FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
AUTOMATION AND SYSTEMS DEPARTMENT
UNDERGRADUATE COURSE IN CONTROL AND AUTOMATION ENGINEERING

Amanda Souza Machado

Ensemble Trust-Region Optimization

Florianópolis
2020

Amanda Souza Machado

Ensemble Trust-Region Optimization

Final report of the subject DAS5511 (Course Final Project) as a Concluding Dissertation of the Undergraduate Course in Control and Automation Engineering at the Federal University of Santa Catarina in Florianópolis.
Supervisor:: Prof. Eduardo Camponogara, PhD.
Co-supervisor:: Thiago Lima Silva, PhD.

Florianópolis
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Machado, Amanda Souza
Ensemble Trust-Region Optimization / Amanda Souza Machado ; orientador, Eduardo Camponogara, coorientador, Thiago Lima Silva, 2020.
77 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2020.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Well control optimization. 3. Geological uncertainties. 4. Trust-Region Optimization. 5. Reservoir Management. I. Camponogara, Eduardo. II. Silva, Thiago Lima. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Amanda Souza Machado

Ensemble Trust-Region Optimization

This dissertation was evaluated in the context of the subject DAS5511 (Course Final Project) and approved in its final form by the Undergraduate Course in Control and Automation Engineering

Florianópolis, Augost 10, 2020.

Prof. Hector Bessa Silveira, PhD.
Course Coordinator

Examining Board:

Prof. Eduardo Camponogara, PhD.
Advisor
UFSC/CTC/DAS

Thiago L. Silva, PhD.
Supervisor
NTNU/IGP

Prof. Julio Normey Elias, PhD.
Evaluator
UFSC/CTC/DAS

Prof. Fabio Luis Baldissera, PhD.
Board President
UFSC/CTC/DAS

ABSTRACT

Well control optimization is a crucial task in the development and operation of petroleum fields. It consists in adjusting the well controls such that the reservoir performance is optimized over its producing time. Optimization of the well controls typically require the simulation of fluid flow in the reservoir with the aim to evaluate the reservoir response for different well controls over time. Although there are several simulators available in the market, only a few provide gradient information with respect to the well controls. Besides, even when the gradients are available, they may degrade due to inaccuracies arising from control variable switching in the simulator or discontinuities in the feasible space. Derivative-free optimization is a common alternative to circumvent such issues.

Derivative-free optimization can be broadly split into direct-search methods and model-building approaches. The first optimize the function of interest without building a surrogate model (trust-region, for example) to guide the search. For this reason, the second approach typically requires fewer simulations in the optimization. This is particularly attractive for field development optimization problems such as well control optimization because the simulations involved in such problems are computationally costly.

Due to the presence of geological uncertainty, the standard well control optimization problem needs to be extended to consider the parametric uncertainty in the reservoir model. This is typically done by creating a discrete set of equally probable reservoir models to represent the geological uncertainty. Then a robust well control optimization problem can be cast to improve the reservoir performance given an ensemble of reservoir models. The robust optimization problem has the same challenges of the standard well control problem with the extra burden of simulating the whole ensemble of models in each step of the optimization procedure.

We propose a method called ensemble trust-region optimization, which is a model-building derivative-free optimization method that uses a trust-region to guide the optimization of the average performance of an ensemble of reservoir models. The proposed method is assessed in both a set of analytical functions and an ensemble of synthetic reservoir models.

Keywords: Well control optimization. Geological uncertainties. Trust-Region Optimization.

RESUMO

A otimização de controle de poços é uma tarefa crucial no desenvolvimento e operação de campos de petróleo. Esta consiste em ajustar os controles dos poços para que o desempenho do reservatório seja otimizado ao longo do tempo de produção. A otimização de controles de poços normalmente requer a simulação do fluxo de fluidos no reservatório com o objetivo de avaliar a resposta do reservatório para diferentes controles ao longo do tempo. Embora existam vários simuladores disponíveis no mercado, apenas alguns fornecem informações do gradiente com relação ao controle. Além disso, mesmo quando os gradientes estão disponíveis, eles podem não ser confiáveis devido a imprecisões decorrentes da troca de variável de controle no simulador ou descontinuidades no espaço de soluções. A otimização sem derivada é uma alternativa comum para contornar esses problemas.

A otimização sem derivadas pode ser amplamente dividida em métodos de pesquisa direta e métodos que se baseiam na construção de modelos. Os primeiros métodos otimizam a função de interesse sem criar um modelo substituto (região confiável, por exemplo) para orientar a pesquisa. Por esse motivo, a segunda abordagem normalmente requer menos simulações durante a otimização. Isso é particularmente atraente para problemas de otimização de desenvolvimento de campo, como otimização de controle de poços, porque as simulações envolvidas em tais problemas são computacionalmente custosas.

Devido à presença de incerteza geológica, o problema padrão de otimização de controle de poços precisa ser estendido para considerar a incerteza paramétrica no modelo do reservatório. Isso geralmente é feito criando um conjunto discreto de modelos igualmente prováveis de reservatórios para representar a incerteza geológica. Em seguida, um problema robusto de otimização de controle de poço pode ser lançado para melhorar o desempenho do reservatório, dado um conjunto de modelos de reservatório. O problema de otimização robusto tem os mesmos desafios do problema de controle de poços padrão com a carga extra de simular todo o conjunto de modelos em cada etapa da otimização.

Propomos um método chamado *Ensemble Trust-Region Optimization*, que é um método de otimização sem derivada do tipo construção de modelo que usa uma região de confiança para orientar a otimização da média de desempenho de um conjunto de modelos de reservatório. O método proposto é avaliado tanto em um conjunto de funções analíticas quanto em um conjunto de modelos de reservatórios sintéticos.

Palavras-chave: Otimização de controle de poços. Incertezas geológicas. Otimização Trust-Region.

LIST OF FIGURES

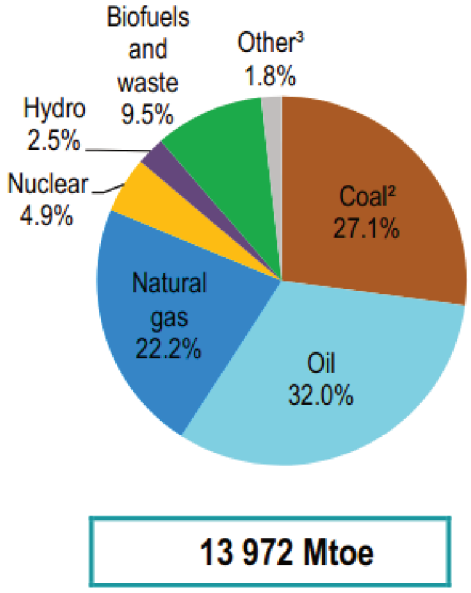
LIST OF TABLES

CONTENTS

1 INTRODUCTION

Since its discovery, petroleum has being a key energy source in the modern society. Its contribution had a great impact in the economic and social development. The oil industry generates employment in diverse industry sectors such as energy and technology. As one of the primary energy sources, oil consumption has increased considerably during the past decades and its demand is still growing. It is the energy source that dominated the 20^{th} century and will continue to domain the most part of 21^{th}. The most versatile and political of energy sources. It makes countries go to war (AMERICA IPAA, 2002).

Figure 1 – World total primary energy supply (TPES) by source.

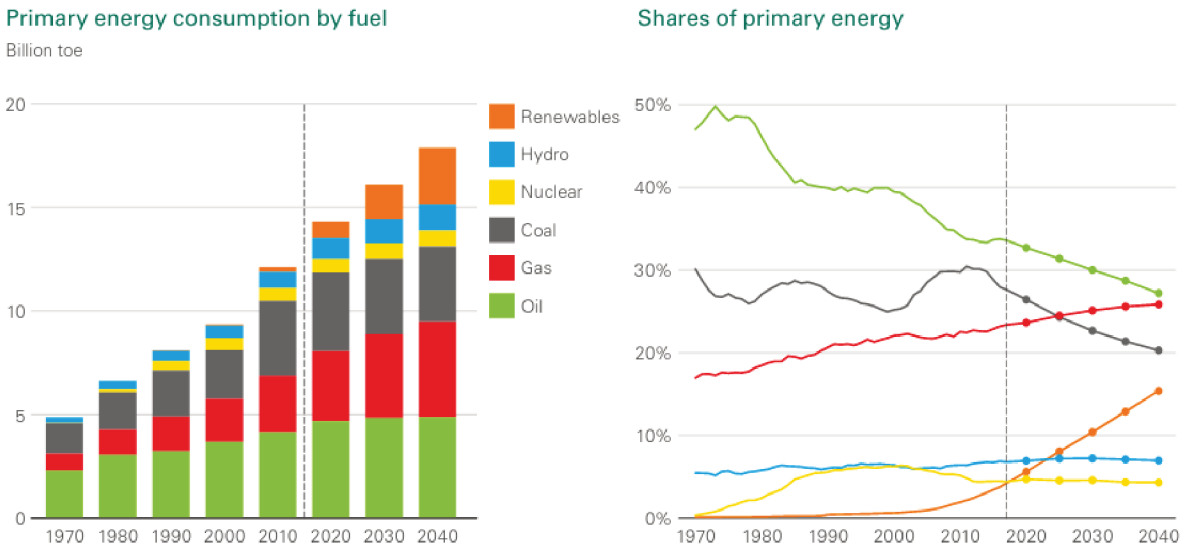


[3]**Other:** Includes geothermal, solar, wind, tide/wave/ocean, heat and other sources.

Nevertheless, there are a lack of electricity access for 850 million people around the world (IEA, 2019) and, with the population increase, the number of people in this situation tends to grow up even more.

Aim to mitigate the energy deficit in the world, global investments are being designated to the improvement of renewable energies. Although this government effort, as it can be seen in Figure 2, until 2019 the renewable energy consumption was around 4% and the estimated growth is of 15% by 2040. Also, Figure 1 shows that Oil and Natural gas represents more than 54% of the current worldwide primary energy supply.

Figure 2 – Forecast of energy consumption



Font: Energy Outlook 2019 (BP, 2019)

The trend points out that renewable energy is the future of energy supply, but fossil fuels still contribute, and will contribute for the next decades, with the majority part of the total energy consumed by society. Also, the petroleum sector is the largest industry in the world in terms of monetary value. Not only is fossil fuels the world's most important energy source, but oil is also the feedstock of a wide range of chemical products that are critical in today's society (MIDTTUN, 2015). Although there exists reserves of unconventional hydrocarbons the mechanisms to recovery such resources are nor economically attractive neither environmentally sensitive (ANN et al., 2014).

The union of these set of factors resulted in a considerable sum of investments in the development of reservoir management strategies. Optimization methodologies have gained space in petroleum engineering activities over the past years, particularly in the called reservoir control optimization, which combines classical geosciences phenomena with cybernetics engineering.

Mathematical optimization is showing its potential in reservoir management by improving the production and collaborating with important decisions, such as number, type, control and placement of wells. However, as the optimization decisions are based on geological models, uncertainty has been the biggest concern in oil and gas production. The model uncertainty might lead to constraint violations. As the objective is generally to maximize the production, the system could select a set of controls that extrapolates the operational capacity, which may lead to system failure. Also, the increase of health, safety and environment requirements has increased the attention to uncertainties in process models. That is why handling uncertainty is crucial and needs to be performed effectively.

Indeed, handling uncertainty is becoming more relevant in most industrial sectors. But handling uncertainty in reservoir management is a complex task. Geological

uncertainty is quantified by generating a number of realizations for a reservoir model, taking into account a discrete range for the uncertain parameters.

1.1 OBJECTIVES

The main objective of this work is to contribute to the field of production optimization by proposing a derivative-free method that is capable of handling geological uncertainties.

The specific goals are divided as follows:

1. Generate a set of realizations that simulates a real scenario of reservoir under geological uncertainties.
2. Extend the trust-region method implemented in the open-source project to deal with geological uncertainties.
3. Demonstrate the concept in a set of analytical realizations of well known functions.
4. Apply the proposed framework in a synthetic case with the set of geological realizations produced on Item 1.

1.2 DOCUMENT STRUCTURE

The first two Chapters 2 and 3 provide an overview of field development optimization and derivative-free methods, which are the background for this work.

Chapter 4 contextualizes and explains the proposed method, i.e. ensemble trust-region optimization. Further, it introduces the concept of geological uncertainty and explains how it can be accounted for in the format of an ensemble of realizations.

A validation of the proposed methodology is presented in Chapter 5, where the method is applied to a set of realizations which are generated empirically from two well known analytical functions.

In Chapter 6 the methodology is applied to the production optimization under geological uncertainty of a synthetic reservoir.

The conclusions of the work and future goals are provided in Chapter 7.

2 FIELD DEVELOPMENT OPTIMIZATION

Petroleum field development encompasses operations involving a broad range of engineering disciplines. Within various field development aspects such as drilling, facility operation and/or reservoir production, a whole range of decisions can be supported by solving associated optimization problems(BAUMANN et al., 2020). The initial phase in the field development is the discovery of profitable petroleum reserves, which can be composed of a single or a portfolio of reservoirs. Optimization routines start to play an important role since the discovery phase, where decisions must be made regarding the locations of the producers and injectors. The problem of deciding the trajectories of the wells is referred to as well placement optimization. In the extraction phase, the injectors and producers wells need to be controlled so that the reservoir can produce their maximal capacity, injecting the wrong quantity of water can implicate negatively in the amount of oil extracted. The latter problem is the focus of this work and it is called *well control optimization*.

Well placement optimization is not that easy as it appears, i.e. finding the optimal locations for producer and injector wells requires simulation studies which need to take into account several aspects such as the type, number and operations settings of wells. The main challenge of well placement problems is to decide the best well trajectories into the reservoir, such that a stable water front is achieved and the NPV is maximized over the reservoir producing time. So, it is also important to know reservoir proprieties such as faults, geological proprieties, among others.

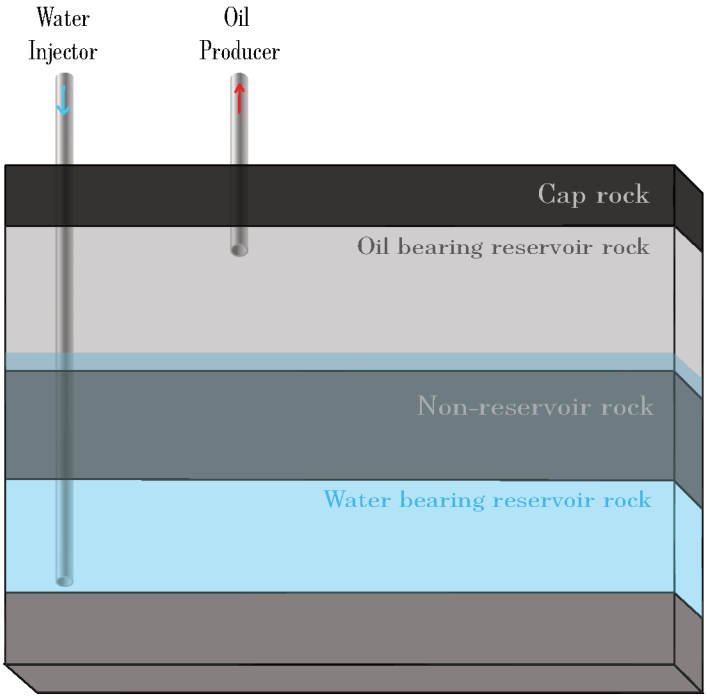
In this work, the derivative-free optimization (DFO) method will be applied to a well control problem to optimize the oil production. The next section of Chapter 2 explains the well control optimization problem and presents the optimization method applied for its solution.

2.1 WELL CONTROL

Well Control as a term and technical word suggests controlling the bottom hole formation pressure being penetrated by the well (GRACE, 1977).

Oil and gas resources are generally contained in sandstones or limestones beneath the earth surface, typically at a depth between 1-5 km (BROUWER, 2004). The cross-section of an oil reservoir, presented in Figure 3, is composed by non-reservoir rocks that has high porosity and permeability. The cap rock, impermeable to fluids, is a natural cover to the reservoir. Water bearing zones as non reservoir rocks can also bound the reservoir. Through the oil bearing rocks the well is drilled for oil production which enables the flow of oil fluid to the surface.

Figure 3 – Reservoir vertical cross-section



Intuitively, well control optimization consists in controlling the bottom hole pressure (BHP) of injectors and/or producers to extract the optimal amount of oil from the reservoir. There are some reservoir proprieties that need to be considered, such as rock and fluid proprieties. The BHP of the wells are the typical control variables of the optimization problem, which seeks for improved reservoir performance by adjusting the well controls over the reservoir producing time.

Figure 4 shows an illustration of a well control problem. It represents a reservoir that has two injectors wells and one producer vertical well. There are three types of conventional wells: vertical, deviated, horizontal and multilateral well, which are types presented in Figure 5 that were obtained from (NEFT, n.d.). Observe that the water is being injected by the injectors, while the producer is lifting oil out of the reservoir. Physically the water injected by the injectors keeps the pressure inside the reservoir. Because of the difference between the density of the elements (oil and water), the oil is pushed toward the producer, given that it has a lower pressure which is controlled by the bottom hole pressure (BHP).

Figure 4 – Well control problem illustration

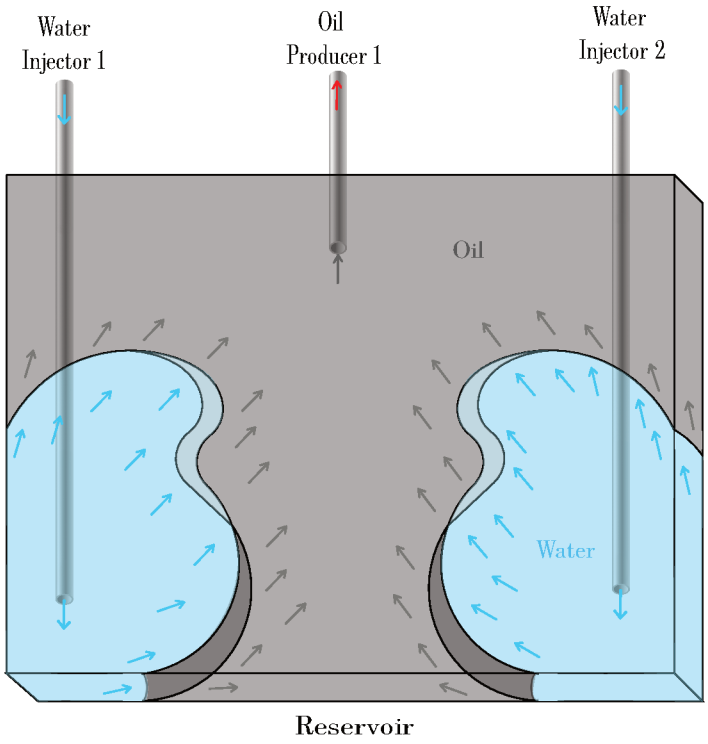
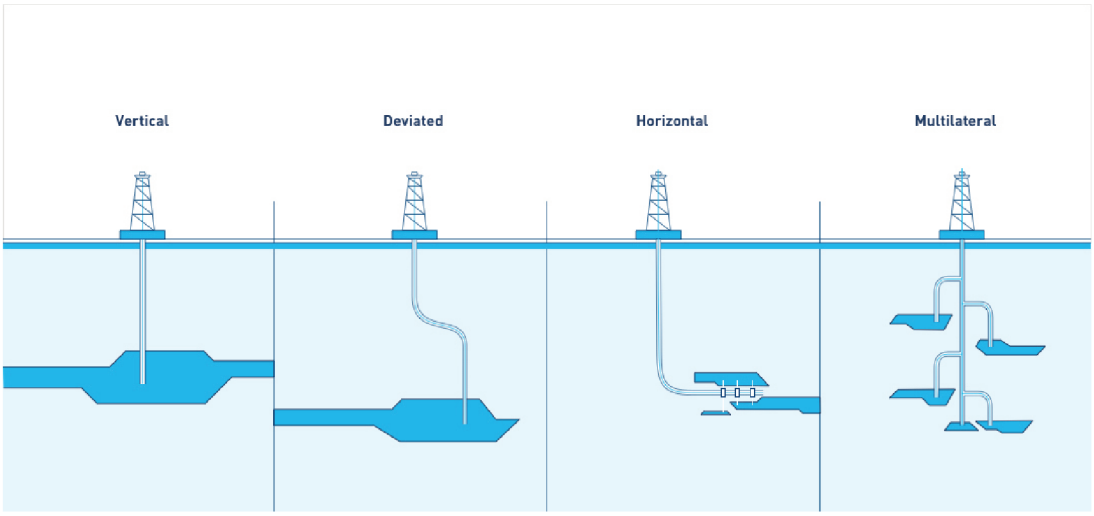


Figure 5 – Different type of wells



Font: (NEFT, n.d.)

The standard objective function employed in field development optimization problems is typically maximize the Net Present Value (NPV), but the cumulative oil production (COP) is a common alternative. It contains parameters such as the cost related to water injection and production and the oil price, which translates the economic gain obtained of a given reservoir control strategy. For two-phase flow of oil and water, the

NPV is defined as follows:

$$NPV(q_u) = \sum_{t=1}^{T} \frac{\Delta t}{(1+d)^{t.\Delta t/\tau}} \left[r_{op}^t \cdot q_{op}^t - r_{wp}^t \cdot q_{wp}^t - r_{wi}^t \cdot q_{wi}^t \right] \quad (1)$$

$$q_u = (q_{op}^t, q_{wp}^t, q_{wi}^t) \quad \forall t = 1, \ldots, T. \quad (2)$$

where q_u is a vector containing the total oil production rate q_{op}^t, the total water production rate q_{wp}^t, and the total water injection rate q_{wi}^t for all time steps $t \in 1, \ldots, T$. The parameters r_{op}^t, r_{wp}^t and r_{wi}^t represent the revenue obtained with the selling of the oil, the cost for water treatment, and the cost of water injection, respectively. The step size is defined as Δt. τ is a normalization term, commonly taken as the number of days in a year, and d is a discount factor (SILVA et al., 2020).

2.2 WHY DERIVATIVE-FREE IN WELL CONTROL OPTIMIZATION?

As stressed in the previous section, well control is a simulated process. This section focus in describing the reasons behind choosing a derivative-free approach. Due to the wide usage of simulations the need of good derivative-free methods is still present and increasing (ANDERSEN, 2018).

Typically, well control problems are smooth and continuous, i.e derivatives are available. However, a process simulation is required to compute the cost function associated to the controls. There are several simulators available for fluid flow simulation in petroleum reservoir, but only few of them provide the sensitivities related to the control variables, also known as the adjoint gradients. Solving a system of adjoint gradients is a common alternative to obtain the gradients (JANSEN, 2011; KOUROUNIS et al., 2014). When the gradients are available, (WANG et al., 2010; JANSEN et al., 2009; VAN ESSEN et al., 2011; SUWARTADI et al., 2011; CAPOLEI et al., 2013) show that good result can be obtained in well control optimization by using adjoint-gradients. However, reliable gradient information are typically unavailable in industrial large-scale simulation systems due to the complexity and computational cost to implement such derivatives.

When the adjoint gradients are not available, a solution would be to estimate the gradient by finite differences at each step. Yet, this alternative has its limitations. The computational cost of this approach is elevated, which makes it infeasible for large-scale problems.

Another issue related to gradient-based methods is that the derivatives need to be calculated with respect to the control variables, for both the objective and the constraints (MIDTTUN, 2015). If the feasible bounds for the control variables are violated during the simulation, an event called *control switching* might occur. The switching in the control variables could degrade the accuracy of the gradients significantly, causing complications for gradient-based methods.

To circumvent the above mentioned issues, derivative-free methods can be utilized to solve well control optimization problems. Since the simulations of fluid flow in reservoir models are costly, the best suited method for such problems would be an approach which requires fewer simulations in the optimization. The next chapter offers an overview of derivative-free optimization methods in order to support our choice regarding the optimization method.

3 DERIVATIVE-FREE OPTIMIZATION

The main idea of derivative-free methods is to optimize functions without the use of derivatives. Traditionally important information can be obtained from gradients, such as the function curvature and descent direction, but in many cases the gradient information might be unavailable or unreliable due to noise or discontinuities. The lack of meaningful gradient information requires optimization without derivatives, which is usually challenging in computational sciences and engineering applications. Performance issues also come about as derivative-based methods tend to be far more efficient than derivative-free optimization methods (CONN et al., 2009).

Herein derivative-free optimization concerns the problem of minimizing a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ over a domain $x \in \mathcal{X}$. The function f is available as a black-box, whose derivatives are either unavailable or unreliable (CONN et al., 2009; RIOS; SAHINIDIS, 2013). Such problems arise, for instance when the evaluation of f is subject to noise or too costly for the application of finite difference methods.

Generally derivative-free methods can be classified as *direct-search* and *model-based* methods. Direct-search methods are based on successive evaluations of the cost function f based on some predefined sampling patterns. On the other hand, model-based methods construct a surrogate for f around the current iterate, which is then combined with standard algorithms to define the next iterate. The models are updated iteratively to capture the behavior of f around the incumbent solution.

To provide an overview of the algorithm used and to contextualize the class of methods in which it is included, this Chapter brings an explanation of derivative-free methods.

3.1 DIRECT-SEARCH METHODS

Direct search methods, such as Hooke and Jeeves (HOOKE; JEEVES, 1961) and Nelder-Mead (NELDER; MEAD, 1965) work based only on function evaluation, without attempting to build any model. They describe direct-search as the sequential examination of trial solutions generated by a certain strategy.

While in the 1960s these methods were heuristic, more recent variations have proven global convergence. Direct-search methods can be divided into two main classes, simplicial and directional (CONN et al., 2009).

- **Simplicial method** covers direct-search based on simplices and operations over simplices, like reflections, expansions or contractions, for example Nelder-Mead and the simplex method.
- **Directional method** addresses direct-search where sampling is guided by sets of directions with appropriate features, like coordinate-search methods.

The focus of this work is on model-based methods, more precisely on the Trust-Region method described at the end of this chapter. For this reason the description of the direct-search methods will be further elaborated here.

3.1.1 Simplicial Methods

The description of simplicial methods will be focused on one of the most popular derivative-free methods, the Nelder-Mead algorithm. The simplex method of Nelder and Mead utilizes a set of points to define a simplex at each iteration. This simplex is then used to determine the next incumbent solution. The definition of simplex will be given later. Various operations are performed around the centroid of the simplex in order to produce an improved solution, with which the simplex is updated.

Simplicial methods which use the concept of Nelder and Mead are the Generalized Pattern Search (GPS) (TORCZON, 1997), the Generating Set Search (GSS) (KOLDA et al., 2003), and the Mesh Adaptive Direct Search (MADS) (AUDET; DENNIS JR, 2006). At each iteration, these methods sample the objective in a finite number of points around the current approximate solution. If the current iterate is still not optimal, one of the search directions is guaranteed to be a descent direction. So descent may be found for a sufficiently small step length.

Before present the Nelder-Mead algorithm, some definitions are needed. At each iteration k, the algorithm has a set of $(n+1)$ points in $\mathbb{R}^n$ which define a simplex, a set $S_k = \{x_k^1, x_k^2, ..., x_k^{n+1}\}$ that are affinely independent, meaning that

$$x_k^2 - x_k^1, ..., x_k^{n+1} - x_k^1$$

is a set of linearly independent vectors in $\mathbb{R}^n$. The formal definition is presented as follows.

The dimension of an affine set is the dimension of the linear subspace parallel to it. Then, in $\mathbb{R}^n$ we cannot have an affinely independent set with more than $(n+1)$ points.

Definition. *A set of $n+1$ points $X = \{x^0, x^1, ..., x^n\}$ is said to be a affinely independent if its affine hull aff($x^0, x^1, ..., x^n$) has dimension n.*

With the definition of affinely independence we can also define the convex hull. The convex hull of a given $S \subset \mathbb{R}^n$ is always uniquely defined and consists of all linear combinations of elements of S whose scalars are non-negative and sum up to one (i.e., such combinations are known as convex combinations).

Definition. *Given an affinely independent set of points $\mathbb{X} = \{x^0, x^1, ..., x^n\}$, its convex hull is called a simplex of dimension n.*

Geometrically, a bi-dimensional simplex is a triangle. The vertices of this triangle are elements of $\mathbb{X}$.

Algorithm 1 Nelder-Mead in three fundamentals steps for a function in $\mathbb{R}^2$.

Initialization: Choose an initial simplex called S_0.

for *k = 0,1,2,...* **do**

1. **Ordering:** Determine the indices 1, 2, 3, based on function values of the worst, second worst and the best vertex, respectively, in the current working simplex S. In other words, $S_k = \{S_k^1, S_k^2, S_k^3\}$, with $n = 2$, S_k^1 is the vertex with the best function value, S_k^2 is the second worst and S_k^3 is the worst vertex (highest function value).

2. **Centroid:** Calculate the centroid c of the best segment, $\overline{S_k^1 S_k^2}$.

$$c = \frac{1}{n}\sum_{i=1}^{n} S_k^i$$

3. **New simplex:** Compute a new simplex from the current one. Try to apply the Reflect, Expansion or Contraction operation with respect to the best segment, to replace only the worst vertex S_k^3. If one of these three operations yields a point with an improved objective value compared to the best vertex objective of the current simplex, then that point is accepted, the worst vertex is spared and the iteration is deemed successful. Otherwise, when all operations fail, a shrink operation is performed whereby the best vertex is kept and the other two vertices are produced by the operation.

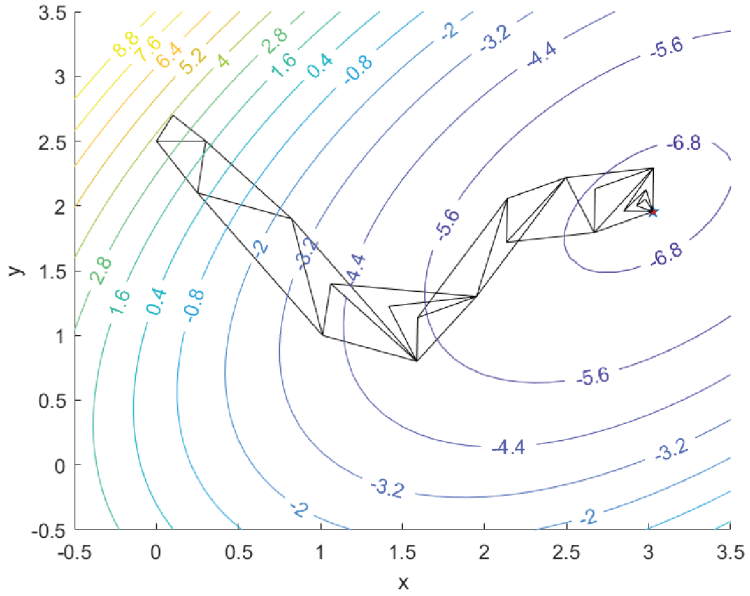
end

Algorithm 1 brings the general idea of the Nelder-Mead method for a function in $\mathbb{R}^2$. First, the current simplex is ordered based on each vertex function value. Second, the centroid c is calculated with respect to the best segment, that is the segment between x_n and x_{n+1}. Then, a new simplex is computed according with the operation that is successful based on c and vertex function values. The method has five different operations: Reflect, Expand, Outside contraction, Inside contraction and Shrink, which are applied to generate the new simplex.

To illustrate the Nelder-Mead algorithm, the Figure 6 applies the method to minimize the function presented in Eq. (3). The initial simplex is the triangle with the following vertices (0.0, 2.5),(0.1, 2.7),(0.3, 2.5). The solution value is indicated by the red star at (3.0, 2.0) with the function value of –7.

$$f(x_1, x_2) = x_1^2 - 4x_1 + x_2^2 - x_2 - x_1 x_2 \tag{3}$$

Figure 6 – Nelder-Mead illustrative example



Observe that the Nelder-Mead method performs operations over the current simplex trying to reach the best solution point. The stop criterion is associated with the triangle size. When the simplex is enough small the algorithm stops and considers the incumbent solution as the optimal solution.

3.1.2 Directional Methods

Consider the problem of minimizing a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ over a domain $x \in \mathbb{X}$. Directional methods are derivative-free methods that sample the objective function f at a finite number of times in each iteration. Solely based on this set of function values the algorithm decides which step to perform without any explicit or implicit derivative approximation or model building (CONN et al., 2009).

In this section we address one of the simplest directional direct-search methods, namely the coordinate or compass search. Of key importance to understand this method are the concepts of positive spanning sets and positive bases, which are explained in what follows.

3.1.2.1 Positive spanning sets and positive bases

In directional methods the existence of a positive spanning set $D \subset \mathbb{R}^n$ guarantees that, given any nonzero vector v in $\mathbb{R}^n$, there exists at least one vector d in D such that v and d form an acute angle. In optimization, the vector d indicates a descent direction. Formally, a descent direction for a function f at point x means that there exists an $\overline{\alpha} > 0$ such that $f(x + \alpha d) < f(x)$ for all $\alpha \in (0, \overline{\alpha}]$. Then, there exists a point such that the function decreases in the descent direction.

Definition 3.1. A positive spanning in $\mathbb{R}^n$ is a set of vectors whose positive span is $\mathbb{R}^n$.

The *positive span* of a set of non-negative vectors $[v_1, ..., v_m]$ in $\mathbb{R}^n$ is the convex cone[1]:

$$\{v \in \mathbb{R}^n \ : \ v = \alpha_1 v_1 + \cdots + \alpha_m v_m, \ \alpha_i \geq 0, i = 1, ..., m\}.$$

Definition 3.2. A positive spanning set $\mathbb{D}$ in $\mathbb{R}^n$ is a set of vectors whose positive span is $\mathbb{R}^n$. The set $[v_1, ..., v_m]$ is said to be positively dependent if one of the vectors is a positive combination of the others; otherwise, the set is positively independent. A positive basis in $\mathbb{R}^n$ is a positively independent set whose positive span is $\mathbb{R}^n$.

Equivalently, a positive basis for $\mathbb{R}^n$ can be defined as a set of nonzero vectors of $\mathbb{R}^n$ whose positive combinations span $\mathbb{R}^n$, but for which no proper set exhibits the same property. The following theorem indicates that a positive spanning set contains at least $n+1$ vectors in $\mathbb{R}^n$.

Theorem 3.3. *If $[v_1, ..., v_m]$ spans $\mathbb{R}^n$ positively, then it contains a subset with $m-1$ elements that spans $\mathbb{R}^n$.*

The proof of this theorem can be found in Conn et al.(CONN et al., 2009). From this theorem we can also show that a positive basis can not contain more than $2n$ elements. Then, the terms *minimal* and *maximal* positive bases are referred, respectively, to positive bases with $n+1$ and $2n$ elements.

The positive basis that we are interested, used in the Compass Search method, is formed by vectors of the canonical basis and their negative counterparts. It is also the simplest maximal positive basis in $\mathbb{R}^2$, defined by the columns of the following matrix D.

$$D = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{4}$$

Basically, a positive basis and positive spanning guarantee the existence of at least one vector $d \in D$ that forms an acute angle with a nonzero vector $v \in \mathbb{R}^n$. Suppose that the nonzero vector v is the negative gradient of a continuously differentiable function f, then any vector d that forms an acute angle with v is a descent direction for f. This guarantees that f decreases in direction d. In this context, D is a positive basis and d a positive spanning.

3.1.2.2 Directional direct-search method

The concepts presented above are fundamental to understand the essence of the directional direct-search method and mainly to perceive the difference between the

[1] A nonempty subset X of $\mathbb{R}^n$ is called a convex cone if for any elements x, y of X and a non-negative number a, ax and $x+y$ are contained in X(JAPAN, 1993).

simple directional method, Compass search, and the OrthoMads method (explained in the next section).

The main steps of the directional direct-search method are described in Algorithm 2. The poll step calculates the set of trial points P_k based on the positive basis D and evaluate the function f for each point of P_k, then, if there exists a poll point $(x_k + \alpha_k d)$ that makes the function value decrease in relation to the current point x_k, it updates the current point (x_{k+1}). If the poll step failed, the current point is kept at the same position. The purpose of the mesh parameter update step is to decrease the step size parameter α_k in order to ensure the convergence of the objective function to a local minimum. The modification of α_k depends on the previous poll step status.

A natural stopping criterion in directional direct search is to terminate the execution when $\alpha_k < \alpha_{tol}$, for a given $\alpha_{tol} > 0$ (For example, $\alpha_{tol} = 10^{-3}$).

Algorithm 2 Simplified description of the directional direct-search method

Initialization: Choose x_0, $\alpha_0 > 0$, $0 < \beta_1 \leq \beta_2 < 1$, and $\gamma \geq 1$. Let D be set of positive spanning bases.

for *k = 0,1,2,...* **do**

1. **Poll step:** Choose a positive basis D_k from the set D. Set $P_k = \{x_k + \alpha_k d : d \in D_k\}$. Start evaluations of f at the poll points[2], sample $f(x)$ for all $x \in P_k$. The iteration finishes and the poll step is declare successful if a poll point $x_k + \alpha_k d$ is found such that $f(x_k + \alpha_k d) < f(x_k)$, then $x_{k+1} = x_k + \alpha_k d$. Otherwise, declare the poll step unsuccessful and set $x_{k+1} = x_k$.
2. **Mesh parameter update:** If the iteration was successful, then maintain or increase the step size parameter $\alpha_{k+1} \in [\alpha_k, \gamma\alpha_k]$. Otherwise, the mesh parameter is decreased, $\alpha_{k+1} \in [\beta_1\alpha_k, \beta_2\alpha_k]$.

end

3.1.2.3 Compass Search Method

The Compass Search method, also known as Coordinate Search, is easier to implement and understand than the Nelder-Mead method. It uses the same procedure showed in the directional direct-search method with the positive basis being the columns of the matrix D, presented in (4). The method is detailed in Algorithm 3.

The positive basis D allows a search in four different points surrounding the current point. An illustrative example is presented in Figure 7, where the blue point (central) is the current point, the black points are the trial points, and alpha (α_k) is the variable that defines the distance of the current point to the trials points (step size). The stopping criterion is based on the step size, i.e. the algorithm stops when the variable α_k is less than a constant tolerance (α_{tol}), typically set to $\alpha_{tol} = 10^{-5}$.

Algorithm 3 Compass search algorithm

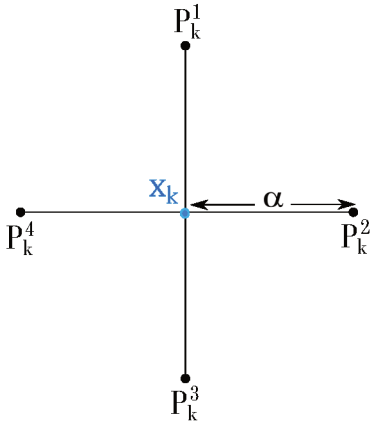
Initialization: Choose x_0, $\alpha_0 > 0$, $0 < \beta_1 \leq \beta_2 < 1$, $\gamma \geq 1$ and $\alpha_{tol} > 0$. Let D be a set of positive spanning bases.

for *k = 0,1,2,...* **do**

1. **Stop criterion:** If $\alpha_k < \alpha_{tol}$, then terminate the execution of the algorithm. Otherwise, evaluate the Poll step.
2. **Poll step:** Set $P_k = \{x_k + \alpha_k d : d \in D_k\}$. Evaluate f at each element of P_k, $f(x)$, $\forall x \in P_k$. If for some $x_k + \alpha_k d$ the function value is better than the value of $f(x_k)$ in the current point, mathematically, $f(x_k + \alpha_k d) < f(x_k)$ for some $x_k + \alpha_k d \in P_k$, then update the current point, $x_{k+1} = x_k + \alpha_k d$. Go to step 3. Otherwise, declare the poll step unsuccessful and maintain $x_{k+1} = x_k$.
3. **Mesh parameter update:** If the iteration was successful, then the step size parameter of the mesh parameter is maintained, namely $\alpha_{k+1} = \alpha_k$. Otherwise, the value of the mesh size parameter is decreased, namely $\alpha_{k+1} = \frac{\alpha_k}{2}$.

end

Figure 7 – Compass step example.



The Figure 8 shows the evaluation of the Compass search algorithm to the illustrative example. The problem is to minimize the function indicated in Eq. (3), same function used to exemplify the Nelder-Mead algorithm in the previous section. The initial parameters were selected as follows, initial point as $(0.0, 2.5)$, matrix D as the canonical one and the initial step length as 0.5. The optimal solution is –7, indicated by the red star at $(3.0, 2.0)$. The algorithm was capable to find the optimum solution in few steps.

Beyond the CS method the class of directional direct-search methods has other well known optimization algorithms. It is not the aim of this section to discuss all this algorithms, the Compass Search was an example because is a easy method to understand. The directional methods are basically differentiated by the mode to search points, as saw CS search using the matrix D to search in canonical directions. Generalized Pattern Search (GPS), introduced by (AUDET; DENNIS, 2000), does not have a fixed

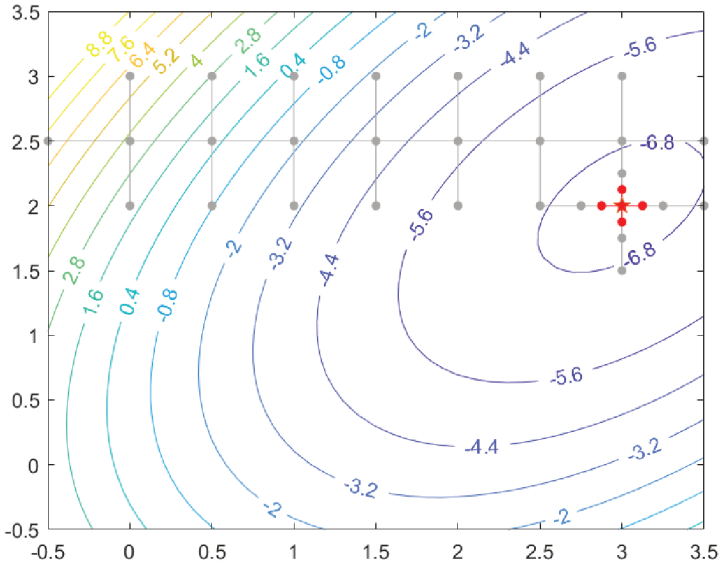


Figure 8 – Compass search illustrative example

matrix D instead of it, a set of positive bases is used. (AUDET; DENNIS JR, 2006) proposes the Mesh Adaptive Direct Search (MADS), capable to converge in nonsmooth problems, once the search for points is not restricted to a finite number of directions, as D, MADS does a local exploration in the space of variables.

3.2 MODEL-BASED METHODS

Model-based methods firstly sample the function around the incumbent solution to build an initial surrogate model of the function being optimized. The optimization search is then conducted such that candidate solutions are then calculated with the aid of this surrogate model. The evaluated points are accepted when there is an improvement in the objective, and rejected otherwise. The surrogate is then updated by a model maintenance procedure, and the optimization continues.

Trust-region methods are model-based methods which rely on polynomial interpolation surrogates valid within the neighborhood of the current iterate, the so-called trust-region. As models are used, these methods tend to be more efficient than direct search methods in finding descent. But this comes at the expense of model maintenance, a procedure that is costly. The size and position of the trust-region are adjusted depending on whether or not the surrogate solution is accepted, and the ratio between the predicted and actual function value decrease.

3.2.1 Trust-Region Algorithm

In this subsection we present a trust-region method for unconstrained derivative-free optimization, which uses a quadratic polynomial interpolation to build a model into

the trust-region, following Conn et al. (CONN et al., 2009) and Giuliani C. (GIULIANI, 2019). The method uses polynomial interpolation, but it could be done with polynomial regression or any other approximation technique. The algorithm builds a model m that approximates the function in a limited region (the trust-region) sampling the objective function f around the incumbent solution x_k. This model m is simply a known mathematical model, which can then be optimized using traditional algorithms to generate a new candidate solution x_{k+1}.

Before presenting the trust-region algorithm, let us define the concept of Fully Linear.

Definition 3.4. A model m is said to be Fully Linear (FL) in the trust-region $\mathbb{B}(x, \Delta)$ if there exist $\neg_{ef}$, $\neg_{eg} > 0$ such that, for all $y \in \mathbb{B}(x, \Delta)$:

$$\|f(y) - m(y)\| \leq \neg_{ef}\Delta^2$$
$$\|\nabla f(y) - \nabla m(y)\| \leq \neg_{eg}\Delta$$

where $\mathbb{B}(x, \Delta)$ is an ball of radius Δ with center at x. An explanation about the values $\neg_{ef}$ and $\neg_{eg}$ can be found in (CONN et al., 2009) page 40. Now, consider the nonlinear optimization problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

In the first iteration the model contains only the initial point x_1, and the trust-region will be a ball surrounding it. More generally, in each iteration k, the trust-region is a ball of radius Δ_k around the iterate (incumbent solution) x_k as follows

$$\mathbb{B}(x_k, \Delta_k) = \{y : \|y - x_k\| \leq \Delta_k\}$$

with the the infinity norm $\|.\|_\infty$ being the usual choice. Once the region is defined, the function f is sampled around the incumbent solution x_1 to build the model m_1, a quadratic polynomial in our case. To ensure that the model has good properties, a criticality test is performed. After passing the criticality test, a trial point x_k^+ is obtained by solving the trust-region *subproblem*, which is a quadratic optimization problem with the objective function defined by the polynomial model m_k and the feasible space being defined inside the bounded region $\mathbb{B}(x_k, \Delta_k)$, as follows:

$$x_k^+ = \underset{x}{\operatorname{argmin}}\ m_k(x)$$
$$\text{s.t. } \|x - x_k\| \leq \Delta_k$$

After the calculation of the trial point x_k^+, the algorithm decides on the acceptance of the trial point based on the function values of the incumbent and the trial points. After calculating the trial point x_k^+ is in the k^{th} iteration of the algorithm, the corresponding

function value $f(x_k^+)$ is calculated and compared to the function value of the current point $f(x_k)$. If there is a descent, the trial point x_k^+ is accepted as the next iterate x_{k+1}, otherwise the trial point x_k^+ is used to improve the model m_{k+1} or cached by the algorithm if it does not improve the model.

In general, the objective function f is evaluated at the trial point x_k^+ and the descent predicted $pred_k$ is compared to the actual descent $ared_k$. The trial point is accepted or not based on the following measure of descent:

$$\rho = \frac{ared_k}{pred_k} = \frac{f(x_k) - f(x_k^+)}{m(x_k) - m(x_k^+)}$$

This equation measures the decrease obtained at iteration k and how well the model agrees with the objective function. If this value is sufficiently high ($\rho > \eta_1 > 0$), this means that the decrease was good and the model is close enough to the function, then the trial point is accepted as the new iterate x_{k+1} and the trust-region center now may be moved to this new point. If the model is already Fully Linear (FL) inside the region $\mathbb{B}(x_k, \Delta_k)$ and has a smaller decrease ($\rho > \eta_0 \geq 0$) the point will also be accepted. Otherwise, the trial point is rejected and the center is maintained at the same point, i.e. $x_{k+1} = x_k$.

If $\rho \leq \eta_1$ and the model is not Fully Linear, the algorithm will be perform one or more improvement steps until the model becomes FL. Further, the value of ρ is used to change the radius of the trust-region. If $\rho > \eta_1$, the radius may be increased for a factor $\gamma_{incr} > 1$, then $\Delta_{k+1} = \gamma_{incr}\Delta_k$. If $\rho \leq \eta_1$ and the model was FL, the bounds of FL definition must be overly high, so the radius has to be decreased for a factor $\gamma_{decr} \in (0, 1)$, then $\Delta_{k+1} = \gamma_{decr}\Delta_k$. When $\rho \leq \eta_1$ and the model was not FL, the reduction of the radius may not improve the model's accuracy, so the radius is maintained the same, i.e. $\Delta_{k+1} = \Delta_k$.

The algorithm proceeds through the iterations by maintaining the model updated, adjusting the radius size when needed, and thereby obtaining candidate solutions until the stopping criterion is reached.

3.2.1.1 Illustrative Trust-region algorithm

The illustrative example is showed in the sub-figures of Figure 9. Each sub-figure represents an iteration of the algorithm. The code used to reproduce the Trust-Region algorithm was the described in (GIULIANI, 2019). The aim is to minimize the function indicated in Eq. (3). The red star, located at $(3.0, 2.0)$ is the optimal solution for the problem. The radius tolerance is $1e^{-5}$ and the initial point is $(0.0, 2.5)$.

In the first iteration, Figure 9a, the red circle indicates the trust-region center, that means the current best solution. The area in gray represents the trust-region, were the sub-problem will be procedure. The other blue circles are points used to build

the polynomial model. The trial point are indicated by the orange circle. The red star indicates the optimum solution.

Observe that the trial point was approved from the first to second iteration. Also, the trust-region radius increased and the last center is incorporated to the model. In iteration 4, indicated by the Figure 9d, the trust-region contains the solution and the trial steps actually is the optimal solution. In the next iterations, the trust-region radius is decreasing to improve the polynomial model until that one of the stop criterions is achieved.

Figure 9 – Iterations of TR illustrative example

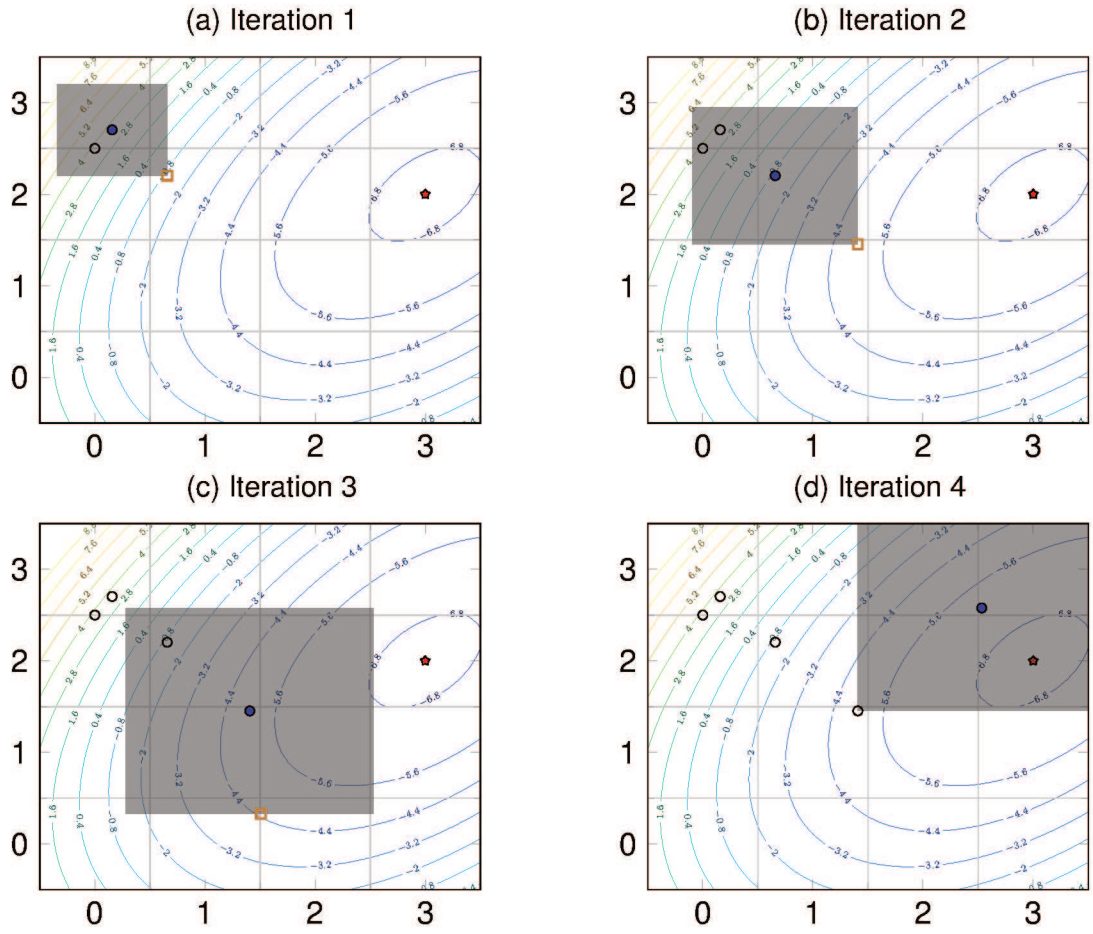
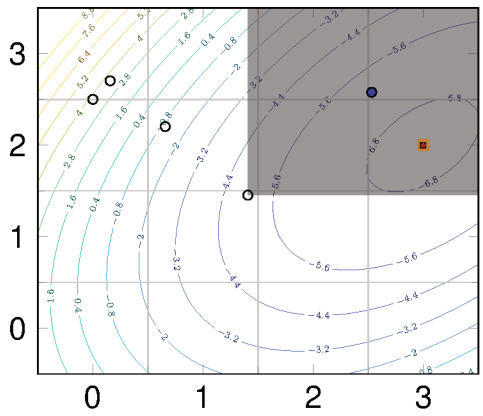
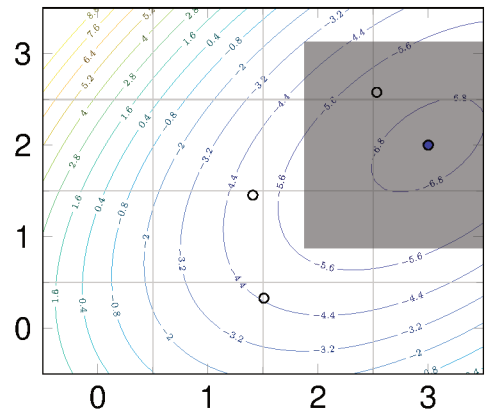


Figure 9 – Iterations of TR illustrative example

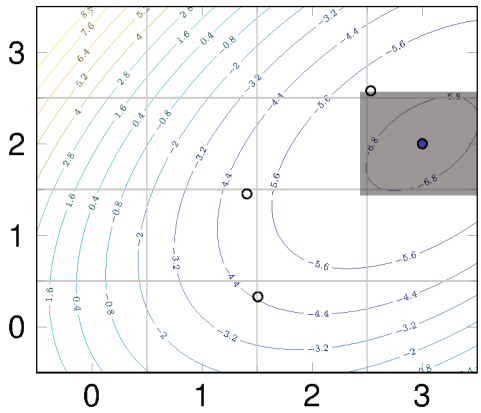
(e) Iteration 5



(f) Iteration 6



(g) Iteration 7



3.2.1.2 Building the model

The trust-region algorithm can employ different approaches for building the surrogate model. In this work we use a quadratic interpolation model. In the remaining of this section we present the trust-region algorithm used in this work as well as some model maintenance procedures.

Firstly, we define a polynomial space using standard linear algebra concepts. The set which forms a basis for such space is composed of $L = (1 + n)(2 + n)/2$ terms in $\mathbb{R}^n$, being defined as the natural basis of monomials as follows:

$$\Phi = \{\Phi_1, \Phi_2, \cdots, \Phi_L\} = \left\{1, x_1, x_2, \cdots, x_n, \frac{1}{2}x_1^2, x_1x_2, \frac{1}{2}x_2^2, \cdots, x_{n-1}x_n, \frac{1}{2}x_n^2\right\}$$

Then, any quadratic or linear polynomial m of $\mathbb{R}^n$ can be written as a linear combination of the terms of this basis of monomials Φ as

$$\begin{aligned} m(x) &= \alpha_1\Phi_1(x) + \alpha_2\Phi_2(x) + \cdots + \alpha_L\Phi_L(x) \\ &= \sum_{i=1}^{L} \alpha_i\Phi_i(x). \end{aligned}$$

For instance, the function will have up to six points $Y = \{y^1, y^2, \ldots, y^6\}$ belonging to the model in each step for the space $\mathbb{R}^2$. Generally, for $\mathbb{R}^n$, assuming that it is known L points $Y = \{y^1, y^2, \ldots, y^L\}$ and their respective function value $f = \{f(y^1), f(y^2), \ldots, f(y^L)\}$, a model m can be obtained by determining the coefficients α_i for all the equations presented above that solve $m(y^k) = f(y^k)$ for all $y^k \in Y$. Notice that this is a system of linear equations, which can be written in matrix form as presented in the following:

$$M(\Phi, Y)\alpha_\Phi = f(Y),$$

$$\begin{bmatrix} \Phi_1(y^1) & \Phi_2(y^1) & \ldots & \Phi_L(y^1) \\ \Phi_1(y^2) & \Phi_2(y^2) & \ldots & \Phi_L(y^2) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_1(y^L) & \Phi_2(y^L) & \ldots & \Phi_L(y^L) \end{bmatrix} = \begin{bmatrix} f(y^1) \\ f(y^2) \\ \vdots \\ f(y^L) \end{bmatrix}$$

This linear system can be solved if the matrix $M(\Phi, Y)$ is invertible, otherwise, $M(\Phi, Y)\alpha_\Phi = f(Y)$ will have either no solution, or an infinite number of solutions. In addition of being invertible the matrix should also be well conditioned. This means that for a small change in the independent variables, only a small change in the dependent variables will be observed. The conditioning number of the matrix is a measure used to indicate how hard it is to solve the system of linear equations. As the well conditioned measure depends of the basis Φ and the set of points Y, the basis will be chosen such that the interpolations points yield a well conditioned matrix $M(\Phi, Y)$ and the polynomial model approximates the functions sufficiently well.

In this work, the linear systems are solved with Gaussian elimination. This procedure which is formalized in Algorithm 4, which is presented in details in (GIULIANI, 2019).

Algorithm 4 Model improvement.

Parameters: the threshold $\epsilon > 0$ for the selection of pivot-elements; **Input:** An initial approximation for the pivot polynomials μ_i, possibly the base:

$$\mu_i(x) = \Phi_i(x), i = 1, \cdots, L = (n+1)(n+2)/2;$$

An initial set Y of interpolation points (possibly incomplete);
for $i = 1, \cdots, L$ **do**
  Find, if possible, $j_i \in \{i, \cdots, L\}$ such that $\|u_i(y^{j_i})\| \geq \epsilon$.
  if *such a* j_i *is found* **then**
    Swap points y^i and y^{j_i} in the set Y.
  else
    Recalculate y^i as

$$y^i \in \underset{x \in \mathbb{B}}{\operatorname{argmax}} \|\mu_i(x)\|$$

  end
  if $\|\mu_i(y^i)\| < \epsilon$ **then**
    break.
  else
  end
  for $j = i+1, \cdots, L$ **do**

$$\mu_j(x) \longleftarrow \mu_j(x) - \frac{\mu_j(y^i)}{\mu_i(y^i)}\mu_i(x)$$

  end
end

The output of Algorithm 4 is the set of points Y modified with good geometry independently of scaling. The algorithm also ensures that the model m, resulting from polynomial interpolation, satisfies the definition of Fully Linear presented before.

The Newton Fundamental Polynomials approach guarantee that the model is FL with $(n+1)$ points, so it is not needed to calculate the derivative of f or m in the formulations showed before.

The criticality test is a way to measure the stationarity of the model. When this stationarity measure is close to zero the model become more accurate because the method may be close to converge. The criticality step updates the trust-region radius forcing it to converge to zero, thereby defining a natural stop criterion for the algorithm.

The main task of the criticality step is to reduce the radius if the iterate x_k appears to be close to converge, if the model gradient at that point is smaller than a certain parameter $\|\nabla m(x_k)\| < \epsilon_c$ for a small $\epsilon_c > 0$. So Algorithm 5 builds a new model $\widetilde{m}(x_k)$ over a smaller radius $\widetilde{\Delta}$ such that $\widetilde{\Delta} \leq \mu\nabla\widetilde{m}(x_k)$ for $\mu > 0$.

Algorithm 5 Criticality step

Parameters: $\mu > 0$, $\omega \in (0, 1)$ and $\epsilon_c > 0$;
Inputs: m_k^{bct} and Δ_k^{bct};
for $i = 0, 1, 2, \cdots$ **do**
  Call the algorithm of model improvement to improve the model $m^{(i-1)}$, resulting a model m^i that if Fully Linear into the region of radius Δ^i.
  if $\Delta^i \leq \mu \|\nabla m^i(x_k)\|$ *or* $\|\nabla m^i(x_k)\| > \epsilon_c$ **then**
    Break;
  else
    $\Delta^{(i+1)} = \omega \Delta^i$;
  end
  Increase radius and set the model as $m_k = m^i$

$$\Delta_k = \min[\max(\Delta^i, \beta \|\nabla m^i(x_k)\|), \Delta_k^{bct}]$$

  return Model m_k and radius Δ_k;
end

Algorithm 6 Trust-Region

Parameters: $\eta_1 > 0$, $\eta_0 \geq 0$, $\gamma_{incr} > 1$, $\gamma_{decr} \in (0,1)$, $\mu > \beta > 0$;
Inputs: $x \in \mathbb{R}^n$, $\Delta_0 > 0$, $\Delta_{max} > 0$ and an initial point x_0;
for $k = 0, 1, 2, \cdots$ **do**

Criticality test **if** $\|\nabla m_k^{bct}\| \leq \epsilon_C$ **then**

If either the model is not Fully Linear (FL), or $\Delta_k^{bct} > \mu\|\nabla m_k^{bct}\|$ call the criticality step.

else

$m_k = m_k^{bct}$;

end

Solve the trust-region subproblem for the model after the criticality test.

$$x_k^+ = \underset{x}{\text{argmin}} \; m_k(x)$$
$$\text{s.t. } \|x - x_k\| \leq \Delta_k$$

Evaluate the function f in the trial point x_k^+, then calculate the variable ρ.

$$\rho = \frac{f(x_k) - f(x_k^+)}{m(x_k) - m(x_k^+)}$$

if $\rho > \eta_1$ *or* $\rho > \eta_0$ *and model is FL* **then**

The trial point is accepted as incumbent solution $x_{k+1} = x_k^+$.

else

$x_{k+1} = x_k$

end

if $\rho \leq \eta_1$ *and the model is not FL* **then**

The model improvement algorithm is called to improve the current model, returning the next model m_{k+1}^{bct}.

else

The current model is used as the next iteration model, $m_{k+1}^{bct} = m_k$.

end

Update trust-region radius according with ρ.

if $\rho > \eta_1$ **then**

$\Delta_{k+1}^{bct} = \min\gamma_{incr}\Delta_k, \Delta_{max}$

else if $\rho \leq \eta_1$ *and the model is FL* **then**

$\Delta_{k+1}^{bct} = \gamma_{decr}\Delta_k$

else

$\Delta_{k+1}^{bct} = \Delta_k$

end

end

3.3 SUMMARY

In this Chapter was provided a complete overview of derivative-free methods. It was explained the main aspects for each derivative-free class and subclass, also a algorithm example was presented for each case, briefly, to clarify the importance and the capability of this methods that do not use the derivative information to optimize. The aim of next Chapter is to expose the optimization problem properly and to formalize the concepts that include geological uncertainties in the implementation.

4 ENSEMBLE TRUST-REGION OPTIMIZATION

Ensemble trust-region optimization is a combination of established strategies. An Ensemble is known to represent the uncertainty in reservoir models, taking into consideration uncertain geological parameters to better represent the reservoir. Trust-region optimization is a derivative-free method that optimizes small regions to find the optimal solution for the whole system. This concept is new in production optimization, albeit it was already proposed in history matching problems.

As the objective function was presented in Chapter 2, Section 4.2 will reformulate the same objective function by this accomplishing its goal using the ensemble knowledge, provided in Section 4.1. At the end, the chapter presents the entire implementation framework of the previous section, considering notions of optimization.

The use of ensemble began in History Matching, which is an optimization area that focuses on modeling reservoirs based on information from inputs, outputs and geological proprieties. One of the first articles that used ensemble in History Matching were (CHEN, C. et al., 2010; CHEN, Y. et al., 2009; CHEN, Y.; OLIVER, 2010). Chen et al. shows in (CHEN, Y. et al., 2009) an improvement of 22% compared to a reactive strategy in the NPV results. They proposed was a closed-loop reservoir management for data assimilation using an Ensemble Kalman Filter (EnKF) and for the model based optimization a robust gradient-free also based on ensemble.

In Production Optimization, (VAN ESSEN et al., 2009) proposes to optimize the Expected NPV over an ensemble of reservoir models using a gradient based method, in which the goal is to reduce the risk that arises from geological uncertainty. The work compares this procedure, called Robust optimization (RO) against a nominal optimization (NO) and a reactive control approach. They used eight injectors and four producers as control to optimize one thousand realizations of a tri-dimensional reservoir modeling in a fluvial depositional environment with a known main-flow direction.

4.1 ENSEMBLE OF GEOLOGICAL REALIZATIONS TO CHARACTERIZE PARAMETRIC UNCERTAINTY

Often the reservoir performance is measured by flow simulation, which provides the field rates to calculate objectives of interest such as the cumulative oil rate (COP) and the NPV. Some of the parameters in a reservoir model are uncertain, i.e. there is lack of assurance about the truth of the statement or about the exact magnitude of an unknown measurement or number (OLEA, 1991). In reservoir model the uncertainty is typically geological. To ignore uncertainty and assume perfect knowledge of the reservoir is generally an unacceptable approach, since the production parameters are highly dependent on uncertain reservoir geological properties, such as porosity or permeability. So, it is important to incorporate uncertainty in the reservoir model.

Under the perspective of reservoir management, the presence of uncertainty means that there are infinitely many possible models that may represent the true reservoir. One alternative to handle uncertainty in reservoir modeling is to sample the probability distribution of the uncertain parameters in order to generate a finite set of possible realizations, called ensemble realizations. Such realizations can be generated by an expert in geology or by a computer program. In the present work the realizations will be empirically created by using a baseline model as a reference and varying the uncertain parameters to generate the other realizations. Notice that there are infinite possibilities to combine the geological uncertainties within the feasible range for the uncertain parameters. Although the ensemble is created from a reference model, from the optimizer perspective, it is impossible to know which realization represents the true reservoir.

4.2 ROBUST OPTIMIZATION USING THE EXPECTED VALUE

As the ensemble is a discrete representation of the geological uncertainty in reservoirs, it enables the handling of uncertainty in a computationally feasible way in optimization procedures. However, there are still some open questions related to the efficient use of ensemble in optimization routines, e.g. how to aggregate the ensemble into the optimization algorithm? This section comes to clarify this and other points.

There are several ways to use the ensemble inside an optimization algorithm. The majority is related to the way of measuring the contribution of each element within the entire ensemble. As an ensemble is a set composed of different functions or models, each element will generate a value and there are different measures to calculate the value of an ensemble. One of these measures is called the Expected Value. Beyond the Expected Value, there are other measures in production optimization such as the Worst-case scenario, Value-at-risk (VaR) and Conditional value-at-risk (CVaR_{α}). A brief explanation about these measures is provided in (CAPOLEI et al., 2017).

This work uses the expected value as the measure to be optimized in the optimization procedure. The expected value, which is also known as probability-weighted average, is simply a way to measure the average of a discrete set of variables based on their associated probabilities. The mathematical definition is given as follows:

$$\text{Expected Value} = \text{E}(\mathbb{X}) = \sum_{i=1}^{n} p_i x_i = p_1 x_1 + p_2 x_2 + \cdots + p_n x_n \tag{5}$$

where $\mathbb{X}$ is a random value which can assume any value from a set of n values, $\mathbb{X} = \{x_1, x_2, \ldots, x_n\}$, x_i is the i^{th} element of the set and p_i is the probability of the element x_i happening, i.e. the weight that this element has in the set.

As all the elements are assumed to have the same probability to happen, i.e. a

uniform distribution, Eq. (5) is simplified as follows:

$$\mathrm{E}(\mathbb{X}) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6}$$

where n denotes the number of elements in the set of variables.

Since the ensemble is a set of reservoir models that represent a model with geological uncertainties, it is not possible to determine which model best describes the real reservoir. Then, the expected value for a given measure z will be calculated with the same probability as follows:

$$\mathrm{Exp}\left[\mathbb{Z}(\mathbb{X})\right] = \frac{1}{n}\sum_{i=1}^{n} z_i(\mathbb{X}) \tag{7}$$

where the ensemble is defined by the set $\mathbb{Z} = \{z_1, z_2, \ldots, z_n\}$, with z_i denoting the i^{th} reservoir model and $\mathbb{X}$ being the set of optimization variables. Notice that $z_i(\mathbb{X})$ denotes the NPV value for the model z_i at $\mathbb{X}$ point.

4.3 ENSEMBLE TRUST-REGION OPTIMIZATION

The formulation of the standard well control optimization problem, inspired from (SILVA et al., 2020), was presented in Chapter 2. Below follows the formulation of the robust production optimization problem, using the definition of the expected value over a discrete set, presented as ensemble.

$$\begin{aligned} \max_{u} \quad & Exp(\mathbb{X}) \\ \text{s.t.} \quad & \mathbb{X}_{lb} \leq \mathbb{X} \leq \mathbb{X}_{ub} \end{aligned} \tag{8}$$

where the objective function is the expected value for the ensemble considering as control variable the set $\mathbb{X}$. As constraints, it is considered the upper and lower bounds that impose conditions over the control variable.

As discussed previously, the simulation is a black-box function. The Eq. (9) shows the relation between the nomenclatures, f_i represents the function value for the simulation of the i^{th} realization z_i under the set of controls $\mathbb{X}$.

$$f_i \rightarrow z_i(\mathbb{X}) \tag{9}$$

The process that occurs in the algorithm is synthesized on Figure 10, where we divided it in three processes, called Optimization, Simulation and Parametrization layer.

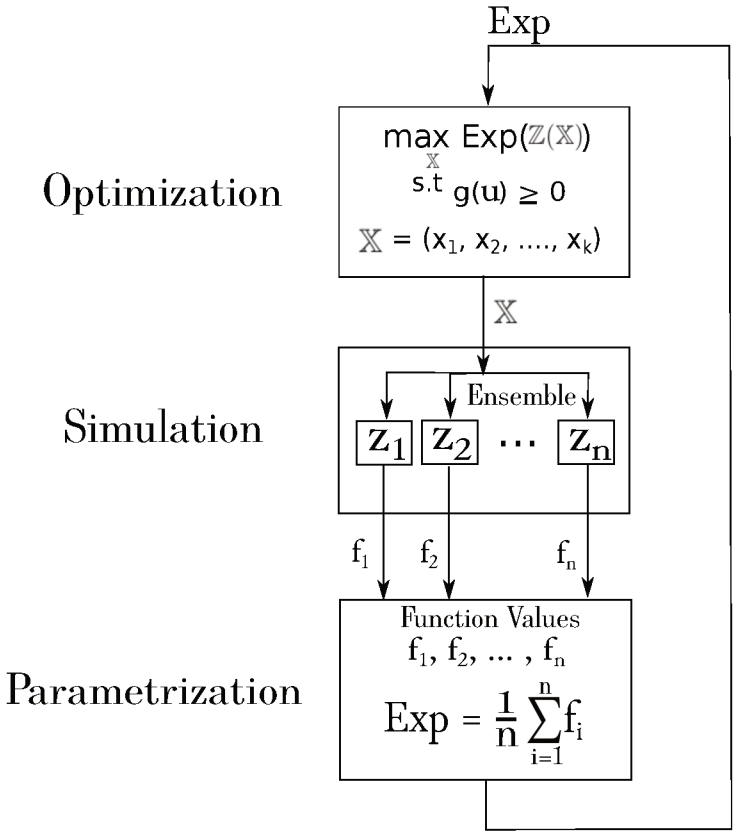


Figure 10 – Formulation layer (Modified from (KRISTOFFERSEN et al., 2020))

The optimization layer has as the objective of maximizing the expected net preset value (NPV) controlling a set of variables $\mathbb{X}$. In our case, it will be used the production bottom hole pressures (BHP) and consider fixed the other parameters, such as the rate of water injection. The controls set $\mathbb{X}$ is sent to the simulation layer, which evaluates $\mathbb{X}$ for all the models of the ensemble $\mathbb{Z}$. Once the simulation finishes, the NPV of each model $f_1, \ldots, f_n$ is sent to the Parametrization layer that calculates the expected value to be returned to the optimization layer. The algorithm repeats this process until reaching the stopping criterion.

5 SIMULATION ANALYSIS WITH EXPLICIT FUNCTIONS

In this Chapter we present a simulation analysis conducted with analytical functions in order to demonstrate the correctness of the robust trust-region algorithm proposed in the previous Chapter. To assess the method's capabilities to deal with non-convexities and noise we use two types of analytical functions, a smooth non-convex function known as the Rosenbrock function, and a noisy function called the Rastrigin function.

Firstly the analytical form of these functions will presented, and then we report the results obtained with the application of the trust-region algorithm to the minimization of the functions under uncertainty. Mathematically, the Rosenbrock and Rastrigin functions are presented in the equations (10) and (11), respectively. In both cases we utilize the domain in $\mathbb{R}^2$.

$$f_{\text{Rosenbrock}}(\mathbb{X}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right] \tag{10}$$

$$f_{\text{Rastrigin}}(\mathbb{X}) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) \right] \tag{11}$$

in which $\mathbb{X} = (x_1, \ldots, x_n)$ is a vector with the function argument.

The surfaces of the functions in $\mathbb{R}^2$ are shown in Figures 11 and 12. Notice that the grids are different in these functions in order to enhance the main characteristics of each function. Nevertheless the grids will be kept constant for all the experiments presented throughout this chapter.

These functions were selected to test the algorithm because they have different properties. The Rosenbrock function is a popular function for testing gradient-based optimization algorithms for being a non-convex and unimodal[1] function, with the global minimum in a parabolic valley. This last aspect can cause trouble for the convergence of standard optimization algorithms. On the other hand, the Rastrigin function is a typical example of a non-linear multimodal[2] function. Because it is a noisy function, it has a large number of local minima. In general, noisy functions are challenging problems for optimization procedures, but this function in particular has one globally optimal solution for the minimization problem, which is at the center of the grid, the point $(0, 0)$. Finding the global minimum is a challenging task for an algorithm, which would have to provide a certificate that the solution has a better objective value than any another solution.

Ten variations of each analytical function, Rosenbrock and Rastrigin, were empirically created to represent the parametric uncertainties of the model. This set of

[1] According to Surhone, Tennoe and Henssonow (2010), a function $f(x)$ is unimodal if for some value m (the mode), it is monotonically increasing for $x \leq m$ and monotonically decreasing for $x \geq m$.

[2] A multimodal function is a function that has more than two local minima or maxima.

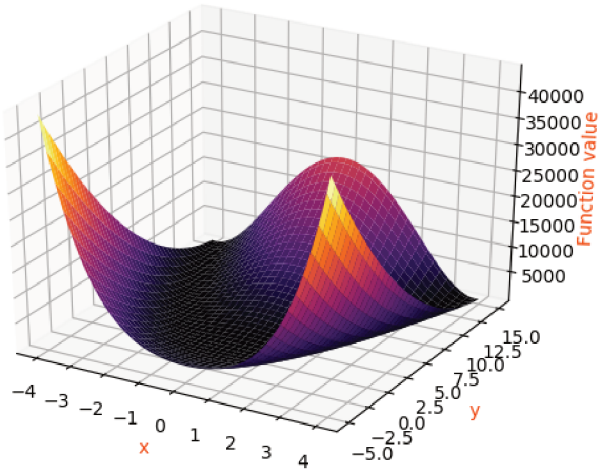


Figure 11 – Rosenbrock function surface

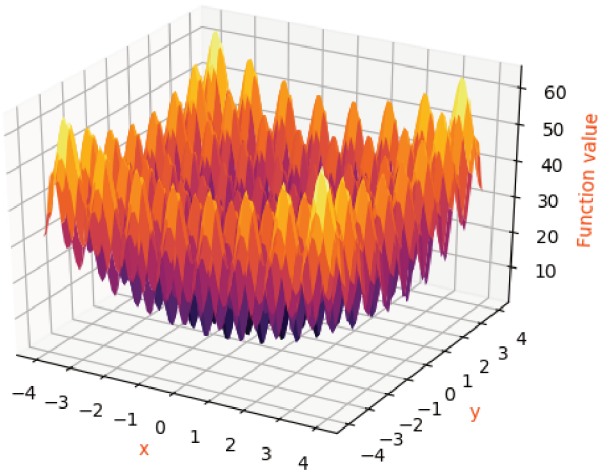


Figure 12 – Rastrigin function surface

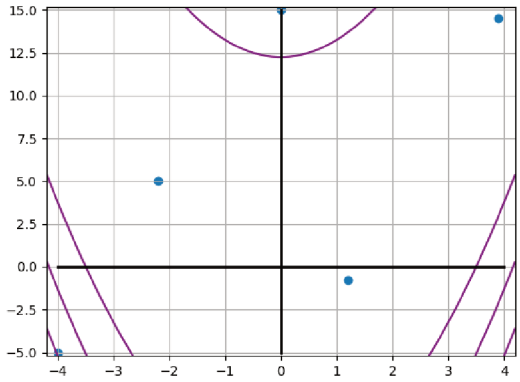
functions containing variations of the original function form the ensembles for each function, which are used in the optimization case studies later on.

As stressed, derivative-free optimization methods are rather dependent on initial points for being local methods. Then, to validate the results some variations of the initial point and Trust-Region radius were tested. One point in each quadrant was chosen arbitrarily, which are indicated in sub-figures of Figure 13 as blue circles. Different initial radius sizes are considered, namely $R_1 = 2$, $R_2 = 5$, and $R_3 = 10$ for Rosenbrock, $R_1 = 2$ and $R_2 = 4$ for Rastrigin. The reasoning behind these choices are the coverage of specific regions of the domain, for example, in Rosenbrock case namely a radius size of 2 covers one quadrant in the x-coordinate, whereas a radius size of 5 covers the entire quadrant related to the y-coordinate, and 10 covers approximately the entire domain. The initial points from which the algorithm starts from are indicated the following Table 1.

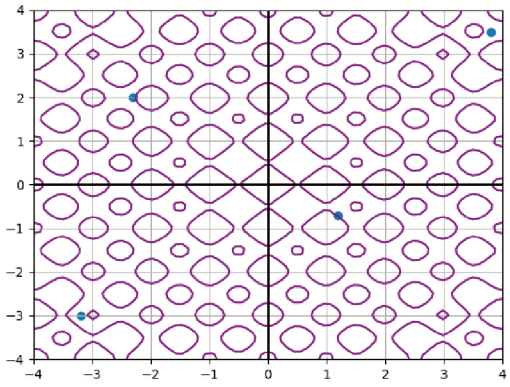
Table 1 – Initial points

Name	Rosenbrock	Rastrigin
P_1	(3.9, 14.5)	(3.8, 3.5)
P_2	(1.2, -0.8)	(1.2,-0.7)
P_3	(-4.0, -5.0)	(-3.2,-3.0)
P_4	(-2.2, 5.0)	(-2.3, 2.0)
P_5	(0.0, 15.0)	–

As each initial point is chosen to be at the center of one of the four quadrants splitting the feasible domain, there is one additional point for Rosenbrock, P_5, which was included in order to observe the behavior of the optimization algorithm starting from the top of the function. In the sub-figures of Figure 13, the purple lines represent the level curves for each respective standard function.



(a) Rosenbrock



(b) Rastrigin

Figure 13 – Initial point distribution

Once the initial set of parameters is created, the ensemble of functions is obtained for each case, Rosenbrock and Rastrigin. Then, a minimization is performed using the expected value as the objective function and the trust-region algorithm as the solver. Some cases will be created varying the initial point and radius to assess the capabilities of this approach. At the end, a discussion of the results will be presented.

5.1 ROSENBROCK ENSEMBLE

The Rosenbrock Ensemble is composed of a set of analytical functions which are variations of the standard Rosenbrock function given in Eq. (10). Such functions are presented in the Equations of Appendix B.1. As discussed before these functions are obtained by varying certain parameters to represent the model uncertainties. The ensemble of functions with the different parameters are represented in Eq. (12), where $\alpha_{Ro} \in [-6, 6]$, $\beta_{Ro} \in [-1.8, 4]$, $\gamma_{Ro} \in [-0.8, 0.7]$ and $\omega_{Ro} \in [-0.3, 0.8]$. The actual parameters that define the functions of the ensemble are given Table 2. We assume these ranges cover the uncertainty spectrum of all the parameters, such that the ensemble of

functions constitute a discrete approximation of the parametric uncertainty.

$$\sum_{i=1}^{n-1} \left[(100 + \alpha_{Ro})(x_{i+1} + \beta_{Ro} - (x_i + \gamma_{Ro})^2)^2 + (x_i - 1 + \omega_{Ro})^2 \right] \tag{12}$$

Table 2 – Rosenbrock: parameters of the function ensemble

Element	α_{Ro}	β_{Ro}	γ_{Ro}	ω_{Ro}
z_1	-5	4	-0.8	0.2
z_2	-3	0.3	-0.4	0
z_3	3	0.3	0.4	-0.2
z_4	-6	-1.8	-0.3	1.8
z_5	-2	0	0.7	1.3
z_6	-5	1.8	-0.5	0
z_7	6	0	-0.7	0.8
z_8	-4	4	0	-0.3
z_9	5	-2	0	1.7
z_{10}	-10	0.6	-0.2	0

The surfaces of the ensemble members are depicted in Figure 14. Notice that all the surfaces are plotted in the same grid which was used to illustrate the standard Rosenbrock function. To highlight the differences between the surfaces, we show in Figure 15 the surfaces of all ensemble members in one plot. To further illustrate such differences, we present a 2D plot for a constant y-coordinate value ($y = 0$) in Figure 16a, and the level curves of the functions in Figure 16b. Notice that the parametric uncertainty described by the ensemble introduced variations both in the neighborhood of the original local optimum, and also in more distant regions, arguably making the ensemble suitable for the studies conducted in this chapter.

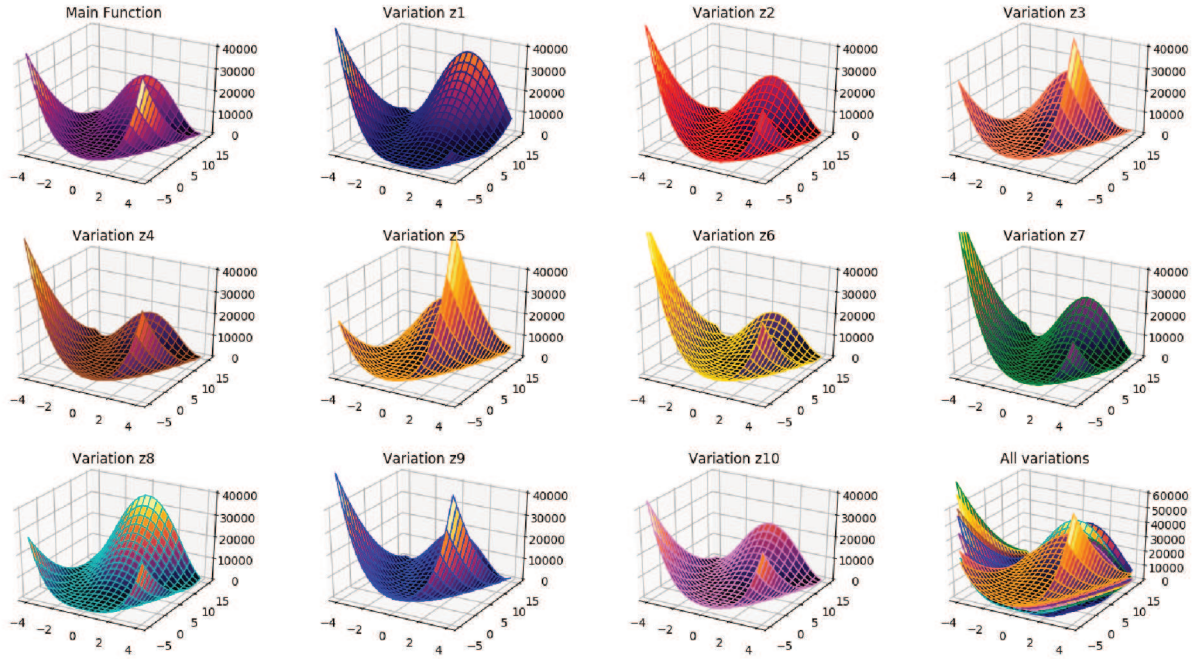


Figure 14 – Rosenbrock ensemble elements: Individual surface

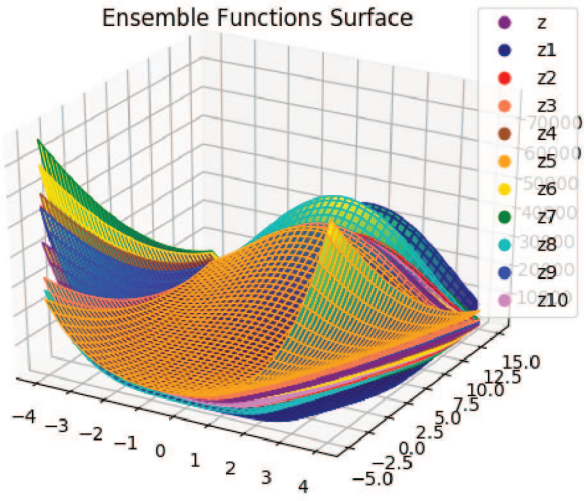
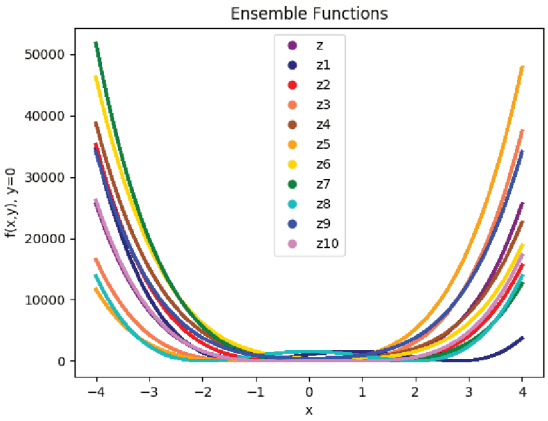
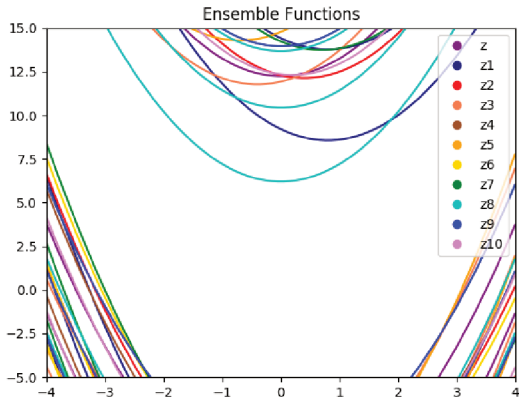


Figure 15 – Rosenbrock ensemble elements: All surfaces in the same graph



(a) Rosenbrock ensemble elements: 2D plot



(b) Rosenbrock ensemble elements: Level curve

Figure 16 – Rosenbrock ensemble elements details

Figure 17 shows the expected value $E(\mathbb{X})$ surface, assuming equal probability for each element of the ensemble to occur. Then, mathematically, $E(\mathbb{X})$ is defined as in Eq. (7). Notice that the expected value surface has the same characteristics (curve format) observed in each one of the ensemble elements. This is not a general rule, and therefore it can be considered just for this set of elements.

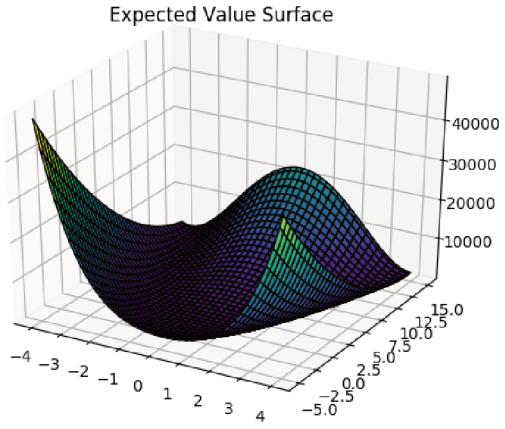


Figure 17 – Rosenbrock expected value surface

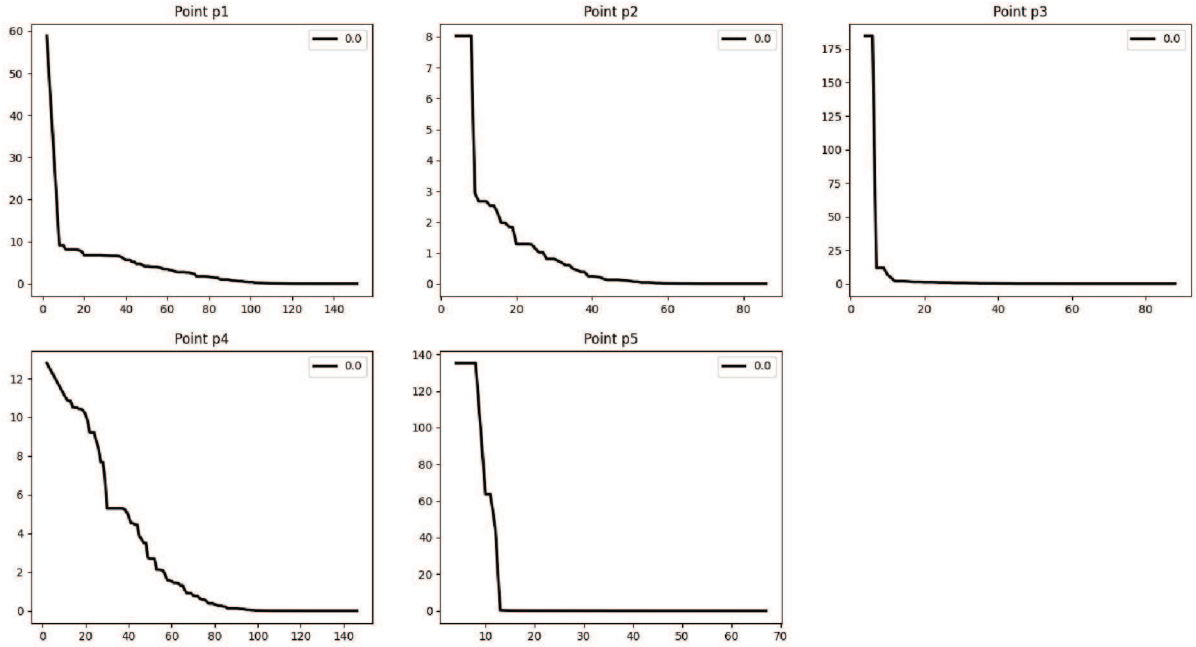
5.1.1 Results

The optimization results obtained in the minimization of the standard function given in Eq. (10), in other words with no ensemble, using the trust-region algorithm are presented in Figure 18. The optimal function value of each individual function of the ensemble is 0.0 despite the optimal points in the domain being different.

With the minimization results over the standard function, it is possible to analyze the capabilities of the expected value approach by evaluating its performance for several conditions. Each sub-figure in Figure 19 shows the evaluations for different initial points and radius sizes. The color lines represent the respective function value for each element of the ensemble $\mathbb{Z} = \left\{z, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}\right\}$, while the curve in black represents the expected function value. Notice that the optimum value, presented in the legend as black, is the same independently of the initial parameters. Observe that, despite the objective value and the solution point are the same for all the cases (Table 3), the convergence curve changes, which happens because the algorithm follows distinct ways during optimization.

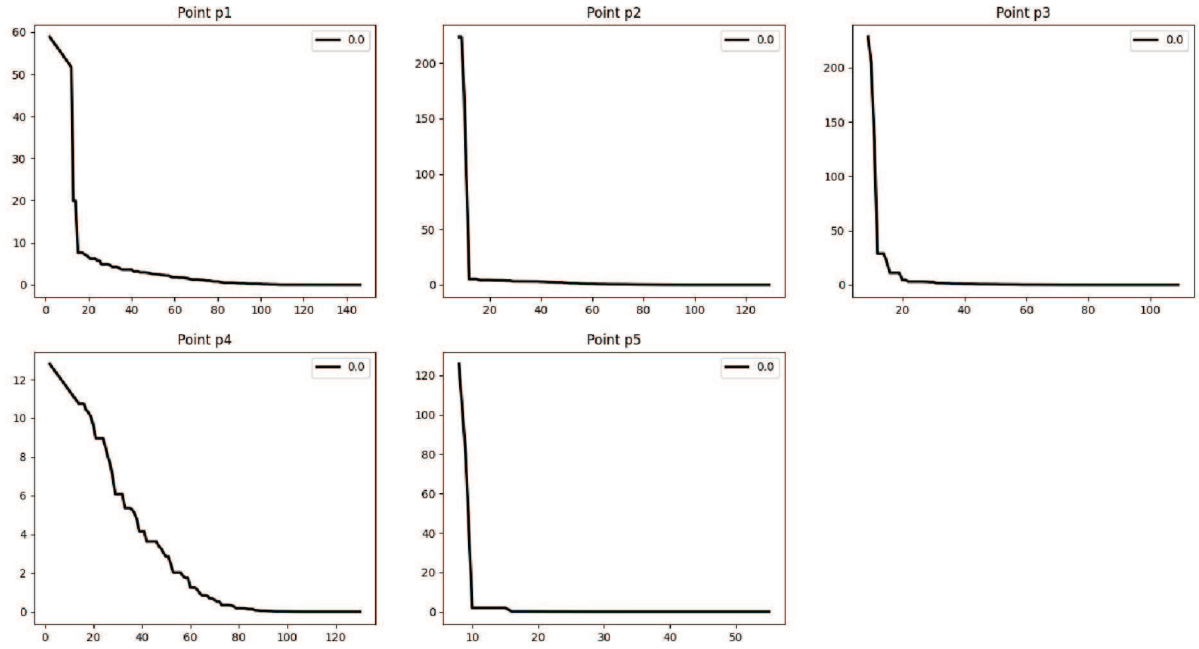
Table 3 – Rosenbrock optimal solutions.

Init. Radius Size	Init. Point	Standard Rosenbrock Solution Point	Standard Rosenbrock OFV	Rosenbrock Ensemble Solution Point	Rosenbrock Ensemble OFV
R_1	P_1	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
	P_2	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
	P_3	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
	P_4	(1.00, 0.99)	0.0	(-0.228, 0.033)	345.74
	P_5	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
R_2	P_1	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
	P_2	(1.00, 0.99)	0.0	(-0.228, 0.033)	345.74
	P_3	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
	P_4	(0.99, 0.99)	0.0	(-0.228, 0.033)	345.74
	P_5	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
R_3	P_1	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
	P_2	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74
	P_3	(1.00, 0.99)	0.0	(-0.228, 0.033)	345.74
	P_4	(1.00, 0.99)	0.0	(-0.228, 0.033)	345.74
	P_5	(1.00, 1.00)	0.0	(-0.228, 0.033)	345.74

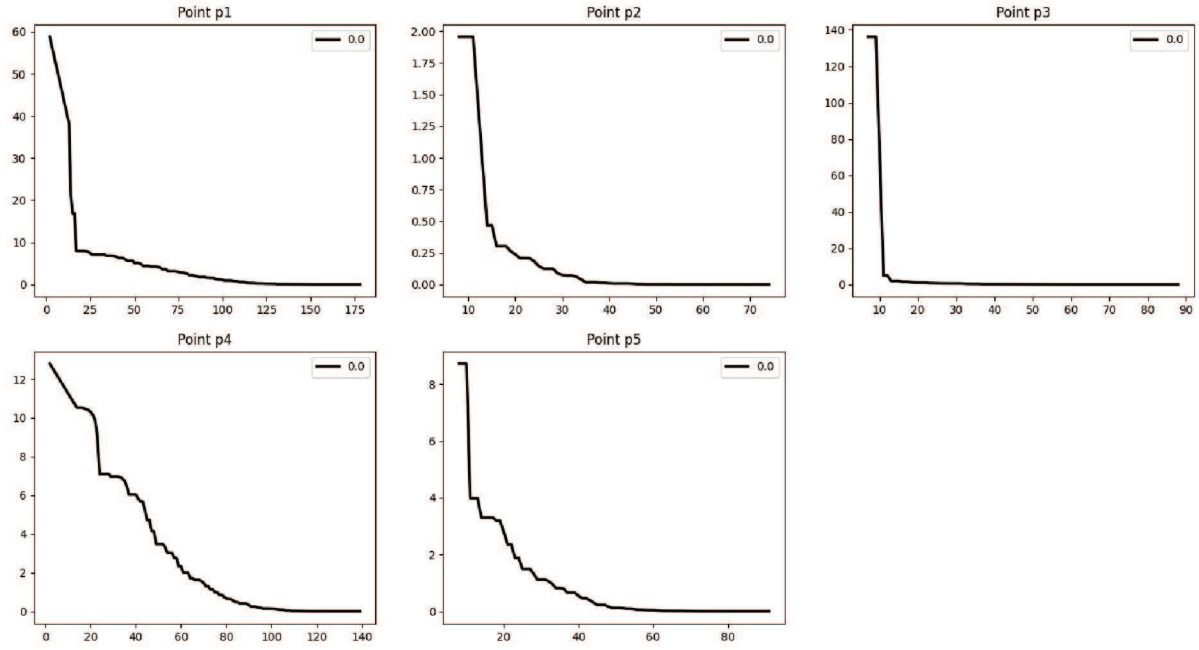


(a) Initial radius size R_1

Figure 18 – Rosenbrock optimization results for different starting points and radius sizes

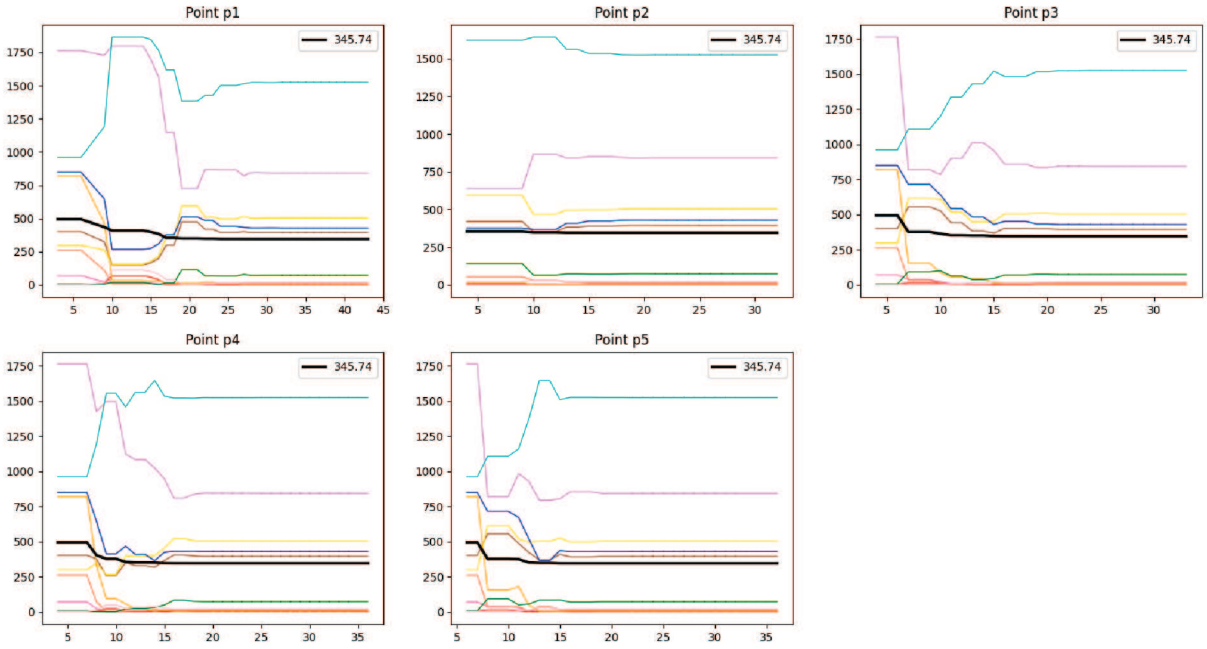


(b) Initial radius size R_2

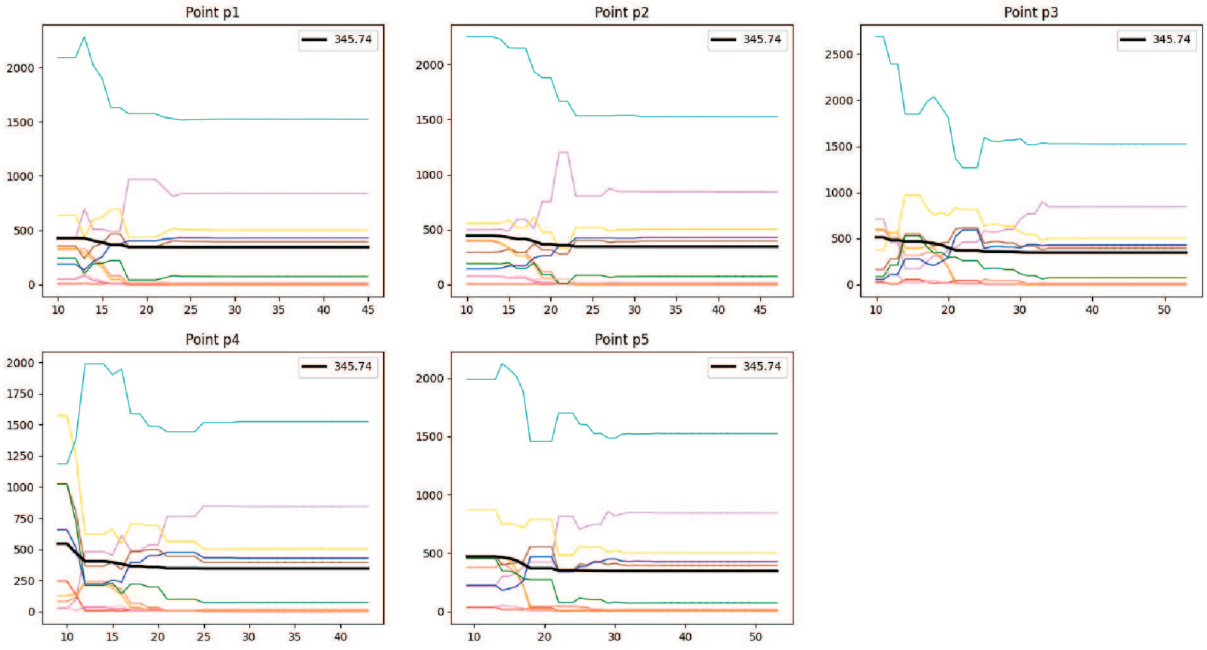


(c) Initial radius size R_3

Figure 18 – Rosenbrock optimization results for different starting points and radius sizes

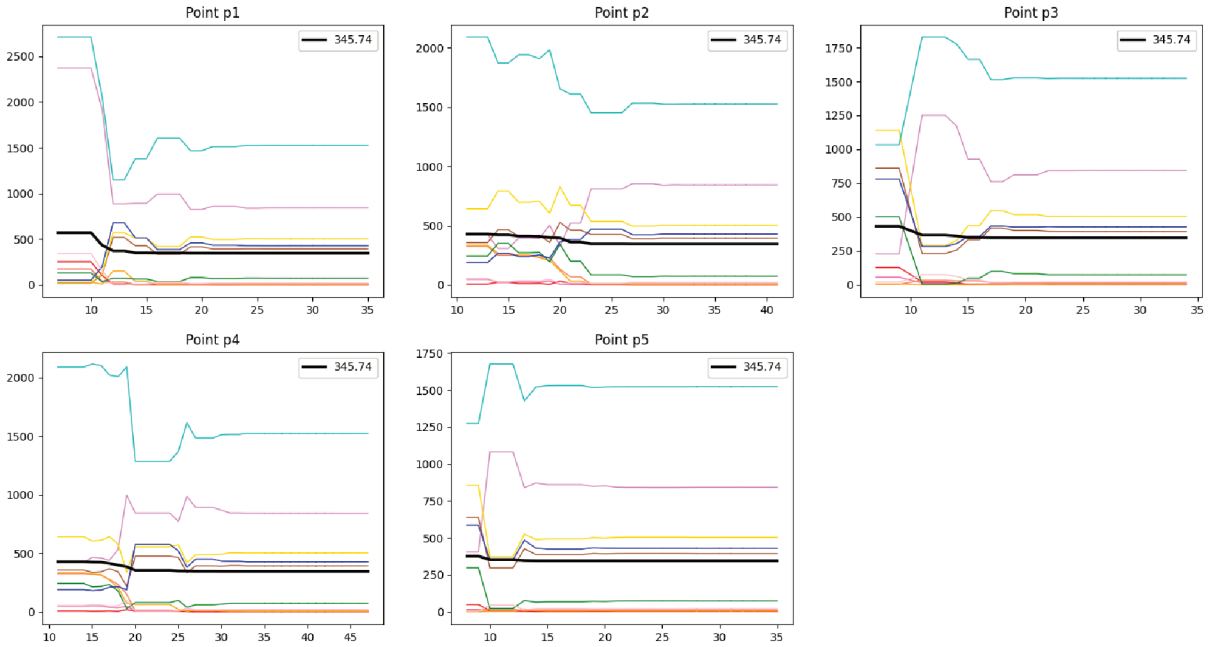


(a) Initial radius size R_1



(b) Initial radius size R_2

Figure 19 – Optimization results obtained with the Rosenbrock ensemble



(c) Initial radius size R_3

Figure 19 – Optimization results obtained with the Rosenbrock ensemble

As showed before, the function value obtained in the optimization without the ensemble set was zero and the optimum solution value obtained considering the ensemble set was 345.74. A large difference between these values can be attributed to the high conservative behavior of the expected value optimization. Observing Figure 19a, for example, it is possible to see that the function values have at least five ensemble members lying above the final expected value, which indicates the conservativeness of the expected value measure. However, the decrease of the objective function was of about 40% in relation with the initial function value, showing a good performance of the algorithm.

As stressed before, the approach will be validated using two well known functions, Rosenbrock and Rastrigin. This section reported results from the Rosenbrock ensemble, a non-convex and unimodal function, for which the algorithm worked well and obtained solutions with a conservative measure. Therefore, the algorithm works well on this first case. The same experimental procedure will be applied to the Rastrigin function in the next section, which aims to study the approach behavior considering a non-linear multimodal function with uncertainties.

5.2 RASTRIGIN ENSEMBLE

A variation set of the standard Rastrigin function was empirically generated to compose the Rastrigin Ensemble. Such member functions are presented in the equations of Appendix B.2 which represent the model uncertainties. The standard form

of the Rastrigin function is the one given in Eq. (13) below:

$$f_{\text{Rastrigin}}(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) \right] \tag{13}$$

and its version with varied parameters in Eq. (14) that follows:

$$\begin{aligned} \widetilde{f}_{\text{Rastrigin}}(x) = (10 + \alpha_{Ra})\, n + \left(x + \beta_{Ra}^{x}\right)^2 - \left(10 + \gamma_{Ra}^{x}\right) \cos\left(2\pi\left(x + \omega_{Ra}^{x}\right) + \psi_{Ra}^{x}\right) \\ + \left(y + \beta_{Ra}^{y}\right)^2 - \left(10 + \gamma_{Ra}^{y}\right) \cos(2\pi(y + \omega_{Ra}^{y}) + \psi_{Ra}^{y}) \end{aligned} \tag{14}$$

which is varied according with the parameters α_{Ra}, β_{Ra}, γ_{Ra}, ω_{Ra}, and ψ_{Ra} which define the variations. Table 4 presents in the columns the numerical values of the parameters $\alpha_{Ra} \in [-5, 5]$, $\beta_{Ra}^{x} \in [-0.6, 0.3]$, $\gamma_{Ra}^{x} \in [-20, 5]$, $\omega_{Ra}^{x} \in [-0.4, 0.3]$, $\psi_{Ra}^{x} \in [0, 1]$, $\beta_{Ra}^{y} \in [-0.5, 0.2]$, $\gamma_{Ra}^{y} \in [0, 5]$, $\omega_{Ra}^{y} \in [-0.4, 0.2]$ and $\psi_{Ra}^{y} \in [0, 0.5]$ which correspond to each ensemble element (variation). The ranges were arbitrarily generated to empirically cover the uncertainty spectrum of all the parameters.

Table 4 – Rastrigin: Ensemble functions parameters

Element	α_{Ra}	x				y			
		β_{Ra}	γ_{Ra}	ω_{Ra}	ψ_{Ra}	β_{Ra}	γ_{Ra}	ω_{Ra}	ψ_{Ra}
z_1	5	0	5	0	0	0	5	0	0
z_2	0	-0.2	0	-0.2	0	0	0	0	0
z_3	0	0	0	0	0.4	0.2	0	0.2	0.5
z_4	1	0	0	0	0.6	0.15	0	0.15	0
z_5	5	-0.2	0	-0.2	0	0	0	0	0
z_6	4	0.3	0	0.3	0	-0.1	0	0	0
z_7	-5	-0.4	-20	-0.4	0	-0.4	0	-0.4	0
z_8	4	0.2	5	0	0	-0.2	5	0	0
z_9	3	0	0	0	0.6	0.1	0	-0.2	0
z_{10}	2	-0.6	0	0	1	-0.5	0	0	0

The ensemble members are plotted in Figure 20 using the same grid as the surface of standard Rastrigin function. In Figure 21 the ensemble elements are showed individually to highlight the variations between the surfaces. Such differences can be better observed in Figure 22 that presents a 2D plot for a constant y-coordinate value ($y = 0$).

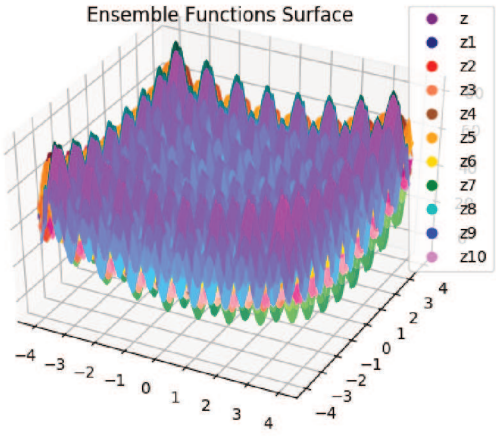


Figure 20 – Rastrigin ensemble elements: All the ensemble surfaces plot

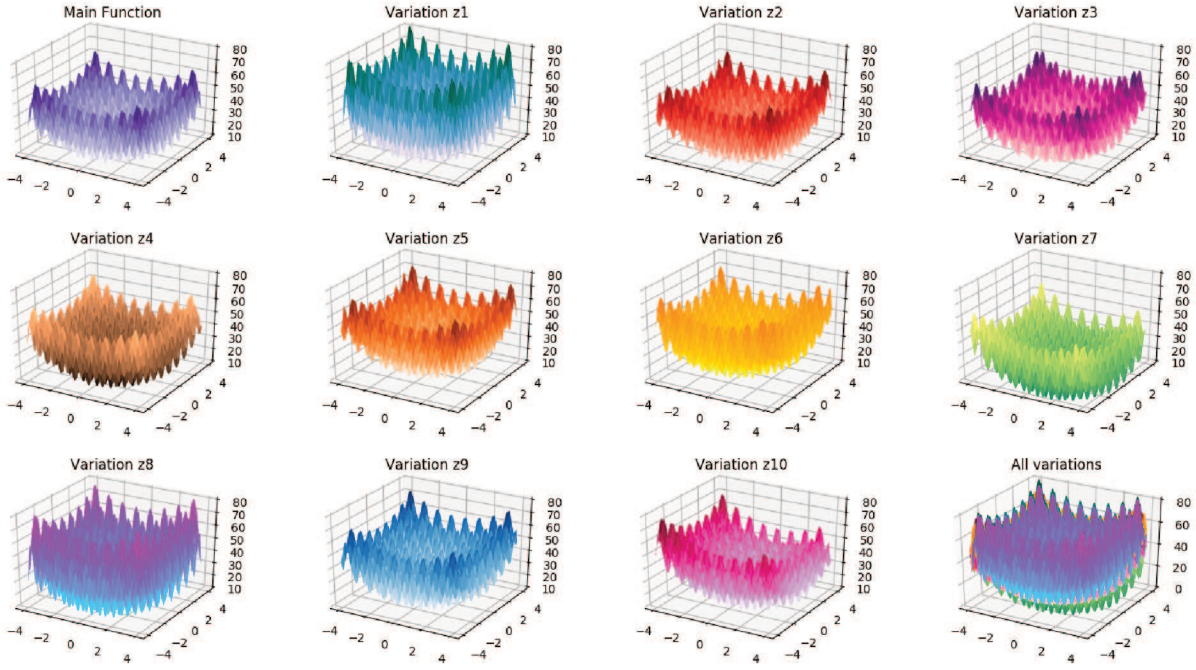


Figure 21 – Rastrigin ensemble elements: Individual surface plot

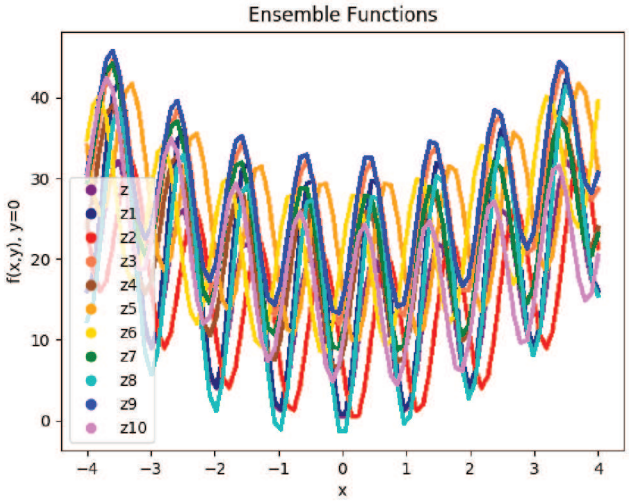
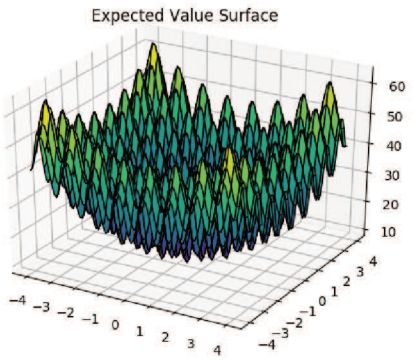


Figure 22 – Rastrigin ensemble elements: 2D plot

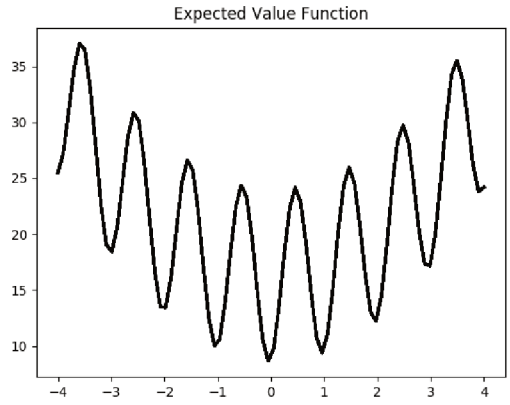
The expected value function $E(\mathbb{X})$, defined mathematically in Eq. (7), is shown for the Rastrigin ensemble case in Figure 23a, where the $E(\mathbb{X})$ surface is plotted.

In Rastrigin case, it is possible to hope for an expected value surface more smooth than the standard surface, but as presented in Figures 23a and 23b the expected value formulation maintain the noisy characteristic of Rastrigin. In the 2D plot presented in Figure 23b one variable was fixed at the origin ($y = 0$). This means that the difficulties towards finding minima of the expected-value function remain unaffected in relation to the baseline Rastrigin function.



(a) Rastrigin expected value surface

Figure 23 – Rastrigin expected value



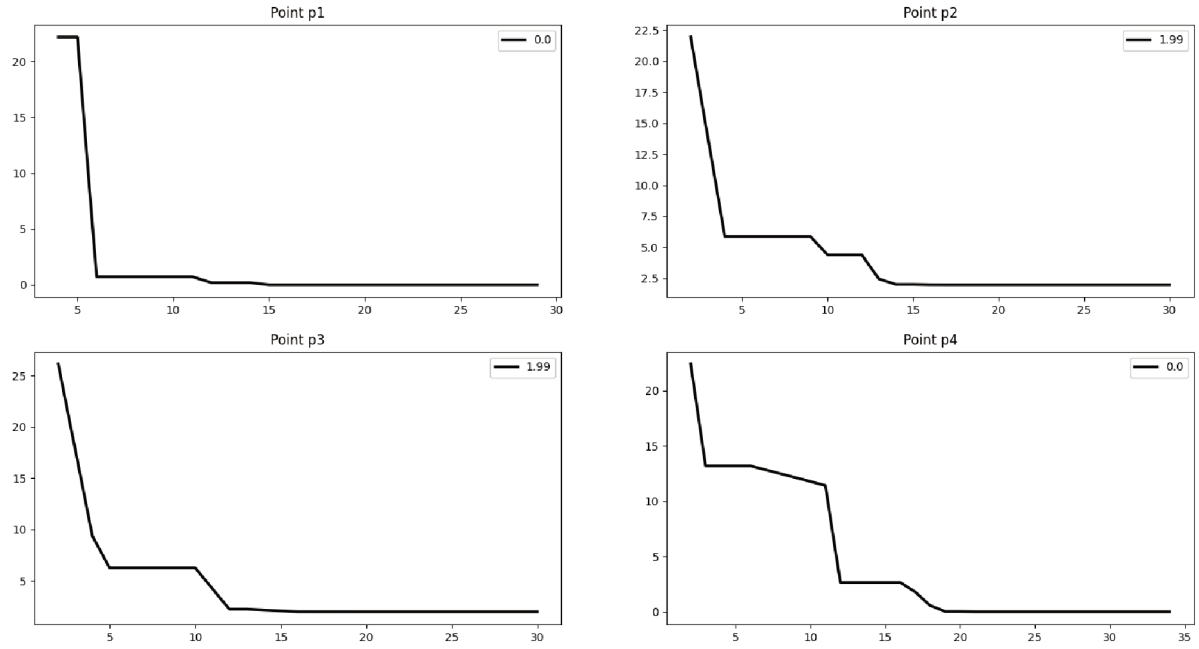
(b) Rastrigin expected value 2D plot

Figure 23 – Rastrigin expected value

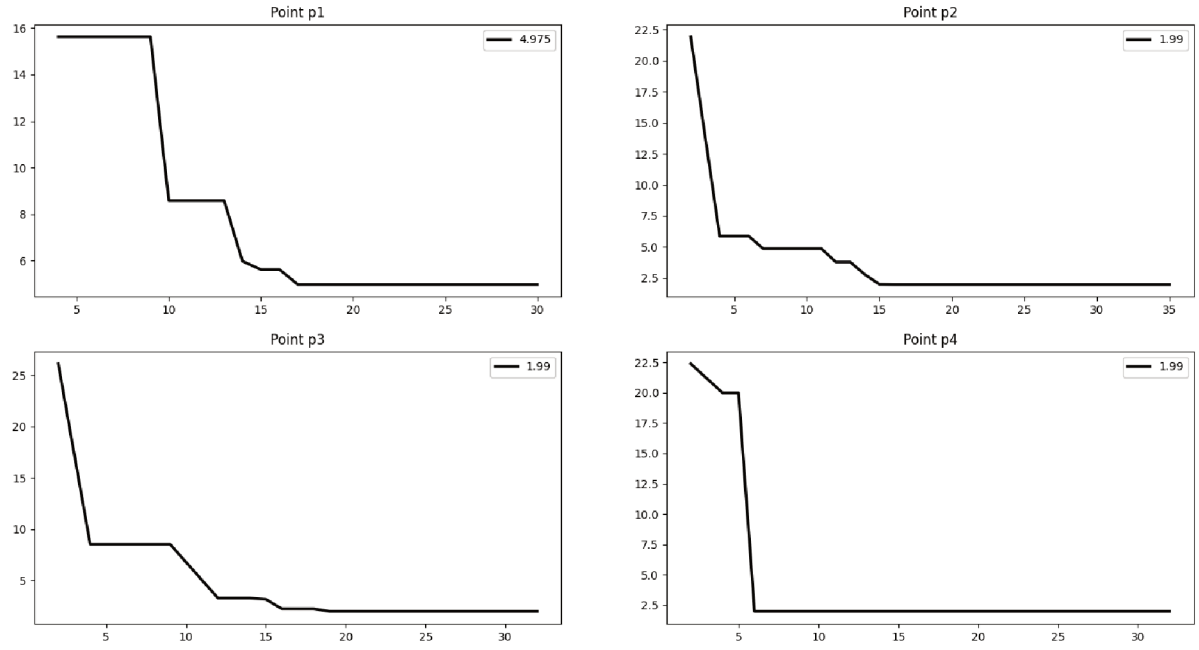
There exist an infinite number of possibilities to generate the ensemble, since each ensemble member is varied empirically. As the expected value function is a combination of the ensemble members, it is possible that the final formulation stays easier to solve, once that the set of ensemble elements can generated an expected value function smoother than the standard one.

5.2.1 Results

Figures 24a and 24b show the optimization results obtained by applying the trust-region derivative-free algorithm to the minimization of the standard Rastrigin function, given in Eq. (13). The optimal function value can be observed in the legend of each function, which varies between 0 and 4.975.



(a) Initial radius size R_1



(b) Initial radius size R_2

Figure 24 – Evaluation of Rastrigin function without ensemble

As stressed, the Rastrigin function has a global minimum at $(0, 0)$. Under certain initial conditions the algorithm was able to find the global optimum solution. Being a kind of local search, the point to which the algorithm converges depends on the initial guess and initial radius size.

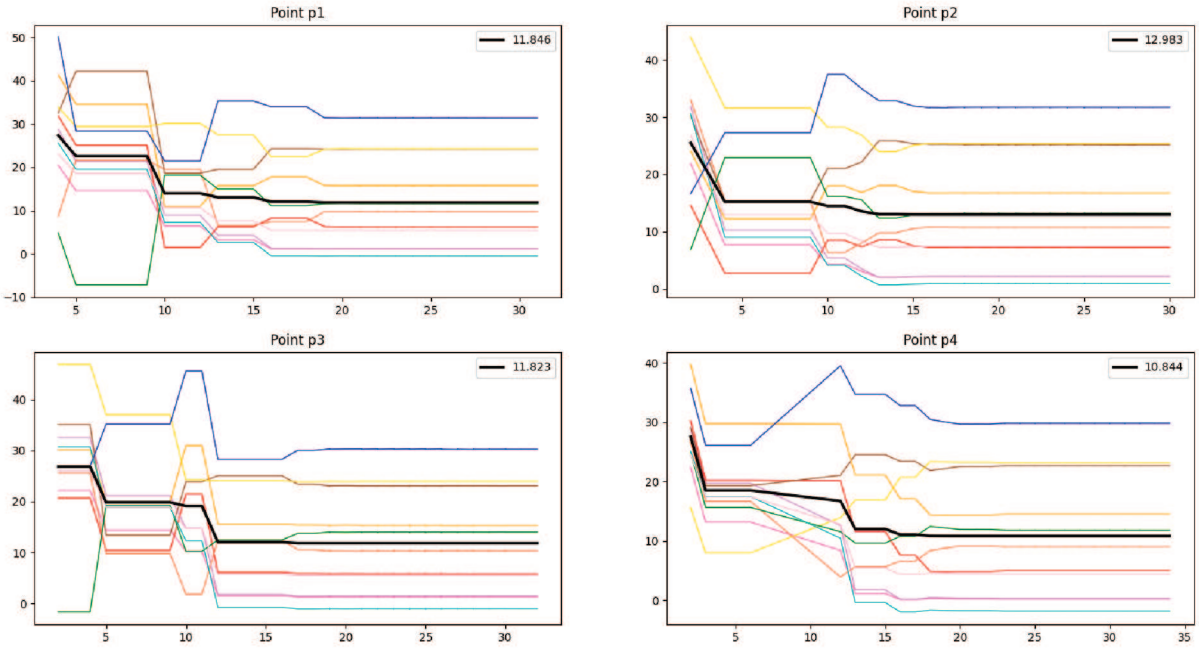
The results from the minimization of the Rastrigin ensemble case are shown in Figures 25a and 25b, where the a colorful curve represent the function value of one ensemble member and the black line is the expected function value. The optimized

solution (expected value) is indicated in the figure with the color black.

Observe that the objective value obtained using the Expected Value increased, i.e is worse, comparing with the standard problem optimization, presented in Figures 24. This behavior was hoped once the measure is conservative. However, the optimal function value in relation with the initial value had a good reduction, approximately among 49% and 63%, as showed in Table 5.

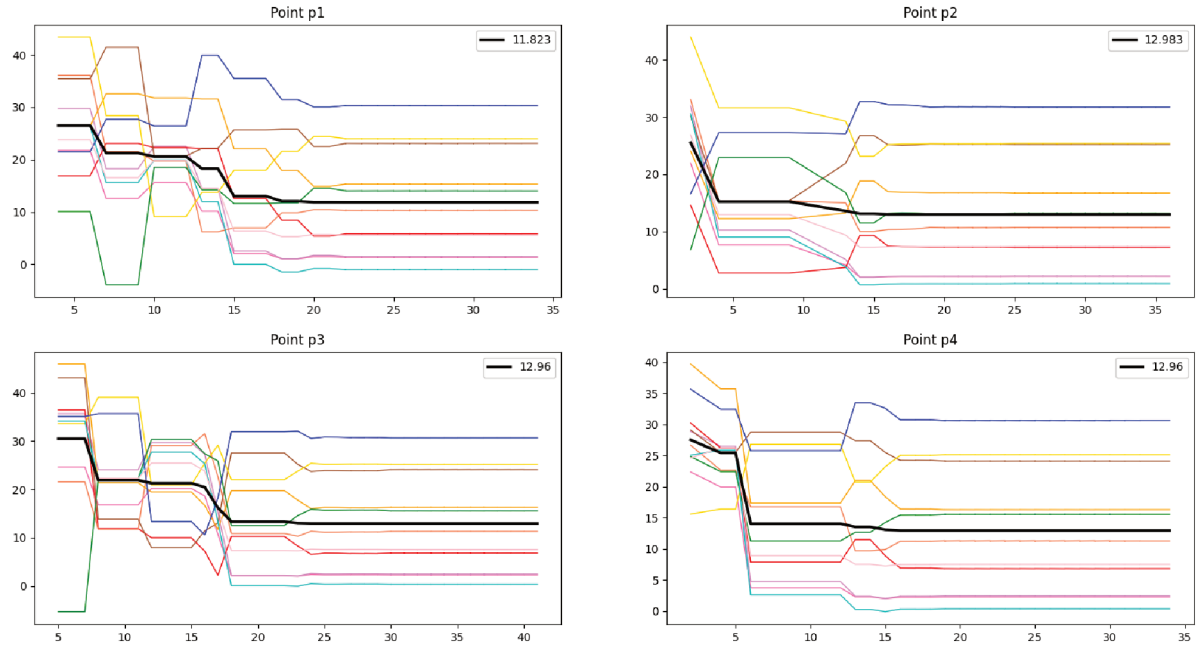
Table 5 – Comparison between the initial and final value of Rastrigin optimization

Init. Radius Size	Init. Point	IFV[3]	OFV	Reduction (%)
R_1	P_1	31.865	11.846	63
	P_2	25.512	12.983	49
	P_3	26.864	11.823	56
	P_4	27.544	10.844	60
R_2	P_1	31.865	11.823	63
	P_2	25.512	12.963	49
	P_3	31.865	12.960	59
	P_4	27.544	12.960	53



(a) Initial radius size R_1

Figure 25 – Evaluation of Rastrigin ensemble



(b) Initial radius size R_2

Figure 25 – Evaluation of Rastrigin ensemble

Table 6 presents the numerical results for the minimization of the standard and expected-value formulation. Because the expected value approach considers uncertainties in the reservoir model, the difference in final results can be disregarded. The algorithm has the capabilities to find the best solution for an objective combining set of several models and, in some cases, find the global optimum.

Table 6 – Comparison between the optimization results

		Standard Rastrigin		Rastrigin Ensemble	
Init. Radius Size	Init. Point	Solution Point	OFV	Solution Point	OFV
R_1	P_1	(0.00, 0.00)	0.00	(1.02, 0.00)	11.84
	P_2	(0.99, -0.99)	1.99	(1.02, -0.99)	12.98
	P_3	(-0.99, -0.99)	1.99	(-0.96, 0.00)	11.82
	P_4	(0.00, 0.00)	0.00	(0.03, 0.00)	10.84
R_2	P_1	(-0.99, 1.99)	4.98	(-0.96, 0.00)	11.82
	P_2	(-0.99, -0.99)	1.99	(1.02, -0.99)	12.98
	P_3	(0.99, 0.99)	1.99	(-0.96, -0.99)	12.96
	P_4	(-0.99, -0.99)	1.99	(-0.96, -0.99)	12.96

5.3 SUMMARY

The expected value approach was computationally validated using different well known analytic functions. Two experimental cases were created to test the capabilities of this methodology and, in this chapter, it was possible to demonstrate that the algorithm optimizes considering uncertainties in the model. The next chapter will extend the

results presented above, by considering the application of the trust-region derivative-free algorithm to a representative oil reservoir model with uncertainties.

6 SIMULATION ANALYSIS WITH RESERVOIR MODELS

This chapter addresses a case study regarding well control optimization of petroleum reservoirs under geological uncertainty. The study assesses the performance of the proposed ensemble trust-region algorithm, which is described in Chapter 4, in the control optimization of an oil reservoir under geological uncertainty. The uncertainty is represented with an ensemble of geological realizations, and the objective function is taken as the expected NPV for a production time of 6 years with one constant control during all the optimization time. The optimization variables in these type of problems are the well controls over the production time. For simplicity, the water rates of the injection wells are kept constant, and only the bottom hole pressures (BHPs) of the production wells are regarded as control variables.

The study is split into two parts. In the previous Chapter, an ensemble of analytic models was built to analyze the capability of the ensemble trust-region algorithm in explicit functions. The study shows the improvement achieved by the optimization algorithm of at least 40% with respect to the initial solution.

This Chapter presents the reservoir model and the geological uncertainties described as an ensemble of geological realizations. All simulations are performed with the simulator OPM Flow[1], and the reservoir model is depicted using the visualization tool ResInsight[2]. The optimization method is implemented in the open-source framework for field development optimization known as FieldOpt[3], and the simulations were executed in a Linux workstation equipped with an Intel Core i7-7500U CPU with 2.70GHz, 4 processors and 16GB RAM.

6.1 THE RESERVOIR MODEL

The reservoir model used in the simulations is illustrated in Figure 26. The model is a cartesian box split in 3600 cells of dimension $60 \times 60 \times 1$. All the wells are vertical wells, of which two are injectors, INJ2 and INJ2, and eight are producers, from PROD1 to PROD8. The well locations are presented in Table 7 and can be observed in Figure 27.

[1] OPM Flow is a open-source black-oil simulator avaiable in (OPM..., n.d.)

[2] ResInsight is a software to visualize reservoir simulators, found in (RESINSIGHT..., n.d.)

[3] FieldOpt is an open-source software of production optimization available on the GitHub platform (FIELDOPT..., n.d.), which was developed at NTNU/Norway.

Table 7 – Well locations

Name	x	y	z
INJ1	8	56	1
INJ2	51	44	1
PROD1	4	7	1
PROD2	11	7	1
PROD3	18	7	1
PROD4	25	7	1
PROD5	32	7	1
PROD6	39	7	1
PROD7	46	7	1
PROD8	53	7	1

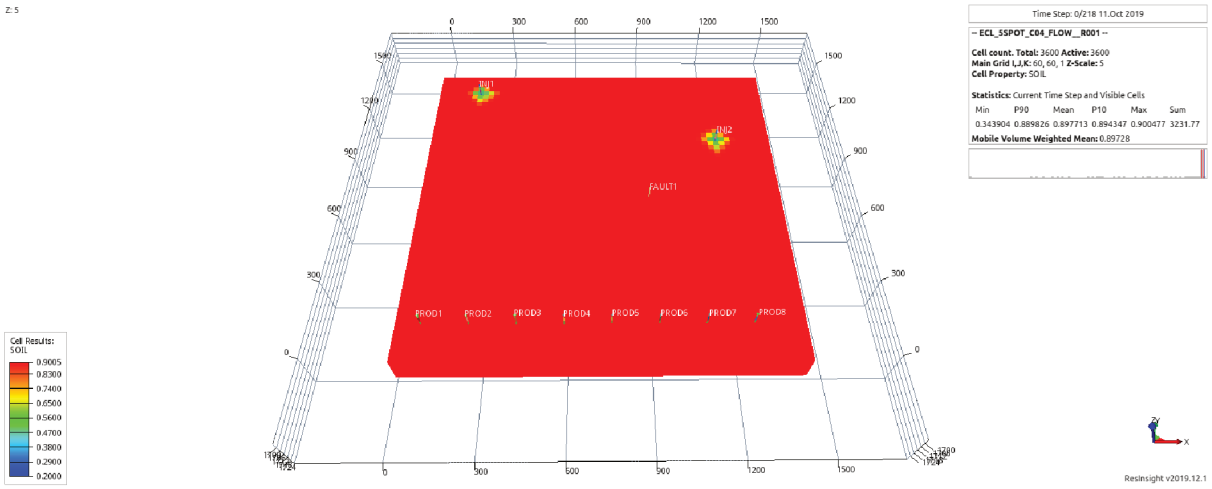


Figure 26 – Reservoir model depicted in ResInsight: Oil Saturation

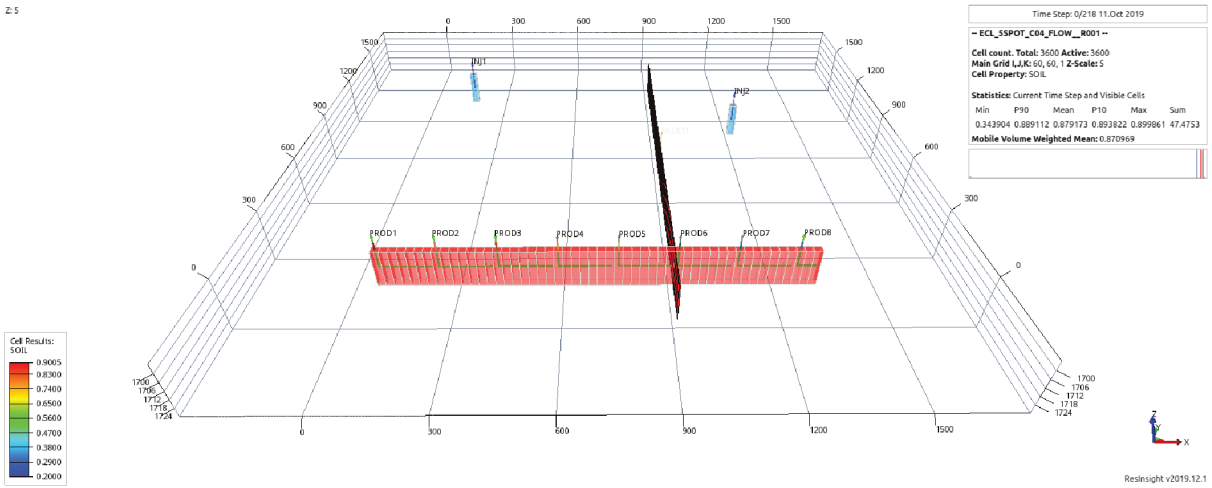


Figure 27 – Reservoir model depicted in ResInsight: well locations, and a geological fault.

The permeability map is presented in Figure 28. As described in the Oilfield Glossary of Schlumberger (OILFIELD. . . , n.d.), permeability means in reservoir produc-

tion the capability to transmit oil and water through the rocks. In the figure the color red means that the permeability is higher, which means, for instance, that the injection of well *INJ1* will reach the wells from *PROD4* to *PROD7* faster.

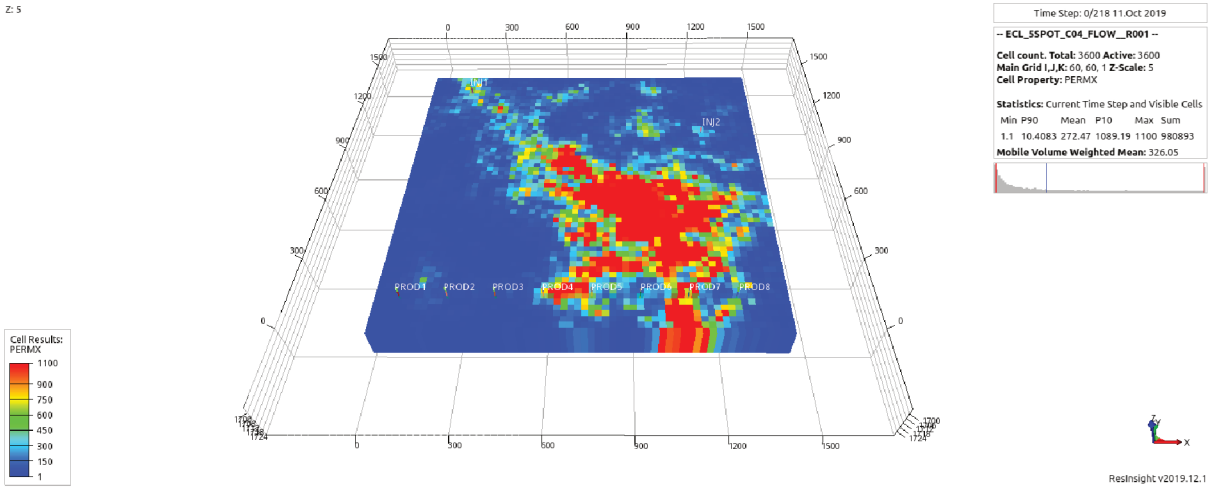


Figure 28 – ResInsight Reservoir: Permeability

In the ensemble, the parameter that changes was the permeability of the whole field, and the variation from one realization to the other is an increase of 10% in the overall field permeability.

A set of 10 geological realizations were generated. The permeability was changed 10% in the overall field from one realization to the other. These permeability changes represent the model uncertainties. The optimization will be applied over this ensemble of realizations and the results are presented in the next section.

6.2 TR-ENSEMBLE OPTIMIZATION RESULTS

In this section, the ensemble trust-region method will be applied to the ensemble of 10 realizations of the presented model. The expected NPV is the adopted objective function, and the BHPs of the producers are the optimization variables. The optimization method is the same used in the Chapter 5 for the analytical experiments.

In order to assess the effectiveness of the ensemble trust-region algorithm, we varied the initial parameters of the algorithm as well as the starting point. Two initial points are considered, in which the BHPs of the producers are set to 120 psi and 150 psi. The initial trust-region radius is the parameters which varied. Two radius sizes were considered, 25 and 50. Therefore, there are four cases to be evaluated indicated in the following Table 8. The BHPs of the injectors are are not optimizing variables, and they are kept fixed at 230 psi in all the cases.

Table 8 – Reservoir optimization cases

Case	Initial Point	TR Initial Radius Size
C_1	120 psi	25
C_2	120 psi	50
C_3	150 psi	25
C_4	150 psi	50

The optimization results obtained in the maximization of the expected NPV, formulated in Eq. (1), using the Trust-Region algorithm are presented in Figure 29. In the figure, the colored curves represent the evolution of the function values of each ensemble member, whereas the black curve is the evolution of the objective function, i.e. the expected NPV. Observe that the objective function had a good increment in all the cases. The actual value can be seen in Table 9, where OFV is used to abbreviate objective function value. The improvement over the initial expected NPV lies between 14.5% and 21.8%, being 17.6% on average. This improvement is significant and demonstrate the benefit of the proposed approach.

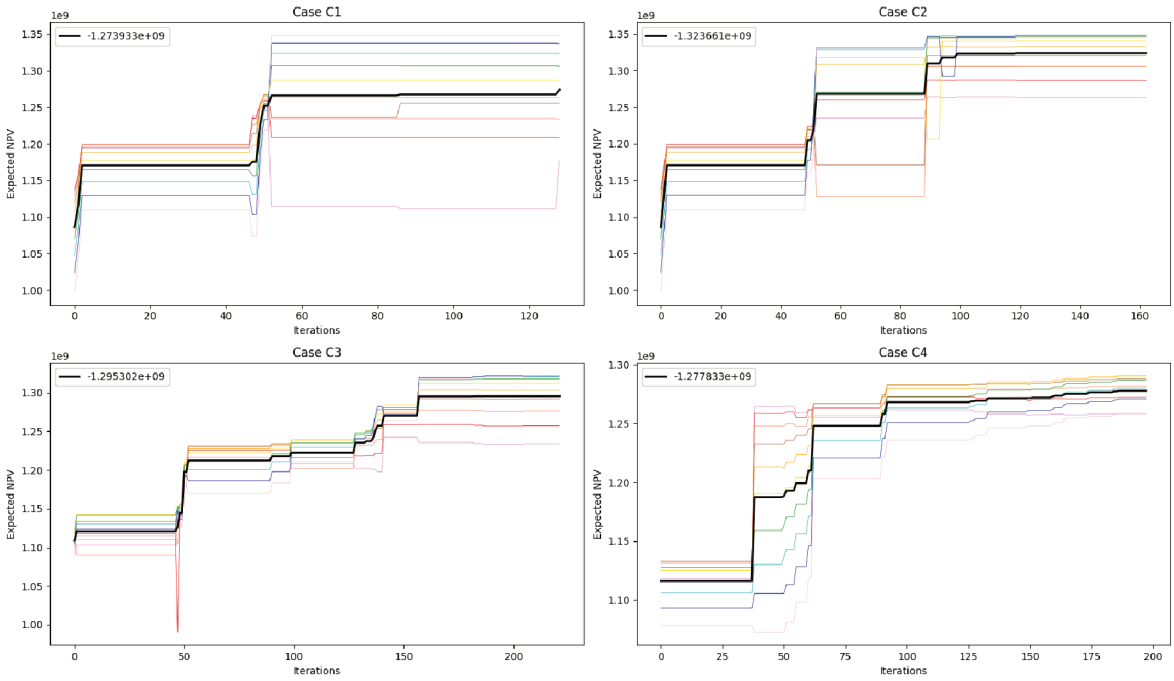


Figure 29 – Trust-Region reservoir results

Table 9 – Numerical results of TR optimization

Case	Initial OFV	Final OFV	Growth rate
C_1	1086043524	1273932712	17.3%
C_2	1086043524	1323660782	21.8%
C_3	1108698985	1295302470	16.8%
C_4	1115905205	1277832544	14.5%

The Figure 30 presents the BHP control at each iteration step. In the legend is indicated the producer that the BHP curve represents. Observe that any control reached the saturation pressure at 200 psi.

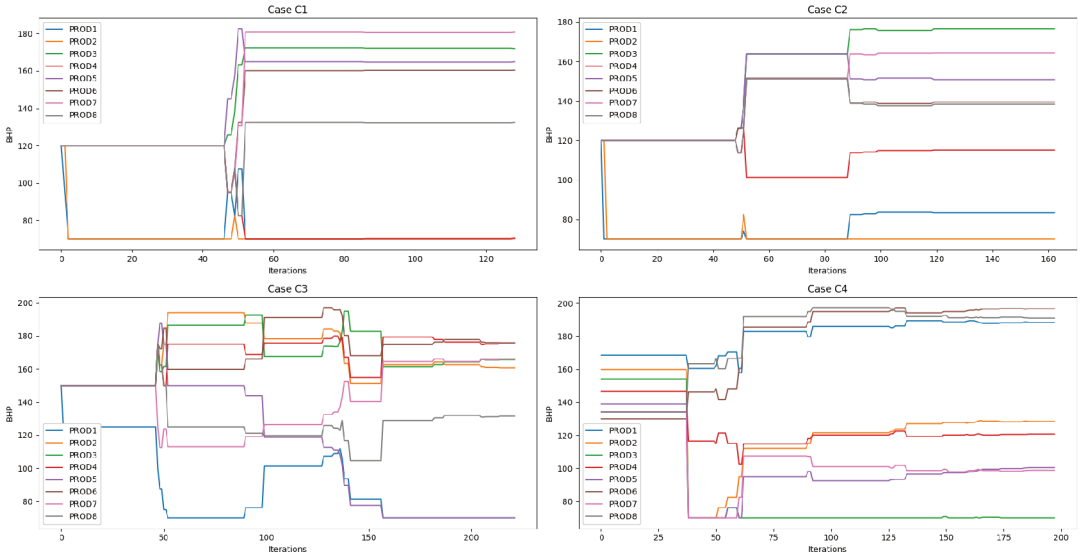


Figure 30 – Producers BHP

6.3 SUMMARY

In this Chapter the trust-region DFO method is assessed in an ensemble of synthetic reservoir models. In the first section the ensemble was presented using the visualization software Resinsight. Then, Section 6.2 addressed the results obtained in the robust well control optimization with the ensemble trust-region algorithm. The results showed reasonable improvements in the objective function using the proposed strategy.

7 FINAL CONSIDERATIONS

7.1 CONCLUSIONS

This works focused on the well control optimization problem considering geological uncertainty. Such problems have as control variables the bottom hole pressures (BHPs) of injectors and/or producers over time. The objective function is generally the net present value (NPV), which considers not only the total volume of produced oil, but also costs related to the production process.

The main challenge in the project consisted in extending the open source framework for field development optimization, FieldOpt, in order to optimize the reservoir performance uncertainty. Due the challenges present in gradient-based well control optimization, the proposed strategy uses a derivative-free method. Among the derivative-free methods, the Trust-Region method was chosen because its convergence properties.

The expected value was used to measure the performance over the ensemble of geological realizations. Although the use of expected value brings conservativeness to the optimization, the algorithm was capable of finding a solution which was 21.8% better than the starting point for a set of synthetic reservoir models.

The proposed methodology, the ensemble trust-region optimization algorithm, was demonstrated to be effective in well control optimization problems with geological uncertainty. The approach improved the expected value of the objective function in both analytical simulations and in synthetic reservoir models. The optimization presents good potential for further application in more complex industrial cases.

7.2 FUTURE WORKS

The proposed method was assessed in a set of synthetic reservoir models for the well control optimization problem. There are other interesting and challenging problems in field development optimization, such as the well placement optimization problem. This problem consists in optimization the trajectory of the wells for which there are not derivatives available, making the trust-region method a good candidate for such problems.

Another possible extension of this work is the use of different measures than the expected value. An interesting alternative can be found in (CHEN, Y., 2008). Chen et al. proposed an ensemble optimization method in which they use the stochastic gradient to solve field development optimization problems. An adaptation of this approach to derivative-free methods is a good direction for future research.

REFERENCES

AMERICA IPAA, Independent Petroleum Association of. Understanding the World Petroleum Association of America Market 2000. **Washington DC:IPAA**, v. 15, 2002.

ANDERSEN, J. R. **A derivative-free model-based trust-region optimization method**. 2018. Master's thesis – Norwegian University of Science and Technology.

ANN, M.; ANDREW, C.; KEVIN, W.; HARRY, F.; IAN, C.; TIM, M.; PETER, S. Recovery rates, enhanced oil recovery and technological limits. **The Royal Society**, v. 372, 2014.

AUDET, C.; DENNIS JR, J. Mesh adaptive direct search algorithms for constrained optimization. **SIAM Journal on Optimization**, v. 17, p. 188–217, 2006.

AUDET, C.; DENNIS, J.E. Jr. A pattern search filter method for nonlinear programming without derivatives. **Technical Report TR00-09, Department of Computational and Applied Mathematics. Rice University, Houston Texas**, 2000.

BAUMANN, E.; DALE, S.; BELLOUT, M. FieldOpt: A powerful and effective programming framework tailored for field development optimization. **Computers & Geosciences**, v. 135, 2020.

BP. Energy Outlook 2019 edition, 2019.

BROUWER, D. R. **Dynamic waterflood optimization with smart wells using optimal control theory**. 2004. Doctor thesis in geochemistry – Delft University of Technology.

CAPOLEI, A.; CHRISTIANSEN, L. H.; JØRGENSEN, J. B. A Novel Approach for Risk Minimization in Life-Cycle Oil Production Optimization. **Computer Aided Chemical Engineering**, n. 40, p. 157–162, 2017.

CAPOLEI, A.; SUWARTADI, E.; FOSS, B.; JØRGENSEN, J.B. Waterflooding optimization in uncertain geological scenarios. **Computational Geosciences**, v. 17, p. 991–1013, 2013.

CHEN, C.; WANG, Y.; LI, G.; REYNOLDS, A.C. Closed-loop reservoir management on the Brugge test case. **Computational Geosciences**, n. 14, p. 691–703, 2010.

CHEN, Y. **Efficient Ensemble based Reservoir Management**. 2008. PhD thesis – University of Oklahoma.

CHEN, Y.; D.S., Oliver; ZHANG, D. Efficient ensemble-based closed-loop production optimization. **SPE**, n. 14, p. 634–645, 2009.

CHEN, Y.; OLIVER, D.S. Ensemble-based closed-loop optimization applied to Brugge field. **SPE Reservoir Evaluation & Engineering**, n. 13, p. 56–71, 2010.

CONN, A.; SCHEINBERG, K.; VICENTE, L. **Introduction to Derivative-Free Optimization**. [S.l.]: SIAM, 2009.

FIELDOPT. [S.l.: s.n.]. `https://github.com/PetroleumCyberneticsGroup/FieldOpt`. Accessed: 2020-07-27.

GIULIANI, C. Thesis. Contributions to Derivative-free optimization: An exact penalty method and decompositions for distributed control. In: Florianopolis, SC: UFSC, 2019.

GRACE, R. D. Practical Considerations in Pressure Control Procedures in Field Operations. **Journal of Petroleum Technology**, v. 29, 1977.

HOOKE, R.; JEEVES, T. A. "Direct Search" Solution of Numerical and Statistical Problems. **Journal of the ACM**, v. 8, n. 2, 1961.

IEA, International Energy Agency. World Energy Outlook 2019, 2019.

JANSEN, J.D. Adjoint-based optimization of multi-phase flow through porous media - A review. **Computers & Fluids**, v. 46, p. 40–51, 2011.

JANSEN, J.D.; BROUWER, R.; DOUMA, S.G. Closed Loop Reservoir Management. **SPE Reservoir Simulation Symposium**, 2009.

JAPAN, Mathematical Society of. Encyclopedic Dictionary of Mathematics. In: [s.l.]: MIT Press, 1993.

KOLDA, T. G.; LEWIS, R. M.; TORCZON, V. J. Optimization by direct search: New perspectives on some classical and modern methods. **SIAM Review**, p. 385–482, 2003.

KOUROUNIS, D.; DURLOFSKY, L.J.; JANSEN, J.D.; AZIZ, K. Adjoint formulation and constraint handling for gradient-based optimization of compositional reservoir flow. **Computational Geosciences**, v. 18, p. 117–137, 2014.

KRISTOFFERSEN, B. S.; SILVA, T. L.; BELLOUT, M. C.; BERG, C. F. An Automatic Well Planner for Efficient Well Placement Optimization Under Geological Uncertainty. In: 17 th European Conference on the Mathematics of Oil Recovery. [S.l.]: ECMOR XVII, 2020. (EAGE).

MIDTTUN, L. S. **Reservoir Management under Uncertainty**. 2015. Master of Science in Cybernetics and Robotics – Norwegian University of Science and Technology, Norway.

NEFT, G. **Drilling high-tech wells**. [S.l.: s.n.]. `www.gazprom-neft.com/technologies/production/`. Accessed: 2020-07-26.

NELDER, J.; MEAD, R. A Simplex Method for Function Minimization. **The Computer Journal**, v. 7, n. 4, p. 308–313, 1965.

OILFIELD Glossary: Permeability. [S.l.: s.n.]. `www.glossary.oilfield.slb.com/en/Terms/p/permeability.aspx`. Accessed: 2020-07-19.

OLEA, R. A. **Geostatistical Glossary and Multilingual Dictionary**. [S.l.]: Oxford University Press, 1991.

OPM | Flow. [S.l.: s.n.]. `https://opm-project.org/?page_id=19`. Accessed: 2020-07-27.

RESINSIGHT | Post Processing of Reservoir Simulations. [S.l.: s.n.]. `https://resinsight.org/`. Accessed: 2020-07-27.

RIOS, L.; SAHINIDIS, N. Derivative-free optimization: A review of algorithms and comparison of software implementations. **Optimization and Engineering**, v. 56, n. 3, p. 1247–1293, 2013.

SILVA, T. L.; BELLOUT, M. C.; GIULIANI, C.; CAMPONOGARA, E.; PAVLOV, A. A Derivative-Free Trust-Region Algorithm for Well Control Optimization. In: 17 th

European Conference on the Mathematics of Oil Recovery. [S.l.]: ECMOR XVII, 2020. (EAGE).

SUWARTADI, E.; KROGSTAD, S.; FOSS, B. Nonlinear output constraints handling for production optimization of oil reservoirs. **Computational Geosciences**, v. 16, p. 499–517, 2011.

TORCZON, V. On the convergence of pattern search algorithms. **SIAM Journal on Optimization**, v. 7, p. 1–25, 1997.

VAN ESSEN, G.; VAN DEN HOF, P.; JANSEN, J.D. Hierarchical Long-Term and Short-Term Production Optimization. **SPE Journal**, v. 16, p. 191–199, 2011.

VAN ESSEN, G.M.; ZANDVLIET, M.J.; VAN DEN HOF, P.M.J.; BOSGRA, O.H.; JANSEN, J.D. Robust waterflooding optimazation of multiple geological scenarios. **SPE**, n. 14, p. 202–210, 2009.

WANG, C.; LI, G.; REYNOLDS, A.C. Production Optimization in Closed-Loop Reservoir Management. **SPE Journal**, v. 14, p. 506–523, 2010.

APPENDIX A – IMPLEMENTATION CODES

The codes were implemented in Matlab and applied in the Eq. (15), the results are presented during Chapter 3 as application examples to the fundamentals algorithms of derivative-free. This Appendix Chapter is dedicated to show the implementation codes to turns possible the reproduction of the presented examples.

$$\underset{x\in\mathbb{R}^2}{\text{minimize}}\ f(x_1, x_2) = x_1^2 - 4x_1 + x_2^2 - x_2 - x_1 x_2 \tag{15}$$

A.1 NELDER-MEADS

Fundamental Nelder-Meads method

The Nelder-Meads method, based on (CONN et al., 2009) is presented below.

```
n = 2; % function dimension
p = 1;
x = 2;
y = 0.5;
s = 0.5;
tol = 0.01;


x0 = 0.3;
y0 = 0.4;
x1 = 1.2;
y1 = 0.0;
x2 = 0.0;
y2 = 0.8;

%Initial condictions
S0 = {[x0,y0],[x1,y1],[x2,y2]};
F = [];
[S,F] = ordering(n,S0);
draw_triangle(S)

while(abs(F(1))-abs(F(2)))>tol
    xr = operation(p,S,n); %Reflect
    fr = fun(xr(1),xr(2));
    if fr<F(1)
        xe = operation(p*x,S,n); %Expand
        fe = fun(xe(1),xe(2));
        if fe<fr %Accepting xe how new min vertex
            [S,F] = accepting(xe,fe,S,F);
            draw_triangle(S)
        else
```

```
32              [S,F] = accepting(xr,fr,S,F);
33              draw_triangle(S)
34          end
35      elseif F(1)≤fr<F(n) % Accepting xr how medium between x1,xn
36          F(n+1) = F(n);
37          F(n) = fr;
38          S{n+1} = S{n};
39          S{n} = xr;
40          draw_triangle(S)
41      elseif F(n)≤fr<F(n+1)
42          xc = operation(p*y,S,n); %Outside contraction
43          fc = fun(xc(1),xc(2));
44          if fc<fr %Accepting xc
45              F(n+1) = fc;
46              S{n+1} = xc;
47              draw_triangle(S)
48          else %Compute Shrink
49              for i=2:n+1
50                  S{i} = S{1} + s*(S{i}-S{1});
51                  F(i) = fun(S{i}(1),S{i}(2));
52              end
53              draw_triangle(S)
54          end
55      else % Inside contraction
56          xc = operation(-y,S,n);
57          fc = fun(xc(1),xc(2));
58          if fc<F(n+1)
59              F(n+1) = fc;
60              S{n+1} = xc;
61              draw_triangle(S)
62          else % Compute Shrink
63              for i=2:n+1
64                  S{i} = S{1} + s*(S{i}-S{1});
65                  F(i) = fun(S{i}(1),S{i}(2));
66              end
67              draw_triangle(S)
68          end
69      end
70      [S,F] = ordering(n,S);
71
72  end
73
74  %Ploting the function
75  f=@(x,y) x.^2 - 4.*x + y.^2 - y - x.*y;
76  [X,Y] = meshgrid(-0.5:0.01:3.5);
77  z = f(X,Y);
78  [C,hContour] = contour(X,Y,z,-6.8:0.6:9,'ShowText','on');
```

```
79  %clabel(C,h,'FontSize',8,'Color','blue')
80  hold on
81  plot3(S{1}(1),S{1}(2),F(1),'-p','MarkerFaceColor','red')
82  fprintf('Solution: (x1,x2) -> (%2.4f,%2.4f), Function value: ...
        %2.4f',S{1}(1),S{1}(2),F(1))
83
84  % Update the contours immediately, and also whenever the contour is ...
        redrawn
85  updateContours(hContour);
86  addlistener(hContour, 'MarkedClean', @(h,e)updateContours(hContour));
87
88  function [s,f] = ordering(n,V0)
89      s = V0;
90      f = zeros(1,n+1);
91      for i=1:n+1
92          f(i) =  fun(V0{i}(1),V0{i}(2));
93      end
94      for i=1:n+1
95          if i>1 && f(i)<f(i-1)
96              f_aux = f(i);
97              f(i) = f(i-1);
98              f(i-1) = f_aux;
99              %Reorganizing vertex
100             s_aux = s{i};
101             s{i} = s{i-1};
102             s{i-1} = s_aux;
103         end
104         if i==n+1 && f(i-1)<f(i-2)
105             f_aux = f(i-1);
106             f(i-1) = f(i-2);
107             f(i-2) = f_aux;
108             %Reorganizing vertex
109             s_aux = s{i-1};
110             s{i-1} = s{i-2};
111             s{i-2} = s_aux;
112         end
113
114     end
115
116 end
117 function y = fun(x1,x2)
118    y = x1^2 - 4*x1 + x2^2 - x2 - x1*x2;
119 end
120 function d = operation(p,S,n)
121     sum = 0;
122     for i=1:n
123         sum = sum + S{i};
```

```
124     end
125     x_ = (1/n)*sum;
126     d = (1+p)*x_ - p*S{n+1};
127 end
128 function [s,f] = accepting(x_value,f_value,S,F)
129     s = S;
130     f = F;
131     f(3) = f(2);
132     f(2) = f(1);
133     f(1) = f_value;
134 
135     s{3} = s{2};
136     s{2} = s{1};
137     s{1} = x_value;
138 end
139 function t = draw_triangle(S)
140     hold on
141     t = plot([S{1}(1) S{2}(1) S{3}(1) S{1}(1)],[S{1}(2) S{2}(2) ...
            S{3}(2) S{1}(2)],'black');
142 end
143 %%%%%%%%%%%%%%%%%%% Only to draw better the level curve
144 function updateContours(hContour)
145     % Update the text label colors
146     drawnow;  % very important!
147     levels = hContour.LevelList;
148     labels = hContour.TextPrims;  % undocumented/unsupported
149     lines  = hContour.EdgePrims;  % undocumented/unsupported
150     for idx = 1 : numel(labels)
151         labelValue = str2double(labels(idx).String);
152         % avoid FP errors using eps
153         lineIdx = find(abs(levels-labelValue)<10*eps, 1);
154         % update the label color
155         labels(idx).ColorData = lines(lineIdx).ColorData;
156     end
157 end
```

A.2 COMPASS SEARCH

The Compass Search based in (CONN et al., 2009) was implemented in Matlab as showed as follows.

```
1 close all;
2 %Ploting the function
3 f=@(x,y) x.^2 - 4.*x + y.^2 - y - x.*y;
4 [X1,Y1] = meshgrid(1:0.01:4);
```

```
z = f(X1,Y1);
[T,hContour] = contour(X1,Y1,z,-6.8:0.6:9,'ShowText','on');

% Update the contours immediately, and also whenever the contour is redrawn
updateContours(hContour);
addlistener(hContour, 'MarkedClean', @(h,e)updateContours(hContour));

D = [1 0; 0 1; -1 0;0 -1];
a = 0.5; % Alpha
tol = 0.1;
X = [2 3];

while a>tol
    f = fun(X(1),X(2));
    P = X +a*D;
    plotSearchPoints(P,X,[0.65 0.65 0.65]);
    F = [fun(P(1,1),P(1,2)) fun(P(2,1),P(2,2)) fun(P(3,1),P(3,2)) fun(P(4,1),P(4,2))];
    f_under = f;
    for i=1:length(F)
        if F(i)<f
            f = F(i);
            X = [P(i,1) P(i,2)];
        end
    end
    if f_under == f
        a = a/2;
    end
end

plotSearchPoints(P,X,'red');

% Print solution
hold on
plot(X(1),X(2),'-p','MarkerFaceColor','red','MarkerSize',8)
fprintf('Solution: (x1,x2) -> (%2.4f,%2.4f), Function value: %2.4f \n',X(1),X(2),f)

function y = fun(x1,x2)
   y = x1^2 - 4*x1 + x2^2 - x2 - x1*x2;
end

function plotSearchPoints(P,X,color)
    hold on
    plot(X(1),X(2),'-o','MarkerFaceColor',color, ...
    'MarkerEdgeColor',color,'MarkerSize',4)
```

```
49      hold on
50      plot([X(1) P(1,1)], [X(2) P(1,2)],'Color',color)
51      hold on
52      plot(P(1,1),P(1,2),'-o','MarkerFaceColor',color, ...
53      'MarkerEdgeColor',color,'MarkerSize',4)
54      hold on
55      plot([X(1) P(2,1)], [X(2) P(2,2)],'Color',color)
56      hold on
57      plot(P(2,1),P(2,2),'-o','MarkerFaceColor',color, ...
58      'MarkerEdgeColor',color,'MarkerSize',4)
59      hold on
60      plot([X(1) P(3,1)], [X(2) P(3,2)],'Color',color)
61      hold on
62      plot(P(3,1),P(3,2),'-o','MarkerFaceColor',color, ...
63      'MarkerEdgeColor',color,'MarkerSize',4)
64      hold on
65      plot([X(1) P(4,1)], [X(2) P(4,2)],'Color', color)
66      hold on
67      plot(P(4,1),P(4,2),'-o','MarkerFaceColor',color, ...
68      'MarkerEdgeColor',color,'MarkerSize',4)
69  end
70
71  %%%%%%%%%%%%%%%%%%% Only to draw better the level curve
72  function updateContours(hContour)
73      % Update the text label colors
74      drawnow  % very important!
75      levels = hContour.LevelList;
76      labels = hContour.TextPrims;  % undocumented/unsupported
77      lines  = hContour.EdgePrims;  % undocumented/unsupported
78      for idx = 1 : numel(labels)
79          labelValue = str2double(labels(idx).String);
80          lineIdx = find(abs(levels-labelValue)<10*eps, 1);  % avoid ...
                FP errors using eps
81          labels(idx).ColorData = lines(lineIdx).ColorData;  % update ...
                the label color
82          %labels(idx).Font.Size = 8;                        % update ...
                the label font size
83      end
84      drawnow  % optional
85  end
```

APPENDIX B – ENSEMBLE FUNCTIONS

B.1 ROSENBROCK ENSEMBLE ELEMENTS

Ensemble functions

Each function showed in the table of parameters 2 are mathematically presented as follows in equations (16)–(25).

$$z_1 = \sum_{i=1}^{n-1} \left[95(x_{i+1} + 4 - (x_i - 0.8)^2)^2 + (x_i - 0.8)^2 \right] \tag{16}$$

$$z_2 = \sum_{i=1}^{n-1} \left[97(x_{i+1} + 0.3 - (x_i - 0.4)^2)^2 + (x_i - 1)^2 \right] \tag{17}$$

$$z_3 = \sum_{i=1}^{n-1} \left[103(x_{i+1} + 0.3 - (x_i + 0.4)^2)^2 + (x_i - 1.2)^2 + 1 \right] \tag{18}$$

$$z_4 = \sum_{i=1}^{n-1} \left[94(x_{i+1} - 1.8 - (x_i - 0.3)^2)^2 + (x_i + 0.8)^2 \right] \tag{19}$$

$$z_5 = \sum_{i=1}^{n-1} \left[98(x_{i+1} - (x_i + 0.7)^2)^2 + (x_i + 0.3)^2 \right] \tag{20}$$

$$z_6 = \sum_{i=1}^{n-1} \left[95(x_{i+1} + 1.8 - (x_i - 0.5)^2)^2 + (x_i - 1)^2 \right] \tag{21}$$

$$z_7 = \sum_{i=1}^{n-1} \left[106(x_{i+1} - (x_i - 0.7)^2)^2 + (x_i - 0.2)^2 \right] \tag{22}$$

$$z_8 = \sum_{i=1}^{n-1} \left[96(x_{i+1} + 4 - (x_i)^2)^2 + (x_i - 1.3)^2 \right] \tag{23}$$

$$z_9 = \sum_{i=1}^{n-1} \left[105(x_{i+1} - 2 - (x_i)^2)^2 + (x_i + 0.7)^2 \right] \tag{24}$$

$$z_{10} = \sum_{i=1}^{n-1} \left[90(x_{i+1} + 0.6 - (x_i - 0.2)^2)^2 + (x_i - 1)^2 \right] \tag{25}$$

B.2 RASTRIGIN ENSEMBLE ELEMENTS

Ensemble functions

Each function showed in the table of parameters 4 are mathematically presented as follows in equations (26)–(35).

$$z_1 = 15n + x^2 - 15cos(2\pi x) + y^2 - 15cos(2\pi y) \quad (26)$$

$$z_2 = 10n + (x - 0.2)^2 - 10cos(2\pi(x - 0.2)) + y^2 - 10cos(2\pi y) \quad (27)$$

$$z_3 = 10n + x^2 - 10cos(2\pi x + 0.4) + (y + 0.2)^2 - 10cos(2\pi(y + 0.2) + 0.5) \quad (28)$$

$$z_4 = 11n + x^2 - 10cos(2\pi x + 0.6) + (y + 0.15)^2 - 10cos(2\pi(y + 0.15)) \quad (29)$$

$$z_5 = 15n + (x - 0.2)^2 - 10cos(2\pi(x - 0.2)) + y^2 - 0.5 - 10cos(2\pi y) \quad (30)$$

$$z_6 = 14n + (x + 0.3)^2 - 10cos(2\pi(x + 0.3)) + (y - 0.1)^2 - 10cos(2\pi y) \quad (31)$$

$$z_7 = 5n + (x - 0.4)^2 + 10cos(2\pi(x - 0.4)) + (y - 0.4)^2 - 10cos(2\pi(y - 0.4)) \quad (32)$$

$$z_8 = 14n + (x + 0.2)^2 - 15cos(2\pi x) + (y - 0.2)^2 - 0.2 - 15cos(2\pi y) \quad (33)$$

$$z_9 = 13n + x^2 - 10cos(2\pi x + 0.6) + (y + 0.1)^2 - 10cos(2\pi(y - 0.2)) \quad (34)$$

$$z_{10} = 12n + (x - 0.6)^2 - 10cos(2\pi x + 1) + (y - 0.5)^2 - 10cos(2\pi y) \quad (35)$$