m}$$ (3.11)

In order to determine the model parameters $(\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_d)$, the function to maximize is then the ln-likelihood function 3.12.

$$\ln(L) = -\frac{m}{2} \ln(\hat{\sigma}_K^2) - \frac{m}{2} \ln |\Psi|$$ (3.12)
The surrogate construction led to a difficult optimization problem as the objective function 3.12 is non differentiable [22, 28]. Different numerical optimization techniques can be used: a gradient-free local optimization algorithm COBYLA [27] or a gradient-free global optimization algorithm such as DE (Differential Evolution) [31] algorithm or DIRECT algorithm [17, 18].

→ Notes:
If the observed data admit ‘measurement errors’, the kriging model will not interpolate the data. We remedy this problem by adding a term $\sigma^2_s$ to the covariance matrix $\sigma^2_K \cdot \Psi$ diagonal [2]. The method is called the Nugget effect. It makes the surrogate easier to build when dimensions get large and if the costly function presents a highly non-linear behaviour.

Further details can be found in Adrianakis and Challenor 2012 [32]. As Kriging modeling implies a matrix inversion (correlation matrix 3.6), building a surrogate gets time consuming and difficult when the problem dimensions are large.

To sum up, the Kriging model construction consists in:

- Get a set of learning samples (from experiments or exact measures)
  \((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\)
- Define the regression model 3.1
- Define the correlation model \((p_j, j \in [1..d])\)
- Estimate $\hat{\mu}$ and $\hat{\sigma}_K$
- Determine the Kriging hyper-parameters $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_d$.

### 3.1.2 Kriging prediction

Once the surrogate model is constructed using the learning sample, one is able to predict an output value \(y\) corresponding to an input vector \(x\) i.e evaluate $\hat{y} = \hat{f}(x)$. A correlation vector 3.13 between the new predicted value and the observed data enable to construct an augmented correlation matrix 3.14.

$$\psi = \begin{bmatrix}
\text{cor}[x^{(1)}, x] \\
\text{cor}[x^{(2)}, x] \\
\vdots \\
\text{cor}[x^{(m)}, x]
\end{bmatrix}$$  \hspace{1cm} (3.13)
Kriging models construction and kriging based optimization

\[\tilde{\Psi} = \begin{bmatrix} \Psi & \psi \\ \psi_T & 1 \end{bmatrix}\] (3.14)

In the same way the model has been constructed, the likelihood of the augmented data, is maximized in order to determine the unknown variables introduced by \(\hat{y}\) [22].

\[\ln(L) \approx \frac{\begin{pmatrix} Y - 1\hat{\mu} \\ \hat{\mu} - \hat{\mu} \end{pmatrix}^T \begin{pmatrix} \Psi & \psi \\ \psi_T & 1 \end{pmatrix}^{-1} \begin{pmatrix} Y - 1\hat{\mu} \\ \hat{y} - \hat{\mu} \end{pmatrix}}{2\sigma^2}\] (3.15)

It is shown that \(\ln(L)\) (equation 3.15) is a quadratic function of \(\hat{y}\) [22] [2]. Setting the derivative to zero permits to find the optimizer. \(\hat{y}\) is then:

\[\hat{y}(x) = \hat{\mu} + \psi^T \Psi^{-1} (Y - 1\hat{\mu})\] (3.16)

The variance in \(x\) can also be estimated and used as a prediction error. Thus \(\hat{\sigma}^2(x)\) is written in equation 3.17 [2].

\[\hat{\sigma}^2(x) = \sigma^2 - \psi(x)^T \Psi^{-1} \psi(x)\] (3.17)

Being able to estimate \(\hat{\sigma}^2(x)\) is one of the advantages of building a Kriging model over other surrogate models.

Using equations 3.16 and 3.17, the 95% confidence interval is analytically written as:

\[CI^{0.95}(x) = [\hat{y}(x) - 1.96\hat{\sigma}(x), \hat{y}(x) + 1.96\hat{\sigma}(x)]\] (3.18)

A representation of a Kriging model in a one dimensional design space is given in figure 3.1 taken from [1].

![Figure 3.1: Example of a Kriging surrogate](image-url)
A test sample \((X_t,Y_t)\) should be kept aside while building the surrogate in order to test the constructed model.

The relative error (3.19) is an important output parameters one could look at.

In this study, we approved a surrogate model once its maximum relative error is under 5%.

\[
err_r = \max_{x^{(i)} \in \text{DOE}} \left( \frac{\hat{f}(x^{(i)}) - f(x^{(i)}) + \epsilon}{f(x^{(i)}) + \epsilon} \right)
\]  

(3.19)

with \(\epsilon\) a positive real number used to avoid division by zero.

It is expected that the larger the learning sample (data used to construct the surrogate) the more \(\hat{f}\) approaches \(f\) and the lower the maximum relative error is. Adding data at some point to build the surrogate wont improve the model globally but only locally. In [22] Alexander I.J.Forrester, Andras Sobester and J.Keane compare it to adding sugar to a cup of tea.

However, samples can be costly and the available samples can be limited. It is therefore interesting to minimize the DOE size while keeping the surrogate model quality.

In the next session a way to expand the DOE is presented.

### 3.2 Construction of a DOE by sequential enrichment

We start by describing the method in section 3.2.1 before applying it to a set of test functions in different dimensions in section 3.2.2.

#### 3.2.1 Method description

The aim is to form an optimal DOE for the surrogate construction.

Instead of increasing blindly the DOE size and therefore risking to describe thinly a domain area at the cost of loosing important information in another one, a suitable point can be added to an DOE. The points added to an initial DOE are meant to cause a decrease of the surrogate error.

The chosen selection criterion is the Mean Squared Error \(MSE\) (equation 3.20).

\((x^*, f(x^*))\) is added to the DOE. \(x^*\) being the maximizer of \(MSE_x: x \rightarrow MSE(x)\) 3.20.

A sample cannot be redundant. In fact, a vector \(x^*\) such that \(MSE_x(x^*) = MSE_{max}\) could not have been used to construct the model itself.

Expanding the experimental design using \(MSE\) introduces \(f\) characteristics into the model construction [2]. A smaller number of observables is then needed to achieve the requirement \(err_{r,max} < 5\%\). The relative error is then tracked and its decrease witnessed.

\[
MSE_x(x) = \hat{\sigma}(x)
\]  

(3.20)
The disadvantage of using the Mean Squared Error is enriching the experimental design around the domain borders where a Kriging model results in high \( MSE \) values [2]. Other criteria could be used such as the Integrated Mean Squared Error \( IMSE \) presented in [2].

In case the initial DOE can be chosen and is not determined by experimental techniques, an initial LHS is profitable. The learning samples are then evenly distributed over the domain. There is no risk to add too close points into the DOE.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure3_2.png}
\caption{Example of LHS sampling}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure3_3.png}
\caption{DOE Space filling: LHS}
\end{figure}

The following scheme draws the sequential enrichment of the experimental design steps.
3.2.2 Test functions and results

Three different test functions are: The 2D Rosenbrock function 3.21, the camel back function 3.22 [2] and the generalized multidimensional Rosenbrock function 3.23.

\[
f_a(x, y) = (1 - x)^2 + 100(y - x^2)^2, \quad (x, y) \in [-5, 5] \tag{3.21}
\]
\[
f_B(x, y) = (4 - 2x^2 + \frac{x^4}{3})x^2 + xy + 4y^2(y^2 - 1), \quad (x, y) \in [-2, 2] \tag{3.22}
\]
\[
f_\gamma(x_{i, j \in [1..D]}) = \sum 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, x_i \in [-5, 5] \forall i \in [1..D] \tag{3.23}
\]
As a first step, we split a set of 100 or 200 samples \((X, f(X))\) (depending on the test function. 100 data are used for 2D examples and 200 for the 7D example) into a subset \((X_{\text{learn}}, f(X_{\text{learn}}))\) of data used to construct the model and a subset \((X_{\text{test}}, f(X_{\text{test}}))\) used to test the built model. More specifically, one third of \((X, f(X))\) is used as \((X_{\text{learn}}, f(X_{\text{learn}}))\) and two thirds are used as \((X_{\text{test}}, f(X_{\text{test}}))\). The best correlation and regression couple is then determined.

→ Possible Correlations

Absolute exponential correlation, squared exponential correlation, generalized exponential correlation, cubic correlation or linear correlation.

→ Regression can be Constant, linear or quadratic.

The choice criterion being the couple producing the minimal Root Mean Squared Error RMSE.

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{m} |\hat{f}(x^{(i)}) - f(x^{(i)})|^2}{m}} \quad (3.24)
\]

Table 3.1 gives the best correlation and regression for each test function based on LHS. The best correlation and regression combination is such that the RMSE of the test samples is minimal. The maximum error \(err_{\text{max}}\), the maximum relative error \(err_r\) and the RMSE are given for each case. Calculation times are also specified.

<table>
<thead>
<tr>
<th>Test function</th>
<th>Regression</th>
<th>Correlation</th>
<th>RMSE</th>
<th>(err_{\text{max}})</th>
<th>(err_{r})%</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosenbrock 2D</td>
<td>quadratic</td>
<td>squared-exponential</td>
<td>0.0079</td>
<td>0.044</td>
<td>0.13</td>
<td>7.23</td>
</tr>
<tr>
<td>Camel back</td>
<td>quadratic</td>
<td>cubic</td>
<td>0.34</td>
<td>1.58</td>
<td>522.186</td>
<td>4.06</td>
</tr>
<tr>
<td>Rosenbrock 7D</td>
<td>constant</td>
<td>cubic</td>
<td>11094.32</td>
<td>57449.55</td>
<td>72.66</td>
<td>182.87</td>
</tr>
</tbody>
</table>

Table 3.1: RMSE, \(err_{\text{max}}\) and \(err_{r}\) for each test function
Let’s consider the regression and correlation couples found in Table 3.1 for each test function and perform a sequential enrichment of initial DOEs. For both two dimensional functions, we start from an initial DOE of 10 learning samples. 50 points form the initial DOE of the multidimensional case. 100 and 500 are used as test samples for the two dimensional cases and the seven dimensional case respectively. Both test samples and initial DOEs are generated from a Latin Hypercube Sampling (LHS) of the design space. We represent the maximum relative error on tested data as the experimental design gets larger. The predictions on the test data are also represented. As a basis of comparison, predictions on the test data issued from a model built from a DOE which size is as big as the final DOE size reached by sequential enrichment are given. The calculation are made several times to limit the random effects induced by the LHS.

![Figure 3.6: Expanding an initial DOE: 2D Rosenbrock Function](image)

![Figure 3.7: Expanding an initial DOE: Camel Back Function](image)

In figures 3.6c and 3.7c, the initial DOE and the added elements are shown. One can see how expanding the experimental design based on the maximum MSE criteria tend to enrich the domain border. The phenomenon grows as the dimension gets larger. The "savings" done on the DOE size in order to achieve the requirement \( \text{max} < 5\% \) are not significant.

The surrogate construction has been done using the scikit-learn library on Python [33]. scikit-learn is a free tool for data mining and data analysis. Its use is simple and
accessible. More specifically, we used the **Gaussian process for Machine learning** (GPML) module in building Kriging models.

### 3.3 Efficient Global Optimization (EGO)

In this section, a surrogate based optimization algorithm is presented in 3.3.1. In this case the surrogate is used in order to find a function optimal value. The EGO algorithm is then tested on a set of given functions in 3.2.

#### 3.3.1 Algorithm presentation

In the present report, the Efficient Global Optimization (EGO) algorithm is considered. Starting from an initial DOE and constrained by a maximum DOE size, the EGO algorithm perform a sequential enrichment of the DOE. Each added element moves closer to the costly function minimizer. The EGO algorithm uses a criterion suited to the purpose of finding the function minimizer. The criterion introduced by Jones [1, 34, 35] is the Expected Improvement EI. Jones puts into equation the idea of adding a DOE element improving the knowledge of the optimal function value. The selection criterion is then written as [1, 34, 35]:

\[
EI_x(x) = E[\max(0, Y(x) - f_{\text{min}}(x))|DOE]
\]

(3.25)

Where \(E\) is the expected value of a random variable

And

\[f_{\text{min}} = \min_{x^{(i)} \in DOE} f(x^{(i)})\]

\(Y(x)\) is the only random variable in 3.8 and it has a Gaussian distribution \(N(\tilde{f}(x), \tilde{\sigma}(x))\) respectively to the Design Of Experiments. The Expected improvement function can then be analytically calculated.

\[
EI_x(x) = [f_{\text{min}} - \tilde{f}(x)] \Phi\left[\frac{f_{\text{min}} - \tilde{f}(x)}{\tilde{\sigma}(x)}\right] + \tilde{\sigma}(x) \phi(0,1) \left[\frac{f_{\text{min}} - \tilde{f}(x)}{\tilde{\sigma}(x)}\right]
\]

(3.26)

Where \(\Phi\) the cumulative density function and \(\phi\) the probability density function.

A schematic representation of \(EI\) function for a 1 dimensional case is given in figure 3.8.
Further details can be found in [34] and [35]. The EGO algorithm implementation is detailed in Algorithm 5.

**Algorithm 5** Surrogate based global optimization

**input**: Initial DOE size $N_{init}$
- Maximum DOE size $N_{max}$
- DOE enrichment criterion $EI$

**output**: $x_{optimal}$
- $f_{min} = f(x_{optimal})$

Generate the initial DOE $X_{init}$: LHS
Evaluate $Y_{init} = f(X_{init})$
$N \leftarrow N_{init}$
$DOE \leftarrow (X_{init}, Y_{init})$

**while** $N < N_{max}$ **do**
- Build a surrogate model $\hat{f}$
- Determine $x^*$ maximizer of the function $EI_x$ 3.8
- Evaluate $f(x^*)$
- $DOE \leftarrow (DOE, (x^*, f(x^*)))$

**end**

Return $f_{min} = \min_{x \in DOE} f(x)$
$x_{optimal} = \arg\min_{x \in DOE} f(x)$

### 3.3.2 Test functions results

The EGO algorithm is applied to the two first test functions ($f_\beta$ and $f_\gamma$) used in section 3.2.2 namely the two dimensional Rosenbrock function equation 3.21, the Camel back...
function equation 3.22.

For both two dimensional functions, the initial DOE is made up 10 learning samples and 50 is the maximum number of learning samples (DOE budget).

<table>
<thead>
<tr>
<th>Function</th>
<th>Expected minimizer (s)</th>
<th>EGO result</th>
<th>T (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosenbrock function</td>
<td>(1 , 1)</td>
<td>(1.00137166, 1.0028959)</td>
<td>17468.53</td>
</tr>
<tr>
<td>Camel-back function</td>
<td>(0.0898,-0.7126),(-0.0898,0.7126)</td>
<td>(0.08962076, -0.71268871)</td>
<td>2845.17</td>
</tr>
</tbody>
</table>

Table 3.2: EGO algorithm results

Table 3.2, shows that using 60 function evaluations for the 2D functions, the EGO algorithms reaches the global minimizer of the function with a negligible error.
4. **Inverse problem resolution using surrogate models**

The final goal is to speed up the optimization process 2.2.3. Instead of using the optimization objective function, an approximation of the objective function is called in the optimization process. The surrogate model is less costly to evaluate. It describes and predicts the function to a certain extent. The criteria $RMSE$, the maximum error $err_{max}$, and the relative maximum error $err_{max, r}$ are used to evaluate a surrogate fitness. Using a 'rather good' surrogate (according to the former criteria) in the optimization process, we aim for the inverse problem solution to exist and to be stable. The two sub-objectives of this chapter 4 are then:

1. Build a model suited for the optimization problem
2. Get a stable solution when solving the new optimization problem (including a surrogate)

First, we consider two ways to build the surrogate. Either $U_0$ only is taken as the input vector of the surrogate (see section 2.2.1) or both $U_0$ and $\lambda$, the regularisation parameter, are the surrogate inputs (see section 2.2.1).

In sections 4.1 and 4.2, the two former options are studied.

Second, we choose in section 4.3 to apply a surrogate based optimization algorithm used to minimize (or maximize) a function 3.3 and try to adapt it to the present inverse problem resolution.

### 4.1 N dimensional surrogate-based optimization

#### 4.1.1 Problem statement and method description

Consider the objective function related to the inverse heat conduction problem stated in chapter 2: $S(U_0)$. Using what is presented in chapter 3, a surrogate model of the function $S$ as a function of $U_0$ can be constructed. Let’s write the surrogate as $\hat{S}(U_0)$. The purpose is to use a less costly function $\hat{S}(U_0)$ instead of $S(U_0)$ when applying the optimization process: here the DIRECT algorithm 2.2.3. And consequently to speed up the optimization process which has proven to be slow compared to a gradient based optimization process based on the Gauss-Newton Method 2.2.2.
As the problem is ill posed and needed regularization to be solved, different approaches are used when building the surrogate model. The list below sums up the following options.

1. \( L^2 \) regularised objective function \( \hat{S}(U_0) \):

   Building the Kriging model \( \hat{S}(U_0) \) of the non regularised objective function:
   \[
   S(U_0) = \| U - U_k(U_0) \|^2
   \]

   The optimization problem is then written as:
   \[
   \min_{U_0 \in \mathbb{R}^N} \hat{S}(U_0) + \lambda \| U_0 \|^2
   \]

2. \( TV \) regularised objective function \( \hat{S}(U_0) \):

   The optimization problem is written as:
   \[
   \min_{U_0 \in \mathbb{R}^N} \hat{S}(U_0) + \lambda \| \nabla U_0 \|_{L^1}
   \]

3. Building the Kriging model \( \hat{S}_{L^2}(U_0) \) to approximate the function \( S_{L^2}(U_0) \) in equation 4.1
   \[
   S_{L^2}(U_0) = \| U - U_k(U_0) \|^2 + \lambda \| U_0 - U_{0ref} \|^2
   \]  
   (4.1)

   The optimization problem is then:
   \[
   \min_{U_0 \in \mathbb{R}^N} \hat{S}_{L^2}(U_0)
   \]

4. Building the Kriging model \( \hat{S}_{TV}(U_0) \) to approximate the function \( S_{TV}(U_0) \) in equation 4.2
   \[
   S_{TV}(U_0) = \| U - U_k(U_0) \|^2 + \lambda \| \nabla U_0 \|_{L^1}
   \]  
   (4.2)

   The optimization problem is then:
   \[
   \min_{U_0 \in \mathbb{R}^N} \hat{S}_{TV}(U_0)
   \]

\( \rightarrow \) Notes
In the different options presented above, the regularization parameter \( \lambda \geq 0 \) is fixed to a certain value.

The observed data \( u_i = u(x_i), i \in [1..M] \) at the points \( x_i \in \mathbb{R} \forall i \in [1..M] \) are fixed and
given from experimental results.

Investigating different approaches permits to identify issues and decide on improvements and changes. Non satisfying approaches are identified.

4.1.2 Results discussion

Using an approximate of the non-regularized objective function is predictably an unsatisfying option. As the non-regularized function lead to an unstable solution when measurement errors are made in section 2.3 figure 2.1b, a non-regularized surrogate model does not conduct to a valid solution.

Results show that the problem inversion lead to an unstable solution even when using a regularized surrogate (options 1 and 2) or a model of the regularized function (\( \hat{S}_{L_2} \) option 3 and \( \hat{S}_{L_2} \) option 4). Either way, the surrogate model is only an approximation of the function. A critical point (a minimizer for example) of the function is very unlikely to be a critical point of surrogate model. The phenomenon of instability is overcome when the surrogate 'perfectly' describes and predicts the initial objective function. More specifically, the indicators \( (RMSE, error_{max}, error_{rel_{max}}) \) reach very low values. In fact, taking a squared regression, when building the surrogate model, perfectly suits the objective function characteristics. The former are not necessary consistent with the set goal: Constructing a 'good enough' surrogate to obtain a stable solution to the inverse problem.

4.2 N+1 dimensional surrogate-based optimization

4.2.1 Problem statement and method description

We justify moving from taking as input vector of the surrogate \( U_0 \) to taking the input vector \( (U_0, \lambda) \) by:

- As presented in chapter 2 section 2.2.3, the DIRECT algorithm is called in a sequential process based on the regularisation parameter \( \lambda \) decrease. If we use approaches 4 or 5 presented in 4.1 in the recursive process, a reconstruction of a surrogate is then needed at each step which can compromise the final goal of speeding up the resolution. Taking the input vector \( (U_0, \lambda) \) appears to be a promising option as only one surrogate is built and used in the recursive optimization process. The disadvantage is though increasing the dimension the surrogate input.
The methods used in 4.1 including approaches 4 and 5 are proven to be ineffective unless the surrogate describes ‘perfectly’ the related function in 4.1.2. The ineffectiveness of different approaches is here dependent on the test case presented in chapter 2 along with the surrogate construction method presented in chapter 3 and the inverse problems inversion tools in chapter 2 section 2.2.3.

Let’s consider the regularised objective functions given in chapter 2 section 2.2.1: equation 2.8 and equation 2.10. We call $\hat{S}_{L^2}$ and $\hat{S}_{TV}$ surrogate models of the functions 4.3 and 4.4 given bellow:

$$S_{L^2}(U_0; \lambda) = \|U - U_k(U_0)\|^2 + \lambda \|U_0 - U_{ref}\|^2$$ (4.3)

$$S_{TV}(U_0; \lambda) = \|U - U_k(U_0)\|^2 + \lambda \|\nabla U_0\|_{L^1}$$ (4.4)

→ Note

$\lambda$ is here part of the input vector of the surrogate model. This means that if $N$ is the dimension of $U_0$, the input space of the surrogate is $N + 1$ dimensional. Each learning sample or test sample is then written as $((U_0, \lambda), S(U_0; \lambda))$, $S$ being either $S_{L^2}$ or $S_{TV}$.

The surrogate models $\hat{S}_{L^2}$ and $\hat{S}_{TV}$ are then used in the optimization process. The DIRECT algorithm is:

- either called in a sequential process decreasing the regularisation parameter $\lambda$.
- or used in a single optimization process able to get to an optimal vector $U_{0_{optimal}}$ and an optimal regularization parameter $\lambda_{optimal}$

### 4.2.2 Results and discussion

We first note that performing a single optimization process to get to an optimal vector $U_{0_{optimal}}$ along with an optimal regularization parameter $\lambda_{optimal}$ is a non viable solution.

In fact, the DIRECT algorithm straightforwardly head to the lower bound set for the regularization parameter $\lambda$. As a result, the method is equivalent to a non-regularized method using a surrogate instead of the costly function (which is already proven to be ineffective 4.1.2).

The second option is to build a surrogate on a $N + 1$ dimension design space i.e the input vector is $(U_0, \lambda)$. The surrogate model is then called in a sequential process following the steps bellow:
1. Fix the regularization parameter $\lambda$

2. Perform a DIRECT optimization on a $N$ dimension hypercube to get $U_{0_{\text{optimal}}}$ calling the surrogate $\tilde{S}_{L^2}(U_0, \lambda)$ or $\tilde{S}_{TV}(U_0, \lambda)$

3. Reset DIRECT optimization bounds

4. Reset the regularization parameter $\lambda \leftarrow \frac{\lambda}{2}$

5. Go back to 2 if the end conditions are not met

The results obtained after one iteration are given in figure 4.1. Both $L^2$ and $TV$ regularization techniques are used. The regularization parameter is fixed: $\lambda = 1$. The Kriging model is built on a $7+1$ dimensional design space. The DIRECT algorithm is applied on a 7 dimensional hypercubes. The inverse problem is constructed based on 30 measurements.

20 calculations issued from different LHS generated DOEs are done. Resulted $U_{0_{\text{optimal}}}$ vectors are shown in a box-plots form.

![Box-plots](image)

**Figure 4.1:** Surrogate based resolution results

According to 4.1 and similar calculations (various dimensions), the first iteration of the studied method is a success. The obtained vector $U_{0_{\text{optimal}}}$ is similar to that obtained in 9a and 2.8a for $L^2$ regularization and $TV$ regularization respectively. Results also show the method to be robust with respect to the DOE.

However, if we can be content with only one iteration when using $TV$ regularization (see figures 2.8a and 4.4b), many successive iterations are needed for the $L^2$ regularised problem. Results obtained after more than one iteration are not viable as the inverse problem solution moved from the correct solution. The issue is preserving the information (related to the minimizer) when moving from one iteration to another. Thus, among others the questions one might ask are when to stop an iteration so that the information is kept? Does the surrogate model looses the information through out iterations?
4.3 Surrogate-based optimization with DOE enrichment

In sections 4.1 and 4.2, a surrogate model is called instead of the objective function in the optimization process. Either built from a LHS DOE or a sequential enrichment of the DOE 3.2, a surrogate model is not a perfect reproduction of the costly function. The model used in an optimization process is not guaranteed to converge towards neither a global optimum or a local optimum of the function which is shown in sections 4.1 and 4.2 results. The optimum (minimum or maximum) of the surrogate is very unlikely to be a critical point of the high cost evaluation function. An alternative choice is a sequential use of the EGO algorithm presented in section 3.3 described in 4.3.1 and tested in 4.3.2

4.3.1 Sequential use of EGO algorithm

Instead of calling a surrogate instead of the function in an optimization process, we can decrease the number of evaluations via evolutionary algorithms more precisely the Evolutionary Global Optimization (EGO) [1, 34, 35] algorithm presented in this report section 3.3 and proven to be effective on test functions.

Simply put, we use an algorithm able to converge towards the global minimizer section 3.3. Then, we try to apply it to the inverse heat conduction problem.

The EGO algorithm is there called sequentially. For each regularization parameter $\lambda$, the objective function $S_{L^2}$ or $S_{TV}$ is minimized using the EGO algorithm. As it has been done for both Gradient based algorithms ($ITGN$ and $TVRGN$) and derivative-free algorithms ($DIRECTL^2$ and $DIRECTTV$), the regularized parameter is then decreased and the search domain reinitialized.

Writing the objective functions as:

$$S_{L^2}(U_0|\lambda, \mathcal{D}) = S(U_0) + \lambda R_{L^2}(U_0) \quad (4.5)$$

or

$$S_{TV}(U_0|\lambda, \mathcal{D}) = S(U_0) + \lambda R_{TV}(U_0) \quad (4.6)$$

Where: $\lambda$ is the regularization parameter and $\mathcal{D}$ is the design space where the minimizer is looked at.

Let’s write $U^*_{0|\lambda, \mathcal{D}} = \arg\min_{U_0 \in \mathcal{D}} S_{L^2}(U_0|\lambda, \mathcal{D})$ and $U^*_{0|\lambda, \mathcal{D}} = \arg\min_{U_0 \in \mathcal{D}} S_{TV}(U_0|\lambda, \mathcal{D})$ the minimizers of functions 4.5 and 4.6 respectively for a given regularization parameter $\lambda$ and a given search space $\mathcal{D}$.

The size of the initial DOE being set and the budget in added samples being fixed, the recursive use of the EGO algorithm is presented in the flowchart 4.2. The regularization
parameter $\lambda$ and the search space $D$ are updated as following:

$$\lambda \leftarrow \frac{\lambda}{2}$$
$$D \leftarrow [0.5 \times U_0^*, 1.5 \times U_0^*]$$

![Sequential use of EGO algorithm](image)

**Figure 4.2**: Sequential use of EGO algorithm

### 4.3.2 Method results

The sequential use of the EGO algorithm is applied to both the TV regularised problem and the $L^2$ regularised problem.

The maximum number of added learning samples is 30.

An additional end condition is considered. If the minimum value of the objective function is constant through the last five added learning samples, the iteration is completed.

For the $L^2$ regularized problem resolution gives results comparable to results given by a non-surrogate based optimization algorithm through the first iterations (5 to 8 iterations).

However, the same problem noticed in section 4.2.2 occurs. The result moves from the right solution. When moving from one iteration to another, the information related to
the minimizer is lost.

The main reasons that could prevent the sequential process to converge are numerical errors or implementation errors. In addition, one can think of enriching the design of experiments with a larger number of learning samples. This option is investigated but do not seem to be convincing. Furthermore, the Krigeing model might not be the most accurate surrogate. The EGO algorithm can be adapted to other surrogate models such as support Vector Regression Models [35].

For the TV regularized formulation of the inverse problem, the sequential use of the EGO algorithm seem to be efficient especially as it needs less iterations before convergence. However, the method needs to be tested on other inverse problems and for larger dimensions.

In figure 4.3, the results corresponding to a $L^2$ regularized problem are given. We show the progress of the optimal vector $U_0$ through the 5 first iterations (corresponding to a particular $\lambda$) of the sequential use of EGO given a set of $N = 30$ $u$ function measurements. The results are similar to the first iterations obtained when the DIRECT algorithm is used in 2.2.3. However, starting from iteration 5, the vector $U_0$ starts to move away from the correct solution.

In figure 4.4, the results corresponding to the TV regularized formulation of the inverse problem are given. After 6 iterations of the sequential process, the methods seems to converge. Yet, the results are to be improved especially as the method is not robust when initial measurements vary.

Figure 4.3: Sequential use of EGO on the $L^2$ regularized problem
Inverse problem resolution using surrogate models

Figure 4.4: Sequential use of EGO on the TV regularized problem

(A) $U_0^{\text{optimal}}$ through 6 iterations

(B) $U_0^{\text{final}}$
5. **Improvements and discussion**

In this chapter 5, each step in achieving this master thesis is discussed. Chapters 3,2 and 4 are first commented one at a time. Kriging models pros and cons associated with different examples took in this report are given. A more accurate heat inverse problem is also stated. Finally, comments on how surrogate models are used in inverse problems resolution are provided.

5.1 **Discussing surrogate models**

Kriging models have the advantage of being flexible through different possible combination of regression and correlation functions making the user able to approximate a wide variety of functions presenting different characteristics. Another important advantage in approximating a given function by a Kriging model is having an estimation of the variance at each input vector $\hat{\sigma}(x)$.

However, building a Kriging model can get time consuming if the problem dimensions are large. Estimating the $d$ Kriging hyper-parameters $(\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_d)$ is a $d$-dimensional optimization problem which gets more costly when the data set is large.

Building a Kriging model implies a matrix inversion. The DOE needs to be well chosen to avoid singularities. In fact, the correlation matrix can become singular when the sample points are close one another. Expanding an initial design of experiments needs then to be done carefully.

An important idea related to models building and stated in this present report is the DOE sequential enrichment either using the MSE criteria in chapter 3 section 3.2. More importantly, the surrogate based algorithm EGO uses the $EI$ criteria to add an sample to the DOE in chapter 4 section 3.3. Regardless of the target: optimizing the DOE size to construct a model or finding the optimizer of a given function, both methods are iterative. Each iteration starts if and only if the previous iteration is completed. Assuming the function estimation to be costly, the surrogate building and then the optimization process risks to be time consuming. Yet, one can think of a choice criteria adding at each step more than one point to the initial DOE. One calculation of the costly function at a given new point is independent of other calculations. An idea is then to use parallel computing. Thus, many calculations are carried out simultaneously. Parallel computing can also be used on the initial DOE. This computational technique can speed up the surrogate construction and eventually the optimization process.

The use of Kriging models could also be discussed when the related costly function is time and space dependant. For example, in [11], the authors prove the Kriging model to
be less effective than other surrogate models such as Radial basis functions and feedforward neural networks models on the article test cases. In [36], different metamodelling techniques (polynomial regression, multivariate adaptive regression spline, radial basis functions and kriging) are compared based on various performance criteria. In [37], Y.Jin discusses the use of approximation models in evolutionary algorithms (EAs) and raises the related main issues. Depending on the studied costly function $f$ and the surrogate model use, a suited approximation of $f$ may then be found.

5.2 Considering the 3 dimensional heat conduction inverse problem

A discretisation of the 3 dimensional problem is obviously needed in order to apply a given algorithm. In [11], a finite element formulation of the direct problem is applied to approximate temperature function. The continuous time is also discretised into time steps. In [11], autors use a sequential implementation. Sequential estimation techniques are generally used, however in [11], a future time step regularisation is chosen, explained and detailed (see [11]). However, we note that the used techniques are problem dependant. Moreover, other derivative-free based optimization methods can be used to solve inverse problems and more specifically heat transfer inverse problem. According to the literature, evolutionary algorithms are of a particular interest in both academy and industry. Genetic algorithm (GN) is used in [11]. Particle swarm optimization (PSO) algorithm is used in both [11] and [10]. Ant colony optimization algorithm (ACO) is used in [9].

5.3 Discussing inverse problems resolution via surrogate models

Different derivative-free optimization methods are proven to be efficient when solving an inverse problem. In this study (chapter 2 section 2.3), the DIRECT algorithm combined to $L^2$ and $TV$ regularization techniques reaches acceptable to good solutions regardless of initial measurements and committed measurement errors as well as inverse problem dimension. The tricky step is to put together surrogate modelling and inverse problems resolution as it raises various issues.
Constructing an approximation \( \hat{f} \) of a given costly function \( f \) is more or less reachable depending on the fitness requirements set by the user. However, a surrogate model should prevent the 'information' from being lost through the optimization process. For example in chapter 4 sections 4.1 and 4.2, different methods turn out to be inconclusive. The surrogate models built either by \( LHS \) generation of the DOE or by a \( DOE \) sequential enrichment using the MSE criteria do not necessary keep the information regarding the function minimizer. The optimum of surrogate models through successive iterations may not be the optimum of the related costly objective functions.

If using the regularization parameter \( \lambda \) as an additional dimension in the input definition domain 4.2 of the surrogate models leads to a viable solution on the first iteration of the sequential process, the correctness of the solution is lost in the higher level iterations. We have up to now no certainties associated to parameters that effect the results. A poor model quality or unsuitable iteration end conditions could be looked at to develop the idea.

Another idea discussed in chapter 4 section 3.3 is found promising. The sequential enrichment of the initial design of experiment (DOE) is done according to the Expected Improvement (\( EI \)) criteria. Choosing the \( EI \) criteria is such that the added DOE element keeps the information regarding the function minimum and moves closer to the related minimize. Consequently, the results show a better performance when applying the sequential EGO based methods 4.2. However, the convergence of such a method is still to be reached. When the optimization algorithm needs a large number of iterations, the convergence is no longer guaranteed. The questions that arises are: When an iteration should be completed? Which stopping criteria should be used in order to ensure suited initial conditions (the information is kept) to the next iteration?

Other issues could prevent the sequential use of the EGO algorithm to converge such as numerical problem, implementation errors or surrogate issues. In fact, the EGO algorithm was developed for Kriging surrogates. But as Kriging is not necessary the most accurate surrogate depending on the studied problem. The EGO algorithm can also be implemented for more accurate surrogates such that support vector surrogate which is done by Felipe.A.C Viana in [35].
6. Conclusions

The final goal set for this project is to solve a heat conduction inverse problem using surrogate models. If the first steps towards achieving this goal do not present major difficulties, putting different studied aspects together is still to be improved.

Kriging models are constructed and shown to be able to approximate various functions regardless of their characteristics but only based on a set of learning samples (DOE). Yet, Kriging models construction is as difficult and time consuming as the problem dimensions get larger. Depending on the study case, other surrogates could be proven to be more effective. A user is then very likely to approximate a given costly function by a satisfactory surrogate model.

Solving inverse problem is a key issue in the present study. If regularizing this ill posed problem is inevitable especially when measurement errors are made, different regularization techniques could be used. The Total Variation (TV) regularization is shown to be more ‘efficient’ (according to robustness, precision and execution time) than the $L^2$ regularization independently of the used optimization algorithm but regarding the studied case.

Gradient-based algorithms and derivative-free algorithms tend to reach comparable solutions. However, if the gradient of the studied function is unknown and only its evaluation is possible, the viable option is to use the DIRECT algorithm based methods. The drawback is thus a far greater computational time. Surrogate models are then used to speed up the optimization process. However, the questions that raised from this idea are related to which objective function should be approximated and which input vector should be considered.

Taking the regularization parameter $\lambda$ as part of the input space is a promising option regarding the project results. Using a surrogate based optimization algorithm (EGO) is also shown to be promising. However, no conclusion can yet be drawn regarding speeding up the optimization process as the convergence of recursive methods used is still to be guaranteed.

For future work, iterations stopping criteria can be improved. A deeper study of parameters dependencies and different types of errors (measurement errors, surrogate errors...) is needed. Moreover, a more realistic engineering issue can be set up.
Bibliography


Appendices

Results for increased errors

Figure 1: $u$ function measurements with an average error of 10%

Figure 2: IRGN method

Figure 3: TVRGN method
Appendices

**Figure 4:** DIRECT $L^2$ method

(A) $U_0$ through iterations  
(B) Final $U_0$  
(C) Objective function

**Figure 5:** DIRECT $TV$ method

(A) $U_0$ through iterations  
(B) Final $U_0$  
(C) Objective function
Results for a larger dimensions

Figure 6: $u$ function measurements with an average error of $2\%$

(A) Final $U_0$  
(B) Objective Function

Figure 7: IRGN method

(A) Final $U_0$  
(B) Objective Function

Figure 8: TVRGN method
Appendices

Figure 9: DIRECT $L^2$ method

Figure 10: DIRECT TV method