EFFICIENT SOLUTION OF OPTIMIZATION PROBLEMS WITH CONSTRAINTS AND/OR COST FUNCTIONS EXPENSIVE TO EVALUATE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AHMET GÖKHAN KURTDERE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2009

Approval of the thesis:

EFFICIENT SOLUTION OF OPTIMIZATION PROBLEMS WITH CONSTRAINTS AND/OR COST FUNCTIONS EXPENSIVE TO EVALUATE

submitted by **AHMET GÖKHAN KURTDERE** in partial fulfillment of the requirements for the degree of Master of Science **in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Kemal Leblebicioğlu _____
Supervisor, **Electrical and Electronics Engineering., METU**

Examining Committee Members:

Prof. Dr. Mustafa Kuzuoğlu _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Kemal Leblebicioğlu _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Kemal Özgören _____
Mechanical Engineering Dept., METU

Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Engineering Dept., METU

Murat Akgül, PhD. in EE _____
İLTAREN, TÜBİTAK

Date : 11.12.2009

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Ahmet Gökhan KURTDERE

Signature :

ABSTRACT

EFFICIENT SOLUTION OF OPTIMIZATION PROBLEMS WITH CONSTRAINTS AND/OR COST FUNCTIONS EXPENSIVE TO EVALUATE

Kurtdere, Ahmet Gökhan
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Kemal Leblebicioğlu

December 2009, 85 pages

There are many optimization problems motivated by engineering applications, whose constraints and/or cost functions are computationally expensive to evaluate. What is more derivative information is usually not available or available at a considerable cost. For that reason, classical optimization methods, based on derivatives, are not applicable. This study presents a framework based on available methods in literature to overcome this important problem. First, a penalized model is constructed where the violation of the constraints are added to the cost function. The model is optimized with help of stochastic approximation algorithms until a point satisfying the constraints is obtained. Then, a sample point set satisfying the constraints is obtained by taking advantage of direct search algorithms based sampling strategies. In this context, two search direction estimation methods, convex hull based and estimated radius of curvature of the sample point set based methods can be applicable. Point set is used to create a barrier which imposes a large cost for points near to the boundary. The aim is to obtain convergence to local optima using the most promising direction

with help of direct search methods. As regards to the evaluation of the cost function there are two directions to follow: a-) Gradient-based methods, b-) Non-gradient methods. In gradient-based methods, the gradient is approximated using the so-called stochastic approximation algorithms. In the latter case, direct search algorithms based sampling strategy is realized. This study is concluded by using all these ideas in the solution of complicated test problems where the cost and the constraint functions are costly to evaluate.

Keywords: Optimization, Direct Search Methods, Response Surface, Stochastic Approximation.

ÖZ

KISITLARI VE GİDER FONKSİYONLARI HESAPLAMASI PAHALI OLAN OPTİMİZASYON PROBLEMLERİNİN VERİMLİ ÇÖZÜMLERİ

Kurtdere, Ahmet Gökhan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Kemal Leblebicioğlu

Aralık 2009, 85 sayfa

Mühendislik uygulamalarında kısıt ve gider fonksiyonları hesaplaması pahalı olan birçok optimizasyon problemi mevcuttur. Bununda ötesinde bu fonksiyonların türevleri elde edilebilir değildir yada elde edilme maliyetleri yüksektir. Bu sebeple türev tabanlı klasik optimizasyon yöntemleri uygulanamayabilir. Bu çalışmada bu önemli problemin çözümüne yönelik literatürde yer alan mevcut yöntemler dahilinde bir çatı sunulmuştur. İlk olarak kısıtların ihlalinin maliyet fonksiyonuna eklenmesiyle bir ceza modeli oluşturulur. Bu model, rasgele yaklaşma algoritmaları kullanılarak kısıtların sağladığı ilk noktaya kadar optimize edilir. Sonra, doğrudan arama metodları tabanlı örnekleme stratejileri ile kısıtları sağlayan örnek noktalar kümesi elde edilir. Bu kapsamda, arama yönü tahmini için, dışbükey kabuk ve nokta kümesinin eğrilik yarıçapını kullanan iki yöntem kullanılabilir. Örnek noktalar kümesi sınıra yakın bölgelerde yüksek maliyet veren bir sınır oluşturmak için kullanılır. Burada amaç, doğrudan arama yöntemlerinden faydalanarak yerel optimum noktaya yakınsamaktır. Gider fonksiyonunun değer tahmini konusuna gelince, aslında izlenecek iki yön vardır: a-) Gradyan tabanlı yöntemler, b-) Gradyan

tabanlı olmayan yöntemler. Gradyan tabanlı yöntemlerde, gradyan rasgele yaklaşma algoritmaları kullanılarak hesaplanabilir. Diğer durumda ise doğrudan arama yöntemleri tabanlı örnekleme algoritmaları ile gider fonksiyonu değer tahmini gerçekleştirilebilir. Bu çalışma tüm bu fikirlerin gider ve kısıt fonksiyonları hesaplaması pahalı olan karmaşık test problemlerin çözümünde kullanılmasıyla sonlandırılmıştır.

Anahtar Kelimeler: Optimizasyon, Doğrudan Arama Yontemleri, Tepki Yüzeyi, Rastlantısal Yaklaşım.

To My Beloved Family…

ACKNOWLEDGEMENTS

I would like to express my appreciation to my supervisor Prof. Dr. Kemal Leblebicioğlu for his wisdom and guidance throughout this study and I would like to thank my family for their love, encouragement and full support.

Moreover, I would like to thank my office and team at İLTAREN Research and Development Group, especially my team leader Dr. Murat Akgül for his support, guidance and understanding.

I would also thank to Dr. Murat Şamil Aslan for his suggestions and comments.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

OLS	Orthogonal Least Squares
RBF	Radial Basis Functions
SPSA	Simultaneous Perturbation Stochastic Approximation
FDSA	Finite Differences Stochastic Approximation
PCA	Principal Component Analysis
MLS	Moving Least Square
LS RBF	Least Squares Radial Basis Function
SA	Stochastic Approximation

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION

It is almost impossible to have the exact analytical solutions for most systems, with increasing complexities in the engineering applications. Then, utilization of modeling methods may be advantageous with the help of modern supercomputers and parallel computing. In these situations, generally an approximate black box model is developed for simulation or computational purposes. The model is called as a “black box” model since closed-form expression and the derivative information are not available or available at a considerable cost. Moreover, each evaluation of the black box model may include a random noise. For many large-scale complex systems, one evaluation of such a black box model which generates a single or a set of output for a given set of input variables has usually high computational cost and/or time consuming.

Although, the developed approximate model enables the designer to simulate the performance of the original system, the problem of finding a setting for possibly a large number of design variables still exists. [42] Then a crucial question arises as how to find the best possible setting requiring minimum number of evaluations. [42]

Specification of the appropriate values for the variables of these kinds of engineering systems is a well studied problem considered in the context of design optimization concept. During design optimization, various numerical techniques can be applied on the mathematical model of a system to provide some improvement on the performance or reliability of the system [43]. In this study, an efficient iterative way for finding a local optimum setting of these types of black box design optimization problems with the help of available methods in the literature is presented.

Utilization of traditional quasi-Newton methods to solve the problem is eliminated first because derivative information is unavailable or hard to obtain. Even if it was available quasi-Newton methods might be poor choices because they are adversely affected by function inaccuracies [8].

Local search, tabu search, simulated annealing are some other approaches to the design optimization problem generally considered in the field of discrete-event simulation [7]. These methods assume that one simulation run or one evaluation of the function can be performed quickly. Moreover, they only deal with (simple) constraints on the design parameters.

Derivative-free optimization algorithms are another approach for solving optimization problems which do not try to estimate or calculate the gradients [15]. These are heuristic methods extensively used to solve design optimization problems, called direct search methods. It is observed that these methods are relatively insensitive to inaccuracies in the cost and/or the constraint functions [1], [2], [3], [4].

Besides derivative free methods, Keifer–Wolfowitz method [5], [6] is an approach from the area of stochastic approximation. The gradient estimation with this method requires the evaluation of the cost function at points $x \pm \alpha_k e_i$, where e_i represents the i^{th} unit coordinate vector, $i = 1,2,\ldots n$ and α_k is a constant depending on the iteration. Then algorithm takes a step of length γ_k considering the gradient direction.

In noisy cases, allowing γ_k and α_k tend to zero very slowly, provides better convergence.

In this context, some approximate gradient based and gradient free optimization algorithms are examined. Simultaneous perturbation and finite differences based stochastic approximation algorithms that require approximation of the gradient is utilized and some direct search algorithms which attempts to minimize a scalar-valued nonlinear function of n real variables using only function evaluations without any explicit or implicit derivative information is considered. (A type of simpex based algorithm, Nelder-Mead simplex algorithm, a pattern search type algorithm, Hooke and Jeeves pattern search method and a random search algorithm, one at a time search method)

In some situations, representation of an approximate local surface model, that is both inexpensive to evaluate and accurate, capturing the underlying relationship between input and output may be desirable. Thus, it can be used to predict the output for some future observations or to estimate the derivatives at specific locations. For that reason, a surface representation concept is also considered in the scope of this study. Radial Basis Functions (RBF) are utilized as interpolants and the Orthogonal Least Squares (OLS) algorithm is preferred for center selection procedure.

This study defines an iterative optimization methodology utilizing direct search methods and stochastic approximation algorithms for solving optimization problems with expensive to evaluate cost functions and/or constraints which is given more detailed in chapter 2.

1.2. PROBLEM STATEMENT

As it is defined in reference [42], optimization problems with cost and constraint functions not known explicitly are common in black-box design optimization. The main characteristic of these types of problems, at least for an important part, is that there is no derivative information available for the responses. This situation generally

comes into being when dealing with simulation models. A preset design parameter is fed to black box model, most probably a simulation tool, and output response parameter set is returned according to unknown input-output relationship. Moreover, in many applications, black-box models require high computation cost or they are time consuming; so developed algorithm should not run the model an unnecessary amount of times.

The aim of the optimization problem is to minimize or maximize underlying function of the design parameters and responses, subject to given set of constraints that may be costly to evaluate or not (i.e., a constrained optimization problem) [42].

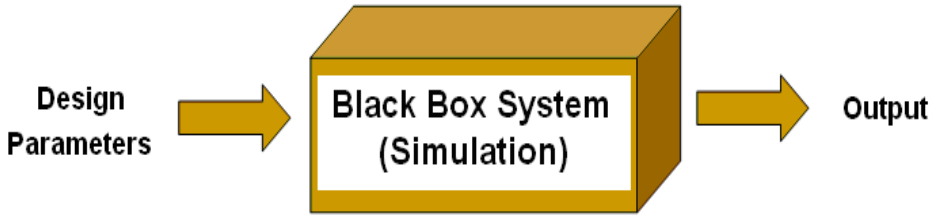

Figure 1.1: Black-box model

Generally a constrained optimization problem P can be stated as a nonlinear optimization problem of the following form;

$$\begin{aligned} P: \max\ & f(\bar{\theta})\ ,\ \bar{\theta} = (\theta_1, \theta_2, ..., \theta_d)^T \in F \subset S \subset R^d \\ & \textit{subject to} \\ & g_i(\bar{\theta}) \le 0 \quad i = 1, 2, ..., q \\ & h_j(\bar{\theta}) = 0 \quad j = q+1, q+2, ..., m \end{aligned} \tag{1.1}$$

whose feasible region, denoted by

$$F = \{\bar{\theta} \in R^d \mid g_i(\bar{\theta}) \le 0, i = 1, ..., q,\ h_j(\bar{\theta}) = 0,\ j = q+1, ..., m\} \tag{1.2}$$

where $\bar{\theta} = (\theta_1, \theta_2, ..., \theta_n)^T$ and S represent the vector of solutions and search space, respectively. There are q inequality and $m-q$ equality constraints. $f(\bar{\theta})$ is generally called the criterion function or cost function. Cost function and constraints can be linear or nonlinear. $\bar{\theta}$, satisfying all the constraints is a feasible solution of the problem. All of the feasible solutions constitute the feasible region. Additionally, the equality constraints $h_j(\bar{\theta}) = 0$, $j = q+1, q+2, ..., m$ can be converted to inequality constraints $h_j(\bar{\theta}) \leq 0$, $-h_j(\bar{\theta}) \leq 0$. So the problem can be expressed with only inequality constraints.

The user in these cases almost usually forced to construct a penalized model, where the violation of the constraints is added to the cost function; thereby enforcing the satisfaction of constraints [38]. In this way, an optimization problem with constraints can be transformed into an unconstrained form.

Barrier and penalty methods are designed to solve these types of problems. In a penalty method, the feasible region of the problem P stated as given in Equation 1.1 is expanded from feasible region F to whole space R^d , but a large cost or "penalty" is added to the cost function for the points that lie outside of the original feasible region F [33].

In a barrier method, we start with an assumption that a feasible point is given and a large cost is added to the cost function on feasible points that ever get closer to the boundary of F . That means, a barrier that prevents exiting the out of the feasible region is created [33].

1.3. OVERVIEW OF THE ALGORITHM

The algorithm presented in this study is based on direct search methods and stochastic approximation algorithms. Penalty and barrier methods are also utilized to handle the hard-to-evaluate constraints of the problem.

The proposed method has four main steps:

- ***Step1*:** Starting from an arbitrary initial guess, a point P satisfying the constraints is obtained. Stochastic approximation methods are employed for this part of the proposed method. A penalized model is first constructed to handle the constraints, and then this model is minimized or maximized until a point satisfying the constraints is obtained. The problem can then be considered as an unconstrained optimization problem from this point on.
- ***Step2:*** New sample points in the neighborhood of the point P are generated with the help of search direction estimation methods. Search direction estimation is critical; good estimates for the direction require smaller number of function evaluations. The obtained points have to be checked if they are satisfying the constraints or not. Points not satisfying the constraints have to be replaced by points in the feasible region. This can be achieved by the way employed in Step1. At the end of this step, a sample point set satisfying the constraints is obtained. This sample set is used for the procedure of barrier functions, employed in the next step.
- ***Step3:*** A barrier function model is constructed and one of the direct search algorithms executed for that model. The aim is to obtain approximately a local minimizer or maximizer of the problem. The point set obtained in the previous step is used as a barrier and the direction of iterations is forced to a better local optima. This procedure provides a more robust convergence. Penalty and barrier methods and some direct search methods are utilized in this step of the proposed algorithm.
- ***Step4:*** When achieving a satisfactory improvement, new constraints are developed; a new penalized model is developed and steps 2, 3 and 4 are repeated, respectively. The constraints constructed here are hypothetical; they are not original constraints of the problem. But original constraints of the problem also have to be checked. This procedure is continued until one of the stopping criteria is met. (Exceeding allowable number of function evaluations or attaining a desirable improvement for the response value, etc.).

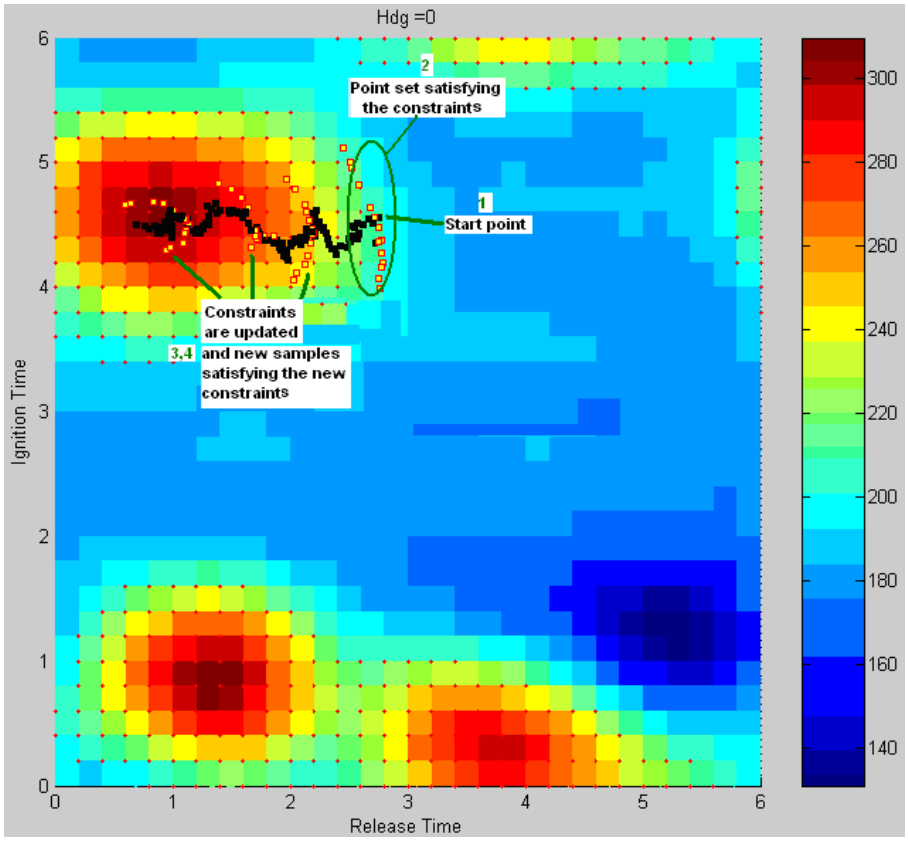

Figure 1.2: The steps of the algorithm

As shown in Figure 1.2 , starting from an initial point (marked as 1), a point set satisfying the constraints is obtained (marked as 2). While converging to local maximum iteratively, four hypothetical constrained surfaces have also determined by the algorithm and sample points on these constrait surfaces are obtained. These four imaginary constrait surfaces are also seen in Figure 1.2 (marked as 3 and 4).

First step of the method is, starting from an arbitrary initial point, approaching to a new point which satisfies the constraints of the problem. Penalized model is constructed to handle the constraints. For this model, simultaneous perturbation stochastic approximation and finite difference stochastic approximation algorithms are executed for gradient evaluation until a point $\bar{\theta}^*$ satisfying the constraints is obtained. For this point $\bar{\theta}^*$, penalized and unpenalized cost functions yields the same

constant value c. ($f(\bar{\theta}^*) = f_p(\bar{\theta}^*) = c$). The set of points that has the same function value $f_p(.) = c$, is used as an artificial constraint surface. In this construction, penalty functions are differentiable and strictly increasing.

Next step is to increase the number of samples in the neighborhood of the point obtained in first step. Each obtained point has to satisfy constraints, in other words, each obtained point has to be on the "artificial" constrained surface $f_p(.) = c$. For that purpose, some direct search methods (Nelder-Mead simplex method, Hooke-Jeeves pattern search and a random search method, one at a time search) are utilized. Two search direction estimation methods are proposed for that part of the algorithm. In the first one; sampling density is determined with respect to the curvature of the local points set. High curvature parts of the point set are determined. These parts have to be sampled more densely opposed to low curvature parts. In this way no expensive evaluations are wasted on parts of less importance. The second one utilizes a convex hull representation. New sampling locations are determined by using the vertices of this convex hull. These sampling methods are discussed in detail in chapter 2.

In the third step, a barrier function model is constructed using the point set obtained in the previous step. This point set used as a barrier and forces the direction of ongoing iterations to be in the feasible region as well as towards better local optima. Direct search algorithms are utilized for execution of the cost function.

Last step is updating the constraints to a better value and obtaining new points satisfying these new constraints. New sample points from new constraint surfaces contribute the convergence towards better local optima. Algorithm stops if an acceptable improvement achieved or the number of function evaluations exceeds the allowed limit.

1.4. OUTLINE OF THE THESIS

This study has been composed of five chapters. The first chapter consists of the introduction, problem statement and overview of the proposed algorithm parts. Furthermore, outline of the thesis has been given in this chapter. In chapter two, theoretical background of algorithms for optimization problems whose constraints and/or cost functions are expensive to evaluate are explained. This chapter also covers details of the proposed algorithm. Surface representation strategy is given in the third chapter. Implementation results of the algorithm to three different special test problems are given in chapter four. Finally, chapter five contains the conclusion part and future work about the study. References and appendices are given at the end of the thesis.

CHAPTER 2

SOLVING OPTIMIZATION PROBLEMS WHOSE COST FUNCTIONS AND/OR CONSTRAINTS ARE EXPENSIVE TO EVALUATE

Design optimization is a field of engineering which employs optimization methods to solve the problems. Literature in design optimization has investigated optimization methods based on simulated annealing, genetic algorithms and gradient based techniques. These algorithms are generally not preferred in the solution of computationally expensive, complex problems because they require large number of function evaluations or measurements. Another way of solving optimization problems is utilizing derivative free methods which do not try to estimate or calculate the gradients. These are heuristic and simple methods called direct search methods [15].

As it is stated in [44], direct search methods for unconstrained optimization problems have been more popular for the past few years. Since these methods do not make gradient estimates and involve relatively few function evaluations, they are widely employed for optimization of costly functions. The most commonly used one in this class is the Nelder-Mead method. This method operates using the repeated operations of reflection, expansion, and contraction applied to a simplex of $n+1$ points in R^n. This method was developed more than 30 years ago; however some extended

versions of this method is still the most commonly applied method for optimization when each function evaluation requires a separate experiment (i.e., very costly to evaluate).

In 1961, Robert Hooke and T.A. Jeeves developed a method for optimization and introduced “direct search” term. They presented the following description of direct search in the introduction of their paper [10]:

“We use the phrase “direct search” to describe sequential examination of trial solutions involving comparison of each trial solution with the “best” obtained up to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be. The phrase implies our preference, based on experience, for straightforward search strategies which employ no techniques of classical analysis except where there is a demonstrable advantage in doing so.“

Direct search methods only rely on the values of the objective function. They do not calculate or estimate the value of any derivatives at any point. So such methods are also called “derivative-free” methods or “zero-order methods” [15].

In addition to derivative free methods, another approach originated from the so-called Stochastic Approximation (SA) area is the Keifer–Wolfowitz method [5], [6]. The SA methods essentially approximate the gradients, where they can be estimated either numerically or analytically. The Kiefer-Wolfowitz [12] algorithm uses the finite difference gradient approximation to estimate gradients. Spall [13], [14] proposed the Simultaneous Perturbation Stochastic Approximation (SPSA) method for gradient estimation, which requires only two function evaluations per estimation, regardless of the dimension of the problem.

In this chapter, we begin by describing the family of underlying optimization algorithms on which the proposed algorithm is based. The direct search methods and stochastic approximation methods which are the two main branches of stochastic optimization are considered in this chapter. In this scope, simultaneous perturbation gradient approximation based stochastic approximation algorithm and finite

difference gradient approximation based stochastic approximation algorithm are discussed and direct search algorithms: simplex based algorithms (Nelder-Mead simplex method), pattern search (Hooke and Jeeves pattern search method) and random search (one at a time search) are also considered. Also the proposed algorithm is formally defined in this chapter.

2.1. STOCHASTIC APPROXIMATION

Stochastic approximation is a popular stochastic optimization technique. In particular, it may be called as "gradient-free" SA sometimes, because it only requires approximation of the gradient information [45]. Consider the problem given below.

$$g(\bar{\theta}) \equiv \frac{\partial f(\bar{\theta})}{\partial \bar{\theta}} \tag{2.1}$$

for a differentiable function $f : R^n \to R$. When direct evaluations of g are not costly, there are, of course, many approaches which can be used as a solution method (e.g., steepest descent). In the case where the exact functional relationship between the function value and the parameters, $\bar{\theta}$, is not known and the function is evaluated by measurements on the system (or by other means, such as simulation) an approximation to $g(\bar{\theta})$ is used [45] (The well-known form of SA called the Kiefer-Wolfowitz type is just an example).

The generally used form of this type of SA recursion is:

$$\bar{\theta}_{k+1} = \bar{\theta}_k - a_k \hat{g}_k(\bar{\theta}_k) \tag{2.2}$$

where $\hat{g}_k(\bar{\theta})$ refers to an approximation (at the $k\,th$ step of the recursion) of the gradient $g(\bar{\theta})$, and the sequence $\{a_k\}$ consists of positive scalars which decreases to zero in the standard implementation [45].

For the approximation of the gradient, $\hat{g}(\bar{\theta})$, in the usual version of this algorithm, based on the "finite difference" method, is discussed in the next section. Another

general version of stochastic approximation is obtained by using a "simultaneous perturbation" gradient approximation.

2.1.1. FINITE DIFFERENCE STOCHASTIC APPROXIMATION (FDSA)

The important part of Equation (2.2) is the gradient approximation $\hat{g}(\bar{\theta}_k)$. Traditionally, the finite difference method is used for constructing the approximation. Equation (2.2) is referred to as the finite difference stochastic approximation (FDSA) when this method is used.

In finite difference based gradient approximation method, each component of the parameter $\bar{\theta} = (\theta_1, \theta_2, ..., \theta_n)^T$ is perturbed one at a time and corresponding measurements $y(.)$ are obtained. Next, each element of the gradient estimate is formed by differencing the corresponding output values and then dividing by a difference interval [46]. This type is the standard Kiefer-Wolfowitz approach to approximating gradient components and is originated from the definition of a gradient as a vector of p partial derivatives [46].

If gradient approximation is one sided, only $y(\hat{\theta}_k)$ and $y(\hat{\theta}_k + perturbation)$ measurements are needed, while two-sided approximations involve measurements of the form $y(\hat{\theta}_k \pm perturbation)$. The two-sided FD approximation to be used with Equation (2.2) is;

$$\hat{\nabla} f(\hat{\theta}_n) = \begin{bmatrix} \dfrac{f(\hat{\theta}_n + c_n\xi_1) - f(\hat{\theta}_n - c_n\xi_1)}{2c_k} \\ \vdots \\ \dfrac{f(\hat{\theta}_n + c_n\xi_p) - f(\hat{\theta}_n - c_n\xi_p)}{2c_k} \end{bmatrix} \tag{2.3}$$

where ξ_i denotes a vector with a 1 in the ith place and 0's elsewhere and $c_n > 0$ defines the difference magnitude. The $\{a_k, c_k\}$ pair are the gains (or gain sequences) for the FDSA algorithm.

2.1.2. SIMULTANEOUS PERTURBATION STOCHASTIC APPROXIMATION (SPSA)

Simultaneous perturbation stochastic approximation (SPSA) method has been developed for difficult multivariate optimization problems. SPSA has recently attracted considerable attention in areas such as simulation-based optimization, signal image processing and statistical parameter estimation because of its power and relative ease of implementation [46].

The gradient approximation by using SPSA algorithm requires only two measurements or function evaluations independent from the dimension of the optimization problem where finite difference based gradient estimation algorithms require two times the number of optimization problem variables for each iteration of the algorithm. This is the most powerful side of the SPSA algorithm. Especially the problems that require large number of variables to be optimized, this feature of the algorithm provides a considerable decrease in the cost of optimization.

In simultaneous perturbation algorithm, all elements of $\hat{\theta}_k$ are randomly perturbed together ("simultaneously") to obtain two measurements $f(\hat{\theta}_k + c_k\Delta_k)$ and $f(\hat{\theta}_k - c_k\Delta_k)$. Each component of the gradient estimate $\hat{g}_k(\hat{\theta}_k)$ is formed from the difference of the two corresponding measurements, divided by an individual component in the perturbation vector [46]. For two-sided SP;

$$\hat{g}_k(\hat{\theta}_k) = \begin{bmatrix} \dfrac{f(\hat{\theta}_k + c_k\Delta_k) - f(\hat{\theta}_k - c_k\Delta_k)}{2c_k\Delta_{k1}} \\ \vdots \\ \dfrac{f(\hat{\theta}_k + c_k\Delta_k) - f(\hat{\theta}_k - c_k\Delta_k)}{2c_k\Delta_{kp}} \end{bmatrix} \tag{2.4}$$

$$= \frac{f(\hat{\theta}_k + c_k\Delta_k) - f(\hat{\theta}_k - c_k\Delta_k)}{2c_k} \left[\Delta_{k1}^{-1}, \Delta_{k2}^{-1}, \ldots, \Delta_{kp}^{-1}\right]^T$$

where $\Delta_k = \left[\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp}\right]^T$ is the distribution of the user-specified p-dimensional random perturbation vector (superscript "T" denotes vector transpose).

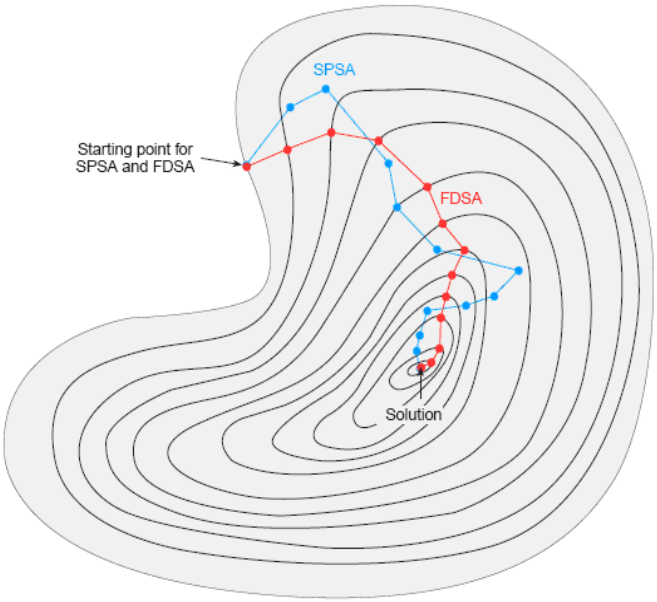

Figure 2.1: Relative search patterns for two dimensional problem for SPSA and FDSA algorithms [13]

The number of function evaluations needed at each iteration of FDSA increases with the dimension p while SPSA algorithm only needs two measurements independent of p since the numerator is the same in all components of p.

2.1.2.1. IMPLEMENTATION OF SPSA

How SPSA algorithm iteratively produces a sequence of estimates is shown in the following, step by step summary where the related material is directly adopted from [13].

Step 1: Initialization and coefficient selection. Set counter index $k=1$. Pick initial guess and non-negative coefficients a, c, A, α *and* γ in the SPSA gain sequences $a_k = a/(A+k)^{\alpha}$ and $c_k = c/k^{\gamma}$. The choice of the gain sequences (a_k and c_k) is critical to the performance of SPSA (as with all stochastic optimization algorithms and the choice of their respective algorithm coefficients). Spall [14] provides some guidance on picking these coefficients in a practically effective manner (In cases where the elements of θ have very different magnitudes, it may be desirable to use a matrix scaling of the gain a_k if prior information is available on the relative magnitudes. The next section discusses a second-order version of SPSA that automatically scales for different magnitudes).

Step 2: Generation of the simultaneous perturbation vector. Generate by Monte Carlo a p-dimensional random perturbation vector Δ_k , where each of the p components of Δ_k is independently generated from a zero mean probability distribution satisfying the preceding conditions. A simple (and theoretically valid) choice for each component of Δ_k is to use a Bernoulli ±1 distribution with probability of 1/2 for each ±1 outcome. Note that uniform and normal random variables are not allowed for the elements of Δ_k by the SPSA regularity conditions (since they have infinite inverse moments).

Step 3: Function evaluations. Obtain two measurements of the function $f(.)$ based on the simultaneous perturbation around the current $\hat{\theta}_k : y(\hat{\theta}_k + c_k\Delta_k)$ and $y(\hat{\theta}_k - c_k\Delta_k)$ with the c_k and Δ_k from Steps 1 and 2.

Step 4: Gradient approximation. Generate the simultaneous perturbation approximation to the unknown gradient $g(\hat{\theta}_k)$:

$$\hat{g}_k(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k\Delta_k) - y(\hat{\theta}_k + c_k\Delta_k)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix} \tag{2.5}$$

where Δ_{ki} is the ith component of the Δ_k vector (which may be ±1 random variables as discussed in Step 2); note that the common numerator in all p components of $\hat{g}_k\left(\hat{\theta}_k\right)$ reflects the simultaneous perturbation of all components in $\hat{\theta}_k$ in contrast to the component-by-component perturbations in the standard finite-difference approximation.

Step 5: Updating estimate. Use the standard SA form;

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k\hat{g}_k(\hat{\theta}_k) \tag{2.6}$$

to update $\hat{\theta}_k$ to a new value $\hat{\theta}_{k+1}$. Modifications to the basic updating step in Equation (2.6) are sometimes desirable to enhance convergence and to impose constraints. These modifications block or alter the update to the new value of θ if the "basic" value from Equation (2.6) appears undesirable.

Step 6: Iteration or termination. Return to Step 2 with k+1 replacing k. Terminate the algorithm if there is little change in several successive iterates or the maximum allowable number of iterations has been reached.

2.2. DIRECT SEARCH METHODS

Direct search method requires points $P \in R^n$ in design space which represent possible candidate solutions in the optimization problem as described by Hooke and Jeeves [10]. For any two points, P_1 and P_2 in the design space, if P_1 is a "better" candidate than P_2, then this situation is expressed as $P_1 \subset P_2$. There is an assumption that a single point P^*, the solution, with the property $P^* \subset P$ for all $P \neq P^*$. The pseudo-code given below which is adopted from [15] explains the direct search method.

Direct Search Method: Select a point a_0 arbitrarily as the first "base point."

$i \leftarrow 1$

repeat

Select a new point P_i

if $P_i \subset a_{i-1}$ then

$a_i = P_i$

else

$a_i = a_{i-1}$

end if

$i \leftarrow i+1$

until no beter points are found.

One of the important steps in the above algorithm is, selecting a new point. Considering the strategy for selecting a new point, direct search methods can be classified into different categories:

- Random Search (One at a time search)
- Pattern search methods (Hooke-Jeeves pattern search),
- Simplex based methods (Nelder-Mead simplex method)

2.2.1. NELDER-MEAD SIMPLEX METHOD

For optimization of a real valued function $f(x)$, $x \in R^n$, the Nelder-Mead simplex algorithm is an effective method. To define a complete Nelder-Mead method, four scalar parameters has to be defined; these parameters are coefficients of reflection (ρ) , expansion (X) , contraction (γ) , and shrinkage (σ) . The material of this part is based on the following reference, [16].

According to the original Nelder-Mead paper [9], parameters of the method should satisfy the conditions below.

$$\rho > 0\,, \quad X > 1\,, \quad X > \rho\,, \quad 0 < \sigma < 1 \tag{2.7}$$

Usually the values of the parameters in standard Nelder-Mead simplex algorithm are selected as:

$$\rho = 1\,, \quad X = 2\,, \quad \gamma = \frac{1}{2}\,, \quad \sigma = \frac{1}{2} \tag{2.8}$$

At the beginning of the $k\,th$ iteration of the algorithm, $k \geq 0$, an initial simplex Δ_k is given, along with its $n+1$ vertices and each of these vertices are defined in the design space R^n . Generally, it is assumed that $k\,th$ iteration begins ordering these vertices as $x_1^{(k)}, x_2^{(k)}, \ldots, x_{n+1}^{(k)}$, such that

$$f_1^{(k)} \leq f_2^{(k)} \leq \ldots \leq f_{n+1}^{(k)} \tag{2.9}$$

where $f_i^{(k)}$ denotes $f(x_i^{(k)})$.

In the $k\,th$ iteration, a new set of $n+1$ vertices are generated so the next iteration starts from a different initial simplex ($\Delta_{k+1} \neq \Delta_k$). Since the aim is to minimize f, we say that $x_1^{(k)}$ to $x_{n+1}^{(k)}$ can be ordered from the best point or vertex to the worst. Consequently, we refer to $f_{n+1}^{(k)}$ as the worst function value but $f_1^{(k)}$ is the best.

2.2.1.1. ONE ITERATION OF NELDER-MEAD ALGORITHM

1. Ordering; Algorithm needs an initial simplex which is the $n+1$ vertices satisfying the condition that $f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$, using the tie-breaking rules given below.

2. Reflection; Reflection point, x_r is computed first from

$$x_r = \overline{x} + \rho(\overline{x} - x_{n+1}) = (1+\rho)\overline{x} - \rho x_{n+1} \tag{2.10}$$

where $\overline{x}$ is the center location of the first n points (all vertices except x_{n+1}) and it is calculated by

$$\overline{x} = \sum_{i=1}^{n} x_i / n \tag{2.11}$$

Evaluate $f_r = f(x_r)$. If $f_1 \leq f_r \leq f_n$, the reflected point, x_r , is accepted and iteration is terminated.

3. Expansion; If the value of the reflection point f_r is smaller than f_1 , the expansion point x_e , is calculated from

$$x_e = \overline{x} + X(x_r - \overline{x}) = \overline{x} + \rho X(\overline{x} - x_{n+1}) = (1+\rho X)\overline{x} - \rho X\, x_{n+1} \tag{2.12}$$

and is evaluated as $f_e = f(x_e)$. If the value of the expansion point f_e is smaller than f_r , x_e is accepted as an expansion point and the iteration is terminated; otherwise if $f_e > f_r$, x_r is accepted and the iteration is terminated.

4. Contraction; If the value of the reflection point is greater than f_n , a contraction between $\overline{x}$ and the better of x_{n+1} and the reflection point x_r is performed.

4.1. Outside contraction. If $f_n \leq f_r < f_{n+1}$ (i.e., x_r is strictly better than x_{n+1}), the outside contraction is applied; x_c is calculated as

$$x_c = \bar{x} + \gamma(x_r - \bar{x}) = \bar{x} + \gamma\rho(\bar{x} - x_{n+1}) = (1 + \rho\gamma)\bar{x} - \rho\gamma x_{n+1} \qquad (2.13)$$

and *f* is evaluated at x_c as $f_c = f(x_c)$. If the value of the contraction point is smaller than f_r , x_c is accepted as a contraction point and the iteration is terminated; otherwise, go to step 5.

4.2. Inside contraction. If the value of the reflection point is greater than f_{n+1} , an inside contraction is performed; x_{cc} is calculated as

$$x_{cc} = \bar{x} + \gamma(\bar{x} - x_{n+1}) = (1 - \gamma)\bar{x} + \gamma x_{n+1} \qquad (2.14)$$

and *f* is evaluated at x_{cc} as $f_{cc} = f(x_{cc})$. If $f_{cc} < f_{n+1}$, x_{cc} is accepted and the iteration is terminated; otherwise, go to step 5.

5. Perform shrink step. Evaluate f at all of the following n points;

$$v_i = x_1 + \sigma(x_i - x_1) \text{ , } i = 2 \text{ , ... , } n+1 \qquad (2.15)$$

At the next iteration the (unordered) vertices of the simplex consist of x_1 , v_2 , ... , v_{n+1} .

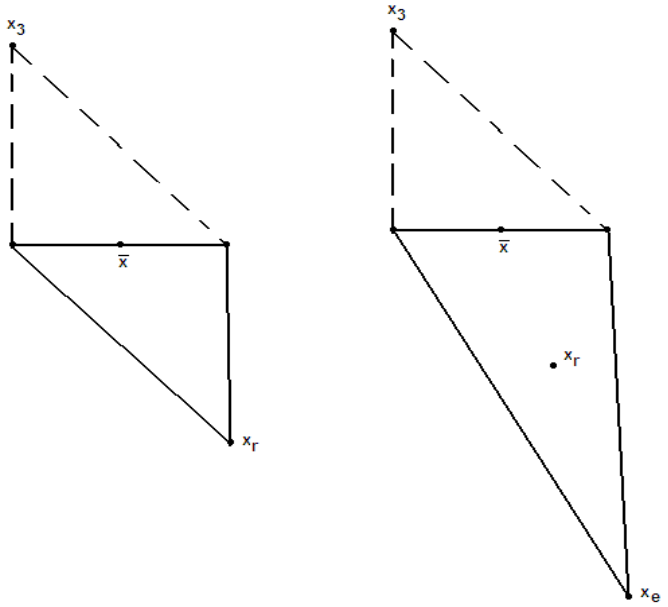

Figure 2.2: After a reflection and expansion step Nelder-Mead Simplex is shown. The dashed lines refer to original simplex [16]

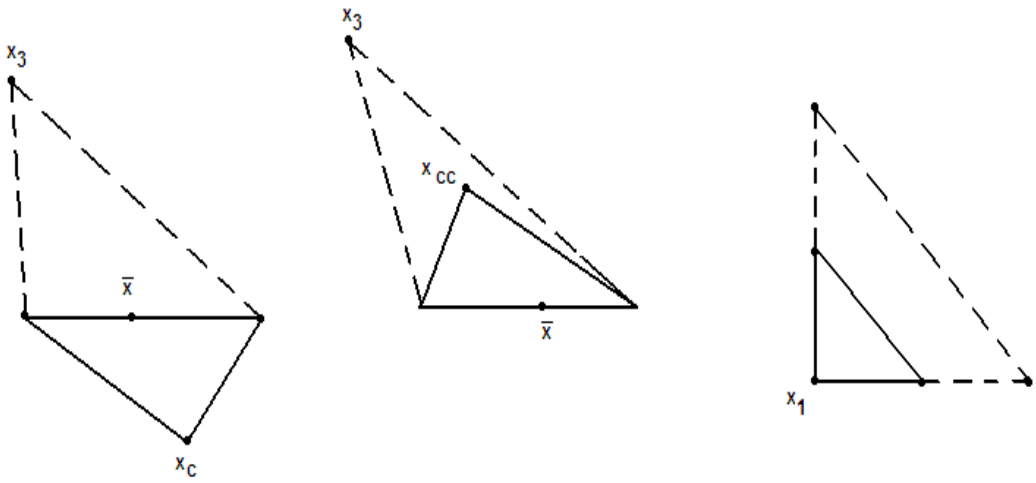

Figure 2.3: After an outside, inside contraction and shrink a Nelder-Mead simplex is shown. The dashed lines refer to original simplex [16]

For two dimensional space (simplex is triangle) the effects of reflection, expansion, contraction, and shrinkage operations are shown in Figure 2.2 and Figure 2.3, respectively.

It is seen that, in all of the operations except the shrink step, the new vertex always lies on the (extended) line joining $\bar{x}$ and x_{n+1}. Furthermore, it is visually evident that the simplex shape changes noticeably during an expansion or contraction operations with the standard coefficients $f(x_1^{(k+1)}) \leq f(x_2^{(k+1)}) \leq, \ldots, \leq f(x_{n+1}^{(k+1)})$.

Nonshrink ordering rule; The worst vertex, $x_{n+1}^{(k)}$, is discarded when a nonshrink step occurs. During the iteration k, the accepted point, denoted by $v^{(k)}$, becomes a new vertex and takes position j + 1 in the vertices of the simplex at next iteration Δ_{k+1} , where

$$j = \max_{0 \leq l \leq n} \{ l \mid f(v^{(k)}) < f(x_{l+1}^{(k)}) \} \tag{2.16}$$

with all other vertices retain their relative ordering from iteration k.

Shrink ordering rule;

When a shrink step occurs, the only vertex carried over from the simplex at iteration k, Δ_k , to simplex at the next iteration, Δ_{k+1}, is $x_1^{(k)}$. For the case, $x_1^{(k)}$ and one or more of the new points are tied as the best point, only one tie-breaking rule is specified, if

$$\min\{ f(v_2^{(k)}), \ldots, f(v_{n+1}^{(k)}) \} = f(x_1^{(k)}) \tag{2.17}$$

then $x_1^{(k+1)} = x_1^{(k)}$.

Define the change index k^* of iteration k as the smallest index of a vertex that differs between iterations k and $k+1$:

$$k^* = \min\{i \mid x_i^{(k)} \neq x_i^{(k+1)}\} \tag{2.18}$$

When Nelder-Mead algorithm terminates at step 2, $1 < k^* \leq n$; when termination is at step 3, $k^* = 1$; with termination at step 4, $1 < k^* \leq n+1$; and with termination at step 5, $k^* = 1$ or 2.

2.2.2. ONE AT A TIME SEARCH

This search strategy is also known as the alternating variable method. It is a kind of random search method. This is the simplest method which consists of optimizing with respect to each variable [15].

As shown in Figure 2.4, for an example of two dimensional cases, variables are changed in turn if no further improvement on the function value is obtained. If an improvement obtained in one variable, algorithm goes on with that variable [15]. This sequence is repeated with ever-decreasing steps.

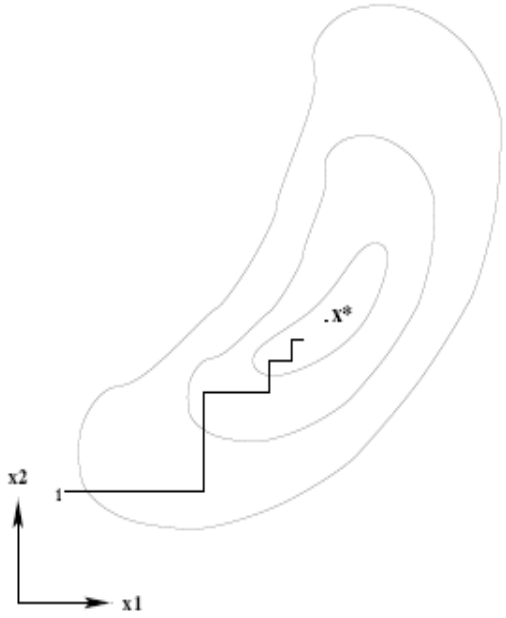

Figure 2.4: One at a time search method for function of two variables [15]

One of the drawbacks of this algorithm is that in most practical cases the direction of optimum is not along any of the coordinate axes. So this algorithm can be employed only for a limited number of cases. The progress is slow and it becomes very inefficient as the number of variables increase.

2.2.3. HOOKE AND JEEVES PATTERN SEARCH

Pattern search methods are better than random search methods from the aspect of determination of the search directions. They try to find better search directions. These directions are found by using exploration in the design search space. The name of going from one location to the new location in design space is called a move. A move results as a success if the value of the function $f(.)$ at the new point is higher then the value of $f(.)$ at the previous point; otherwise, it is a failure. The sequence of exploratory moves and pattern moves are the base of the pattern search methods.

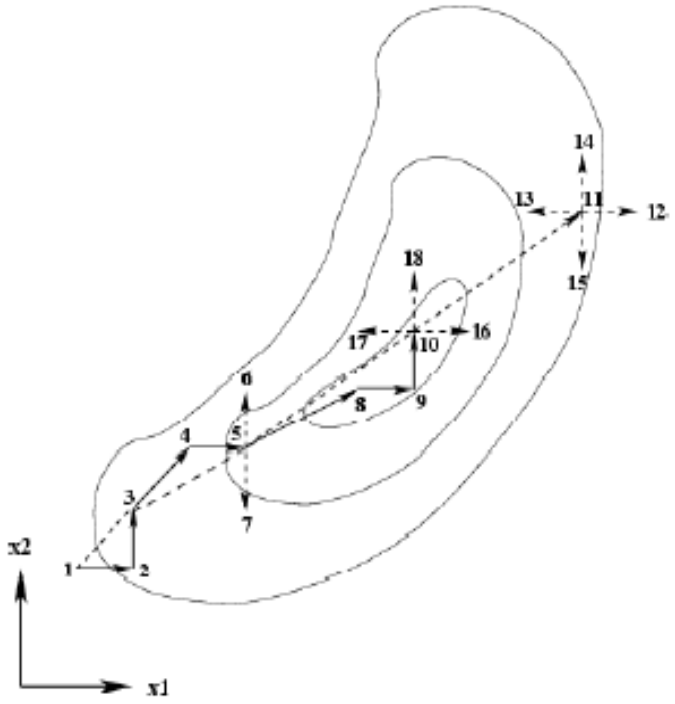

Figure 2.5: Hooke and Jeeves Pattern Search for two dimensions [15]

a. Exploratory move: In an exploratory move, a step is taken along the direction i. If this move results with a success, then new point obtained is retained. If the result is failure, then a step is taken in the opposite direction. Steps are taken by adding an increment to the variable or subtracting a decrement from the variable.

When all the n directions originated from coordinate axes are investigated, the exploratory move is complete. The point arrived at as a result of this exploratory move step, may or may not be distinct from the original point, which is called as the base point.

b. Pattern move: Initial starting base point and the obtained intermediate base point using the exploratory move define the search direction or, in other words "pattern". A pattern move takes one step from present base point in the direction specified by the pattern. This point is the new starting point of the next exploratory move step.

If, both a pattern move and the following exploratory move result with a failure, the algorithm returns to the previously obtained base point. Again, if the exploratory move around this base point also has failure the pattern is deleted and increment is reduced. The algorithm is repeated starting from this point. The search is terminated when the determined increments fail below a prescribed limit.

In Figure 2.5, point P_1 (marked 1) is the initial base point B_0. First exploratory move from B_0 begins by incrementing the direction on x_1 and the resulting point is P_2. At this point $f(P_2) < f(P_1)$, so the point P_2 is retained and exploration is continued by incrementing on the direction x_2. Since $f(P_3) < f(P_2)$, point P_3 is retained in place of P_2. The exploratory move is ended and P_3 becomes the second base point B_1. The direction of pattern move is $B_1 - B_0$ from P_3 to P_4 ($B_1 - B_0 = P_4 - P_3$). Now, $f(P_4)$ is not calculated, but an exploratory move is performed. The best point found along the direction of x_1 coordinate is P_5. Move along x_2 fails, as the points obtained P_6 and P_7 are not better than P_5; so exploratory move stage is complete and P_5 is retained.

Since $f(P_5) < f(B_1) = f(P_3)$, it becomes the new base point B_2. The material of this part is based on the reference [15].

2.3. DESCRIPTION OF THE ALGORITHM

The details of the proposed method for solving black box optimization problems with expensive-to-evaluate cost functions and/or constraints are presented in this section. It utilizes some direct search methods and stochastic approximation algorithms from the literature in a specific order.

2.3.1. OBTAINING A POINT SATISFYING CONSTRAINTS

The first step of the method consists of the use of finite difference stochastic approximation and simultaneous perturbation stochastic approximation algorithms in the context of finding a point satisfying constraints starting from a given initial parameter set.

Recall that, the problem can be stated as a constrained optimization problem. A general constrained optimization problem is generally written as a nonlinear optimization problem of the following form:

$$\begin{aligned} &\min f(\bar{\theta}) \ , \ \bar{\theta} = (\theta_1, \theta_2, ..., \theta_d)^t \in F \subset S \subset R^d \\ &\textit{subject to} \\ &g_i(\bar{\theta}) \le 0 \quad i = 1, 2, ..., q \end{aligned} \tag{2.19}$$

where $\bar{\theta} = (\theta_1, \theta_2, ..., \theta_n)^T$ is the vector of solutions, F denotes the feasible region and S is the search space. In that problem, there are q number of inequality constraints, and $f(\bar{\theta})$ is usually called the criterion function or the objective function. A feasible solution of the problem $\bar{\theta}^*$ satisfies all the constraints.

Generally a penalized model is constructed, where the violation of the constraints is added to the cost function as a penalty term. A large cost or "penalty" is added to the cost function for points that are outside of the original feasible region F. In this way,

an optimization problem with "costly" constraints can be transformed into an unconstrained optimization problem. A general penalized formulation of the problem is as follows;

$$f_p(\bar{\theta}) = f(\bar{\theta}) + \sum_{i=1}^{m} C_i d_i^k$$
$$\textit{where} \qquad (2.20)$$
$$d_i = \left\{ \alpha_i \, g_i(\bar{\theta}) \,, \; \textit{for } i = 1,...,q \right\}$$

$f_p(\bar{\theta})$ is the penalized cost function, $f(\bar{\theta})$ is the unpenalized cost function, and C_i is a constant imposed for the violation of constraint i. If constraint i is violated $\alpha_i = 1$, else $\alpha_i = 0$. d_i is the distance measure of constraint i at the feasible point $\bar{\theta}$, and k is a user defined exponent, usually taking the values 1 or 2.

Thus, rewriting the unconstrained form of the problem;

$$\max f_p(\bar{\theta}) = f(\bar{\theta}) + c.p(\bar{\theta}) \qquad (2.21)$$

where c is a sequence of constant penalty parameters and $p(\bar{\theta})$ is called the penalty function. Generally, the penalty parameter c has to be selected larger than a threshold value depending on both functions $f(\bar{\theta})$ and $p(\bar{\theta})$. But, since there is no complete information about the function $f(\bar{\theta})$, slowly increasing the parameter c during the optimization process, helps the convergence ($\lim_{n\to\infty} c_n = \infty$).

Now, the standard stochastic approximation technique can be applied with the simultaneous perturbation or finite difference gradient estimate to optimize the cost function $f_p(\bar{\theta})$. In other words the original problem can be solved with an algorithm of the following form:

$$\begin{aligned}\bar{\theta}_{k+1} &= \bar{\theta}_k - a_k \hat{\nabla} f_p(\bar{\theta}_k) \\ &= \bar{\theta}_k - a_k \hat{\nabla} f(\bar{\theta}_k) - a_k c_k \hat{\nabla} p(\bar{\theta}_k)\end{aligned} \tag{2.22}$$

The gradient estimates can be calculated by simultaneous perturbation or finite difference stochastic approximation algorithms. At the end of this process, a point $\bar{\theta}$ satisfying the constraints is obtained.

The penalized cost function and original cost function are identical for the case when all the constraints are satisfied ($f_p(\bar{\theta}) = f(\bar{\theta})$). The cost function value at this feasible point constitutes an artificial surface which can be called as the constrained surface $(f_p(\bar{\theta}) = const)$.

2.3.2. SAMPLING THE CONSTRAINED SURFACE

After finding a point which satisfies the constraints, the aim is to increase the number of sample points. In this part, the problem of choosing the best sampling instances from the parameter space is discussed.

It is obvious that, a sampling plan is needed in the design variable space that attempts to maximize information from an evaluation with as few evaluation points as possible, which can be important when the points are computationally expensive to evaluate.

In order to get samples on the constrained surface, a sampling strategy based on direct search algorithms is employed. Moreover, eigen-analysis of the covariance matrix of obtained samples (starting from a sample point, $\bar{\theta} = \{\theta_1, \theta_2, ..., \theta_n\}$ nearly on the artificial constrained surface) is utilized for determination of the sampling step size. High curvature parts of the surface have to be sampled densely.

Direct search algorithms which do not make gradient estimates and involve relatively few function evaluations, are the main tool for the proposed sampling method. Further, the locations where the direct search algorithms are planned to execute, is

determined by considering the curvature estimates of the surface. For example, at an arbitrary instant of the algorithm, suppose that, k samples are obtained ($\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_k$) on the constrained surface and now the concern is getting the next $(k+1)th$ sample. This sample is searched at a distance u (step size) from lastly obtained $k\,th$ sample, the distance u is determined according to the eigen-analysis of the covariance matrix of recently obtained specified amount of samples, starting from the last sample $\bar{\theta}_k$ ($\bar{\theta}_k, \bar{\theta}_{k-1}, \ldots$).

In the previous section, the penalized model $f_p(\bar{\theta})$ of the problem is constructed. For this model optimization process is employed with the help of stochastic approximation algorithms. Then the initial point, $\bar{\theta}^*$ which satisfies all the constraints, is obtained. At this point, penalized cost function and original cost function give identical results ($f_p(\bar{\theta}) = f(\bar{\theta}) = c$). Now the aim is increasing the number of sample points satisfying the condition $f_p(\bar{\theta}) = c_i$ which constitutes a surface (constrained surface). In chapter 3, representation procedure of a surface using sample points from this surface is presented.

The steps of the sampling procedure are given below;

Step1: In a specified neighborhood of initially obtained sample point $\bar{\theta}$ on the constrained surface ($f_p(\bar{\theta}) = c_i$), generate m number of new sample points. Giving arbitrary perturbations to each component of that sample point (design parameter), $\bar{\theta} = (\theta_1, \theta_2, \ldots \theta_n)$, a design parameter set $D = \{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_m\}$ is obtained. For example, $\bar{\theta}_1 = (\theta_1 + perturbation, \theta_2, \ldots \theta_n)$, $\bar{\theta}_2 = (\theta_1, \theta_2 + perturbation, \ldots \theta_n)$.

Step2: For each design point in the parameter set $D = \{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_m\}$, execute one of the direct search algorithms. This results in a new set of design points; this time, each of them is located on the constrained surface, $D^* = \{\bar{\theta}^*_{S1}, \bar{\theta}^*_{S2}, \ldots, \bar{\theta}^*_{Sm}\}$.

Step3: For the sample point set $D^* = \{\bar{\theta}^*_{S1}, \bar{\theta}^*_{S2}, ..., \bar{\theta}^*_{Sm}\}$, define a sampling direction d such that the next sample point is searched in that direction. Estimation of sampling direction is detailed in section 2.3.2.2 .

Step4: Eigen-analysis of the covariance matrix of design point set $D^* = \{\bar{\theta}^*_{S1}, \bar{\theta}^*_{S2}, ..., \bar{\theta}^*_{Sm}\}$ is used to estimate the curvature. This curvature value is used to determine the distance of the next sampling location. Sampling step size, u is defined by using this curvature.

Step5: Sampling step size, u and search direction, d are used for calculating the approximate location of the new sample ($\bar{\theta}_{Snew}$).

$$\bar{\theta}_{Snew} = \bar{\theta}_S + u * d \tag{2.23}$$

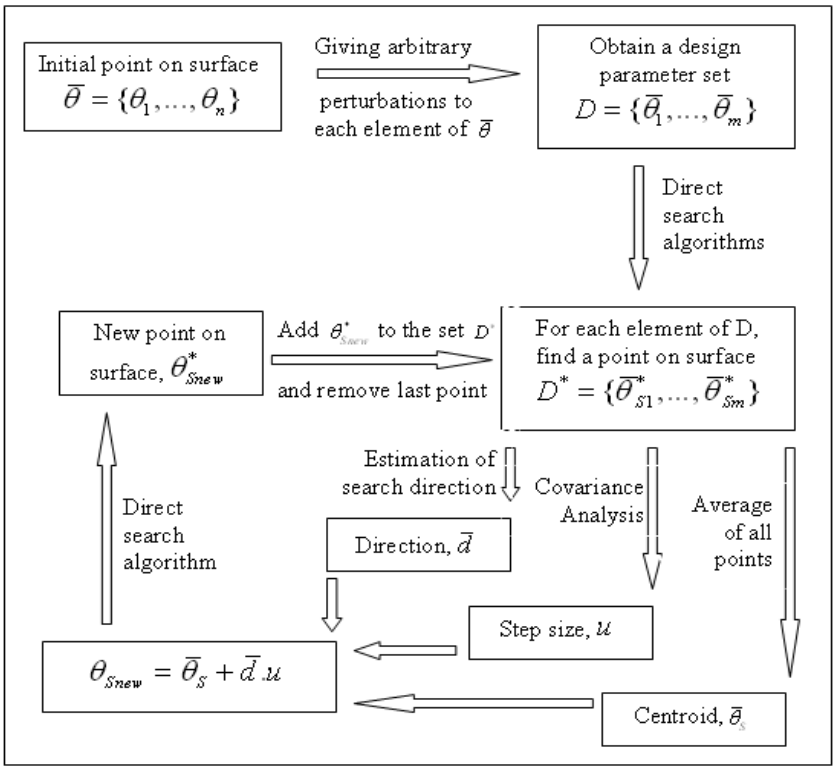

Figure 2.6: Sampling procedure

Executing one of the direct search algorithms at the point $\bar{\theta}_{Snew}$, a point on the constrained surface, $\bar{\theta}^{*}_{Snew}$ is obtained accurately, where $\bar{\theta}_{S}$ refers to the centroid of the point sample set D^{*}.

$$\bar{\theta}_{S} = 1/m \sum_{i=1}^{m} \bar{\theta}_{i} \tag{2.24}$$

After obtaining each new sample point, the last sample is removed according to the sampling direction d from the set $D^{*} = \{\bar{\theta}^{*}_{S1}, \bar{\theta}^{*}_{S2}, ..., \bar{\theta}^{*}_{Sm}\}$ and the new sampling point $\bar{\theta}^{*}_{Snew}$ is inserted into the set D^{*}, again considering the sampling direction.

As a result, high curvature parts of the constrained surface are sampled more densely compared to the low curvature parts.

2.3.2.1. ESTIMATING THE SURFACE VARIATION

A clever sampling procedure is required to obtain more information requiring smaller function evaluations. Regions with high variation need to be sampled more densely. In contrast, regions with low variation can be sampled sparsely to reduce the computational costs. For this reason, sample size has to be determined according to the variation of the region.

Corresponding to the principal components of the sample point set, the magnitude of each eigenvalue can be taken as a measure of the variation, along the direction of the corresponding eigenvector. The concept of interest here is generally named as the Principal Component Analysis (PCA) which is a searching technique for patterns in a high dimensional data. The eigenvalues in principal component analysis represent the total variance explained by each factor. First principal component accounts for the main variability in the data, and each of the following components accounts for the remaining variability.

In this part, the estimation of the curvature of the underlying point cloud is considered. Suppose that, we are given an nxm input matrix $D = (\bar{\theta}_{1}, \bar{\theta}_{2}, ..., \bar{\theta}_{m})^{T}$,

where each of the m input vectors of $\bar{\theta}_i,\ i=1,.....,m$, is defined on an n dimensional space and let $Y=(y_1,y_2,...,y_m)^T$ be the resultant vector whose components y_i 's are the output values, corresponding to the input vectors $\bar{\theta}_i,\ i=1,.....,m$.

In other words, we are given a simulation data set;

$$D=\{\,(\bar{\theta}_i,y_i):\bar{\theta}_i\in R^n,\ y_i\in R,\ i=1,....,m\,\} \tag{2.25}$$

in which, both the values of the inputs and the corresponding outputs are made available. The aim is to analyze the curvature of this set.

2.1.3.2.1.1 Covariance Analysis

Eigen analysis of the covariance matrix of a point sample set obtained from a certain part of the surface can be used to estimate the curvature of that part of the surface, as it has been demonstrated in studies referenced by [18], [19].

Let $\bar{\theta}$ be the centroid of the point sample set D, i.e.,

$$\bar{\theta}=\frac{1}{m}\sum_{i=1}^{m}\bar{\theta}_i \tag{2.26}$$

Then, the nxn covariance matrix, C for this set is given by

$$C=\begin{bmatrix}\bar{\theta}_1-\bar{\theta}\\ \vdots\\ \bar{\theta}_m-\bar{\theta}\end{bmatrix}^T\bullet\begin{bmatrix}\bar{\theta}_1-\bar{\theta}\\ \vdots\\ \bar{\theta}_m-\bar{\theta}\end{bmatrix} \tag{2.27}$$

The covariance matrix, C describes the statistical properties of the distribution of the sample point set by accumulating the squared distances of these points from the center location.

Let's consider the eigenvector problem;

$$C \cdot v_l = \lambda_l \cdot v_l, \quad l \in \{0, 1, 2, ...\} \tag{2.28}$$

Because covariance matrix, C is positive semi-definite and symmetric, all eigenvalues λ_l are real-valued and the eigenvectors, v_l corresponding to the principal components of the sample set D, form an orthogonal frame. The eigenvalues λ_l are a measure of the variation of D, along the direction of the corresponding eigenvectors. The total variation, as the sum of squared distances of $\bar{\theta}_i,\ i = 1, ..., m$ from the center point is given by

$$\sum_{i=1,...,m} \left|\bar{\theta}_i - \bar{\theta}\right|^2 = \lambda_0 + \lambda_1 + \lambda_2 + ... + \lambda_n \tag{2.29}$$

Assuming $\lambda_o \leq \lambda_1 \leq ... \leq \lambda_n$, the eigenvector v_0 corresponding to the minimum eigenvalue, gives the direction of minimum variation.

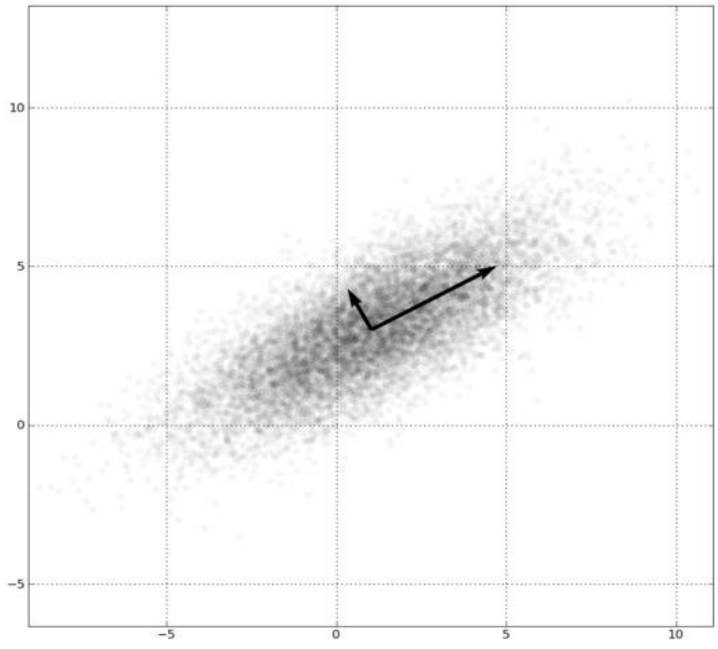

Figure 2.7: Principal components of a Gaussian distributed point set [41]

Now, a measure for the surface variation has to be defined. λ_0 quantitatively defines the variation along the surface normal, In other words, it estimates how much the points deviate from the space spanned by eigenvalues $\lambda_o, \lambda_1, ..., \lambda_{n-1}$. The value σ_n gives an insight about the curvature of the sample point set.

Then the surface variation can be defined as;

$$\sigma_n = \frac{\lambda_0}{\lambda_0 + \lambda_1 + ... + \lambda_n} \tag{2.30}$$

2.3.2.2. DIRECTION ESTIMATION

Search direction estimation to obtain a new sample point on constrained surface is critical for expensive-to-evaluate functions with no derivative information available. Starting from an estimated direction, convergence to the constrained surface needs too many function evaluations for badly estimated search directions which increases the computation cost. On the contrary, a thoroughly estimated direction prevents unnecessary evaluations.

2.1.3.2.2.1 Convex Hull Based Method

Two approaches are considered in this scope. First one utilizes a convex hull of previously sampled points to estimate the direction. This algorithm can be executed for multi dimensional problems requiring no additional preprocessing steps, in contrast to covariance analysis based method, detailed in following part.

The pseudo code and the flow diagram of the algorithm are given below;

- **Pset :** A newly generated point set on according to the estimated directions (most probably not on surface)
- **Used_P :** Points used on direction estimation
- **Points_On_Surf :** All points on constrained surface
- **Not_On_Hull :** Points not on convex hull
- **New_P_Surf :** Newly generated points on Surface
- **Cset :** Point set of convex hull

- stop = 0;
- While stop == 0
    - Converge each point on **Pset** to constrained surface and add them to **Points_On_Surf**
    - If number of points on **Points_On_Surf** < 3;
        - stop=1
    - Generate convex hull (**Cset**) of point set of **Points_On_Surf**
    - **Not_On_Hull =** Find points not in **Cset** but in **Points_On_Surf**
    - **Dir_C** = Erase points of **Used_P** from the set **Cset**
    - If number of points on **Dir_C** < 3;
        - stop=1
    - For each point in **Dir_C**
        - **Nearest** = Find nearest point in point set **Not_On_Hull**
        - **Vec** = Generate vector from **Nearest** to point in **Dir_C**
        - **New_P** = point in **Dir_C + Vec** x constant
        - **New_P_Surf** = Use one of the direct search algorithms to converge for **New_P**
        - **New_P_Surf =** Generate points between points **New_P_Surf** and add to **New_P_Surf** after converge
    - End For
    - **Pset = new_P**
- End While

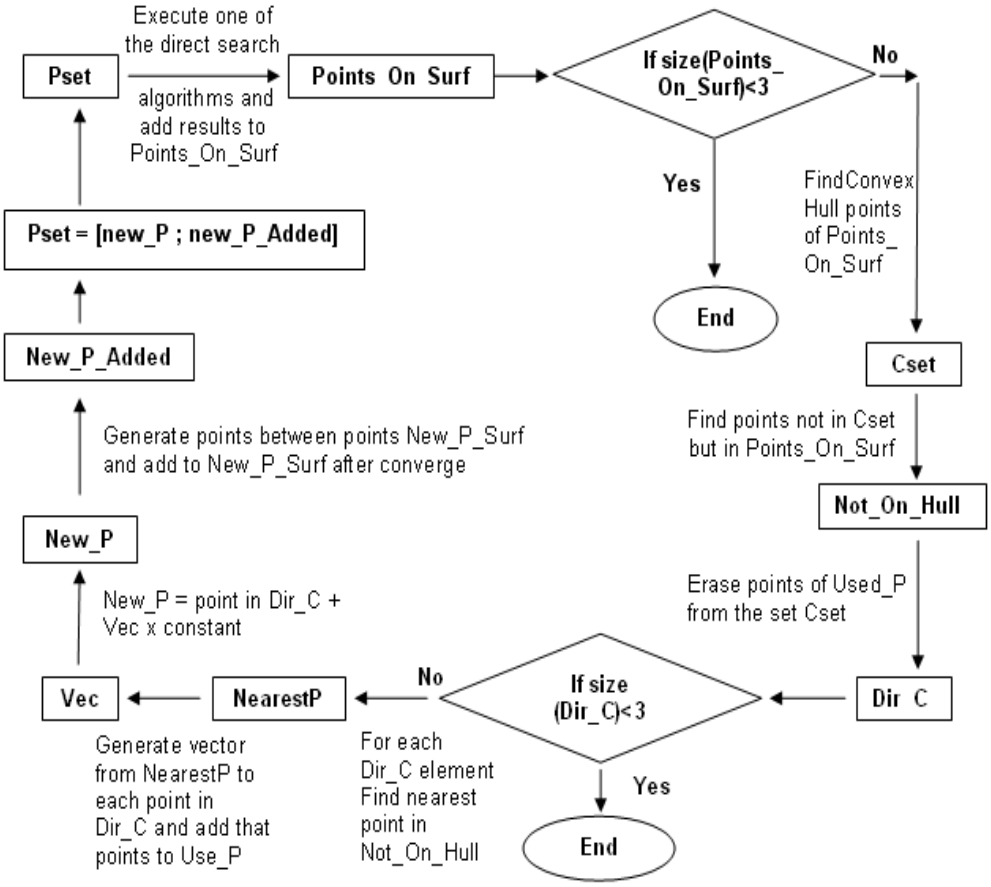

Figure 2.8: Flow diagram of convex hull based search direction estimation method

2.1.3.2.2.2 Covariance Analysis Based Method

The other direction estimation algorithm utilizes the covariance analysis of lastly obtained N number of points to estimate the direction. The direction of principal component of the point set is calculated first. Then, this direction is selected as a search direction considering the previously calculated direction. The angle between the lastly calculated two directions has to be smaller than 90 degrees. As it is seen, that direction estimation algorithm assumes that the surface is smooth. If the selected N value is high, sensitivity to sharp edges decreases. If selected N value is small, sensitivity to noisy observations increases and this may cause wrong convergence.

The algorithm works for three (two of them are design parameters) and smaller dimensional problems. If a higher dimensionsional problem is given, It has to be

converted to three dimensional problems. For example, a four dimensional problem can be solved for some specific values of a selected dimension. In this way, a four dimensional problem can be analyzed as a three dimensional problem for some values of the fourth dimension. That approach is applicable to all four or higher dimensional problems.

The pseudo code of the algorithm and flow diagram is given below;

- **Pset :** A specified number of lastly obtained points on constrained surface
- **Old_Dir :** Previously calculated direction (to control the consistency of the new direction)
- **New_P_Surf :** Lastly obtained point on surface (obtained one of the direct search algorithms)
- **New_P :** Lastly obtained point by using estimated direction(most probably not on the surface)
- **P_Dir :** Estimated Direction (calculated by covariance analysis of the point set)
- **Stop :** Variable used to control stopping condition

- Stop = 0;
- While stop == 0
  - **Old_Dir = P_Dir**
  - **P_Dir =** Find principal direction of the point set **Pset**
  - If Dot(**P_Dir, Old_Dir**) < 0
    - Reverse **P_Dir**
  - **New_P = P_Dir** x constant + Last point in **Pset**
  - **New_P_Surf =** Use one of the direct search algorithms to converge on surf from **New_P**
  - Add **New_P_Surf** to **Pset**
  - If **New_P_Surf** is close to starting point or out of bounds
    - **Stop = 1;**
  - Reorder points according to **P_Dir**
  - Erase last point in **P_set**
- End While

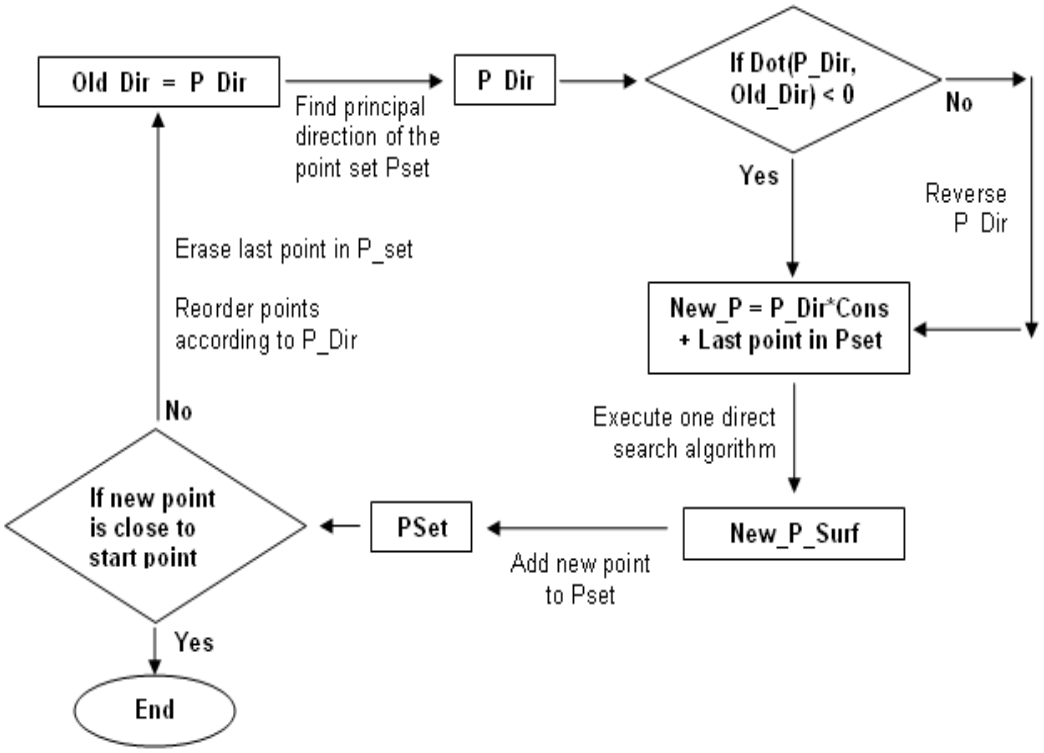

Figure 2.9: Flow diagram of covariance analysis based search direction estimation method

2.3.3. CONVERGE TO LOCAL OPTIMA

The sample point set ($D^* = \{\bar{\theta}^*_{S1}, \bar{\theta}^*_{S2}, ..., \bar{\theta}^*_{Sm}\}$) on the constrained surface is obtained by using the procedure detailed in previous section. This point set is used to define a barrier function. The aim in this section is to obtain convergence to local optima.

Firstly, a barrier function is constructed; the idea in a barrier method is to dissuade points $\bar{\theta}$ from ever approaching the boundary of constrained surface, defined by the sample point set.

A barrier function is any function $b(\bar{\theta}, D^*)$ that satisfies;

- $b(\bar{\theta}, D^*) \geq 0$ for all $\bar{\theta}$ that satisfies constraints, and
- $b(\bar{\theta}, D^*) \to \infty$ as distance between $\bar{\theta}$ and constrained surface converges zero.

For example, a barrier function $b(\bar{\theta}, D^*)$ can be defined as $b(\bar{\theta}, D^*) = 1/dist(\bar{\theta}, D^*)$. $dist(\bar{\theta}, D^*)$ is a function that gives the euclidean distance between parameter $\bar{\theta}$ and sample point set D^* .

The new form of the problem after adding a barrier function is

$$\max f_b(\bar{\theta}) = f(\bar{\theta}) + \frac{1}{c}.b(\bar{\theta}, D^*) \tag{2.31}$$

Barrier function forces the direction of the iterations to local optima. So the algorithm converges to local optima more robustly. It is obvious that, as the obtained points are getting away from the constrained surface, the effect of the barrier function decreases. Defining new constraints are necessary. For that reason, while execution of the algorithm is going on, constraints are updated and new sample points on this new constrained surfaces are obtained with the help of the procedure defined in previous section. Naturally, the barrier function uses, lastly obtained sample point set. The update condition for the constraints are predefined sufficient improvements on the cost function. The new value of the cost function defines a new constrained surface ($f(\bar{\theta}) = c_i$).

CHAPTER 3

SURFACE REPRESENTATION

Representation of an approximate local response surface model, which is both inexpensive and accurate, capturing the underlying relationship between input design parameter set and corresponding outputs, may be desirable, in some situations. Thus, it can be used to predict the output at some future observations or to estimate the derivatives at specific locations. For that reason, response surface development strategy is also considered in the scope of this study.

A rigorous way to develop an approximate surface model with predefined functions whose coefficients are to be determined by the measurements at some design points is defined. Such an approximate function is called a response surface model, (also known as metamodels, or surrogate models) which explores the relationships between several design variables and one or more response variables. The response surface model is an interpolation based on Radial Basis Functions (RBF). Radial basis functions are preferred as interpolants because they are computationally inexpensive and it is very easy to introduce locality. The Orthogonal Least Squares (OLS) algorithm is employed for the center selection procedure.

Development procedure of local response surface model with RBF is presented in sections 3.1. Section 3.2 is about selecting the most significant centers among the input sample set.

3.1. RBF BASED SURFACE GENERATION

Response surface methodology is a collection of statistical and mathematical techniques that can be used to approximate a system [20]. In this section, development procedure of a response surface model based on radial basis functions, matching the measurements obtained from the original model by experimentation or numerical simulation is considered. The developed models are cheaper to run; so they can be used in place of the exact model. In this way required computation time and/or cost can be reduced.

Since the original model may possess multiple local maxima or minima, employing linear or quadratic least squares model approaches would be more problematic for modeling. Radial basis functions are more suitable for these cases.

Because of its accurate and stable interpolation properties, radial basis functions are used in several fields of engineering. In this section the least squares radial basis function (LS RBF) is used to solve the problem of development of the local response surface model.

3.1.1. RESPONSE SURFACE METHODOLOGY

The field of response surface methodology consists of both of the following; experimental procedure for exploring the space of the design parameters and mathematical modeling to obtain a relationship between experiment results and design parameters. From mathematical aspect, a response surface model is an interpolation technique that fits complex multi dimensional function to its function values on specified function domain at some finite sample points [21].

The developed response surface model accuracy largely depends on the number of measurements or evaluations employed and their distribution on the design parameter space, as well as the approximation functions used for the development procedure of the response surface model. The general purpose of response surface methodology is usually one or both of the following: to gain some insight into the

operating principles of the related system and/or to seek some optimum settings for the variables involved.

It is assumed that we are concerned with the analysis of a complex system whose input-output relationship is given by

$$y = f(\bar{\theta}) \tag{3.1}$$

where $\bar{\theta} = (\theta_1, \theta_1, ..., \theta_n)^T$ is a design parameter vector. The function *f(.)* is assumed to be unknown or partially known and -as mentioned before- can only be evaluated experimentally or through a computationally intensive numerical simulation. The system is represented by scattered data $y_i = f(\bar{\theta}_i)$, $i = 1, ..., m$. To study such a system at a considerable cost, an approximation to the original function *f(.)* has to be developed.

The developed approximate function model constitutes a response surface model. Since, many interesting scientific and engineering systems are complex, the processing times are important. A model which can significantly reduce the processing time is absolutely desirable.

3.1.2. SURFACE RECONSTRUCTION WITH RBF

The problem of surface reconstruction from scattered point cloud has been studied extensively in the field of computer graphics for decades. Quickly and robustly reconstruction of a continuous surface from the unorganized points is desirable, for that reason, some useful methods are offered by researchers to overcome this important problem [18],[22],[23].

RBF attracts more attentions recently in multidimensional data interpolation [22], [24],[25],[26]. It is identified as one of the most accurate and stable methods to solve scattered data interpolation problems.

3.1.2.1. RADIAL BASIS FUNCTIONS

A radial basis function is a real-valued function whose value depends only on the distance from the origin ($\phi(\bar{\theta})=\phi(\|\bar{\theta}\|)$) or alternatively on the distance from a point c , called a center point, ($\phi(\bar{\theta},c)=\phi(\|\bar{\theta}-c\|)$).

Any function φ that satisfies the property $\phi(\bar{\theta})=\phi(\|\bar{\theta}\|)$ is a radial basis function. The norm is usually euclidean.

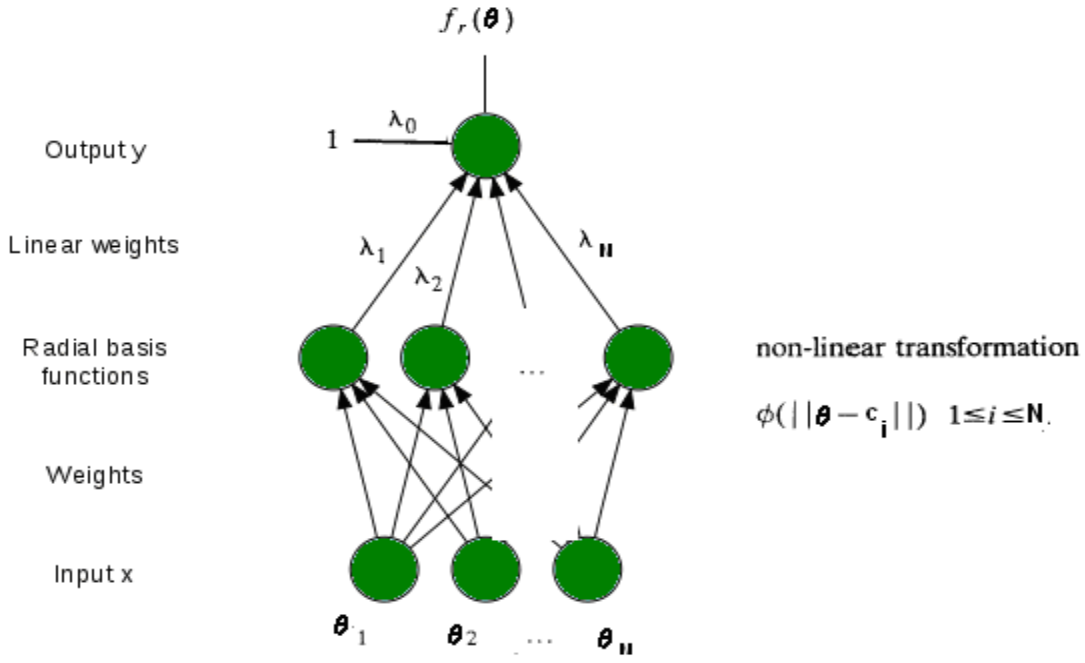

Figure 3.1: Radial Basis Function Network

A schematic of the RBF network with n number of inputs and corresponding scalar output is depicted in Figure 3.1. Such a network implements a mapping $f_r : R^n \to R$ according to;

$$f_r(\bar{\theta}) = \sum_{i=1}^{N} \lambda_i \cdot \varphi(\left\| \bar{\theta} - c_i \right\|) + \lambda_0 \qquad (3.2)$$

where $\bar{\theta} \in R^n$ is the input vector, $\varphi(.)$ is a given radial basis function defined from R^n into R, $\| \cdot \|$ denotes the euclidean norm, λ_i, $0 \le i \le N$ are the weights or parameters, $c_i \in R^n$, $1 \le i \le N$ are known as the RBF center points, and N is the number of center points.

Theoretical investigation and practical results shows that the choice of the nonlinearity of the function $\varphi(.)$ is not very important for the performance of the constructed RBF network [27]. For example, the thin-plate-spline function

$$\varphi(v) = v^2 \cdot \log(v) \qquad (3.3)$$

and the gaussian function

$$\varphi(v) = \exp(-v^2 / \beta^2) \qquad (3.4)$$

where β is a real constant, are two typical choices for radial basis functions. The function $\varphi(\upsilon)$ in Equation (3.3) goes to infinity as $\|\upsilon\|$ goes to infinity, whereas the function $\varphi(\upsilon)$ in Equation (3.4) goes to zero as $\|\upsilon\|$ goes to infinity. These two nonlinearities have quite different properties but RBF networks constructed by using any of them have good approximation capability [27].

3.1.2.2. LS RBF SURFACE RECONSTRUCTION

As mentioned before, in RBF methods, surface reconstruction problem is considered as a scattered data interpolation problem. This problem can be stated as, given a set of fixed N number of data points $\{\bar{\theta}_1, \bar{\theta}_2, ..., \bar{\theta}_N\}$ sampled from a surface S in R^{n+1} and a set of function values $\{f_i\}_{i=1}^{N} \in R$, find an interpolant $f_r : R^n \to R$ such that

$$f_r(\bar{\theta}_i) = f_i \quad i = 1,2,...,N \tag{3.5}$$

By providing a set of inputs and the corresponding outputs to the interpolating function given in Equation (3.2), a linear system whose least square (LS) solution gives the unknown weights λ_i, can be obtained.

Let us rewrite the RBF formulation in Equation (3.2) into

$$f_r(\bar{\theta}) = \sum_{i=1}^{M} \lambda_i \varphi(\left\|\bar{\theta} - c_i\right\|) \tag{3.6}$$

where, M is the number of points used in reconstruction $(M << N)$ and T is the transpose operator. Define α and g, such that $\alpha = [\lambda_1, \lambda_2, ..., \lambda_M]$, $g = [\phi_1, \phi_2, ...\phi_M]^T$. Let $G = [\phi_1^T, \phi_2^T, ..., \phi_N^T]_{NxM}$. Then we have

$$f_r(\bar{\theta}) = G \cdot \alpha \tag{3.7}$$

The RBF interpolant interpolates the sample points with function values $h = \{f_i\}_{i=1}^N$, so the interpolation equations can be derived from the following optimization problem

$$J = \frac{1}{2} \cdot \left[(G \cdot \alpha - h)^T \cdot (G \cdot \alpha - h) \right]^2 \rightarrow \min \tag{3.8}$$

To minimize the cost function in Equation (3.8), we proceed as

$$\frac{\partial J}{\partial \alpha} = G^T \cdot (G \cdot \alpha - h) \tag{3.9}$$

Let the right hand side of Equation (3.9) be equated to zero:

$$G^T \cdot G \cdot \alpha = G^T \cdot h \tag{3.10}$$

Then α can be obtained as

$$\alpha = (G^T \cdot G)^{-1} \cdot G^T \cdot h \tag{3.11}$$

After the coefficients α are determined, the surface can be reconstructed with much fewer centers $(M << N)$ the subset of centers, M can be efficiently selected from the original samples using the OLS method.

3.2. CENTER SELECTION

In this part, orthogonal least squares algorithm is utilized to select the most significant centers among the input sample set. Since the problem of developing a response surface model from selected suitable center points has the same characteristics with the problem of surface reconstruction from an unorganized point cloud in the field of computer graphics, we take advantage of the literature related with this field.

Researchers in this field offer useful strategies and methods to solve the problem of surface reconstruction from an unorganized point cloud. Hoppe's [18] signed distance function based method, Amenta's [28], [29], Voronoi and crust based method, moving least square (MLS) of Shen [22], etc.

In general, subset of suitable centers is selected arbitrarily or randomly from the whole point set. It is clear that this approach is unsatisfactory and may suffer from numerical ill-conditioning problems. To select suitable centers, orthogonal least square (OLS) method can be employed as a forward selection procedure. This is a regression procedure such that the selected set of centers is the regressors maximizing the desired output.

3.2.1. ORTHOGONAL LEAST SQUARES (OLS) ALGORITHM

It is a generally recognized fact that utilizing significantly fewer points, a surface defined by a large point set can be approximated with desired precision; so center

selection is an important process for developing surfaces with radial basis functions. It reduces time for processing without much sacrifice of accuracy. The OLS algorithm is a greedy algorithm which chooses the suitable centers for radial basis functions systematically considering the individual contribution to error reduction. Surface developed by using these selected suitable centers approximate the original surface well.

All the material in this part is referenced from Chen's OLS algorithm [27]. To understand the working principle of the algorithm, consider the RBF interpolation function given in Equation (3.8); ($f_r(x)=G\cdot\alpha$) where G_{NxM} is a regression matrix that a fixed center point c_i with a given nonlinearity $\varphi(.)$ corresponds to a regressor g_i , $i=1,...,M$.

$$G_{NxM}=[g_1,g_2,...,g_M] \tag{3.12}$$

Now, the problem of selecting a suitable set of RBF centers from the given data set can be considered as an example of selecting a subset of significant regressors from a given candidate set.

The least square solution obtained in Equation (3.11) satisfies the condition that $G\cdot\alpha$ be the projection of measurements vector (h) onto the space spanned by the regressor vectors (g_i) which form a set of basis vectors.

To calculate the individual contribution to the error reduction from each basis vector the OLS algorithm investigates the transformation of the set of g_i into a set of orthogonal basis vectors.

The regression matrix G can be decomposed into

$$G=WQ \tag{3.13}$$

where Q is a MxM upper triangular matrix with ones on the diagonal, and zeros below the diagonal, that is,

$$Q = \begin{bmatrix} 1 & q_{12} & q_{13} & \cdots & q_{1M} \\ 0 & 1 & q_{23} & \cdots & q_{2M} \\ 0 & 0 & \cdots & & \vdots \\ \vdots & & \ddots & 1 & q_{M-1M} \\ 0 & & \cdots & 0 & 1 \end{bmatrix} \tag{3.14}$$

and W is a NxM matrix with orthogonal columns w_i such that

$$W^T W = D \tag{3.15}$$

where D is diagonal with elements d_i :

$$d_i = w_i^T \cdot w_i = \sum_{t=1}^{N} w_i(t) \cdot w_i(t), \quad 1 \le i \le M \tag{3.16}$$

The space spanned by the set of orthogonal basis vectors is the same space spanned by the set of g_j.

RBF linear system can be rewritten as

$$h = W\eta \tag{3.17}$$

The orthogonal least square solution is given by

$$\eta = H^{-1}W^T h \text{ ,or } \eta_j = w_j^T h/(w_j^T w_j), \ 1 \le j \le M \tag{3.18}$$

The coefficients α and vector η satisfy the following triangular system;

$$Q\alpha = \eta \tag{3.19}$$

the classical Gram-Schmidt method computes one column of Q at a time and orthogonalizes G as follows: at the $k\,th$ step, the $k\,th$ column is made orthogonal to each of the $k-1$ previously orthogonalized columns and this operation is repeated for $k = 2,...,M$.

The computational procedure can be represented as

$$\begin{aligned} & w_1 = g_1 \\ & q_{ik} = w_i^T \cdot g_k / (w_i^T \cdot w_i), \qquad 1 \le i \le k, \quad k = 2,...,M \\ & w_k = g_k - \sum_{i=1}^{k-1} q_{ik} \cdot w_i \end{aligned} \tag{3.20}$$

An error ration due to w_j is defined as

$$[err]_j = \eta_j^2 w_j^T w_j / (h^T h) \tag{3.21}$$

This ratio offers a simple and effective means of significant center selection in a forward-regression manner. This procedure can be summarized as follows:

The first selected center point;

For $1 \le i \le N$

$$w_1^{(i)} = g_i$$

$$[err]_1^{(i)} = ((w_1^{(i)})^T h)^2 / (((w_1^j)^T w_1^{(i)})(h^T h))$$

end for

Find $[err]_1^{(i_1)} = \max[err]_1^{(i)}$, $1 \le i \le N$

At the kth step, where $k \ge 2,$

For $1 \le i \le N$, $(i \ne i_1,...,i \ne i_{k-1})$

For $1 \le j \le k$

$$q_{jk}^{(i)} = w_j^T g_i / (w_j^T w_j)$$

$$w_k^{(i)} = g_i - \sum_{j=1}^{k-1} q_{jk}^{(i)} w_j$$

end for

$$[err]_k^{(i)} = ((w_k^{(i)})^T h)^2 / (((w_k^j)^T w_k^{(i)})(h^T h))$$

Find $[err]_k^{(i_k)} = \max[err]_k^{(i)}$, $1 \le i \le N,$ $\quad i \ne i_1, ..., i \ne i_{k-1}$

Select $w_k = w_k^{i_k} = g_{ik} - \sum_{j=1}^{k-1} q_{jk}^{i_k} w_j$

The process is stopped at the Mth step when the error ration satisfies

$$1 - \sum_{j=1}^{M} [err]_j < \rho \qquad (3.22)$$

where $0 < \rho < 1$ is a user defined tolerance.

CHAPTER 4

EXPERIMENTS AND RESULTS

In this section we report the results of some computational tests of the algorithm. The aim is to investigate the performance of the described method on test functions when utilizing different direct search methods and stochastic approximation methods. The numerical results from performed tests are presented as number of function evaluations and number of obtained points. Three test problems studied have been described in next section.

For the experiments, MATLAB environment has been preferred and all mentioned algorithms have been implemented in MATLAB (Version 6.5, Release 13).

4.1. DESCRIPTION OF TEST PROBLEMS

Rosenbrock functions with three and four dimensional variants, previously prepared offline four dimensional data set of simulation results of a special military scenario, generated with a special super computer provided by TÜBİTAK-İLTAREN Research Group and a stock exchange strategy data recorded for specific time period are the three selected test problems.

Rosenbrock function is a non-convex function generally used in testing the developed optimization algorithms. Rosenbrock function is also known as

Rosenbrock's valley or Rosenbrock's banana function. Two variants of the Rosenbrock function for multi dimensional applications are given in Equations 4.1 and 4.2 respectively.

$$f(x_1, x_2, \ldots x_N) = \sum_{i=1}^{N/2} [100(x_{2i-1}^2 - x_{2i}) + (x_{2i-1} - 1)^2] \tag{4.1}$$

A more involved variant is;

$$f(x) = \sum_{i=1}^{N-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2] \quad , \quad \forall x \in R^N \tag{4.2}$$

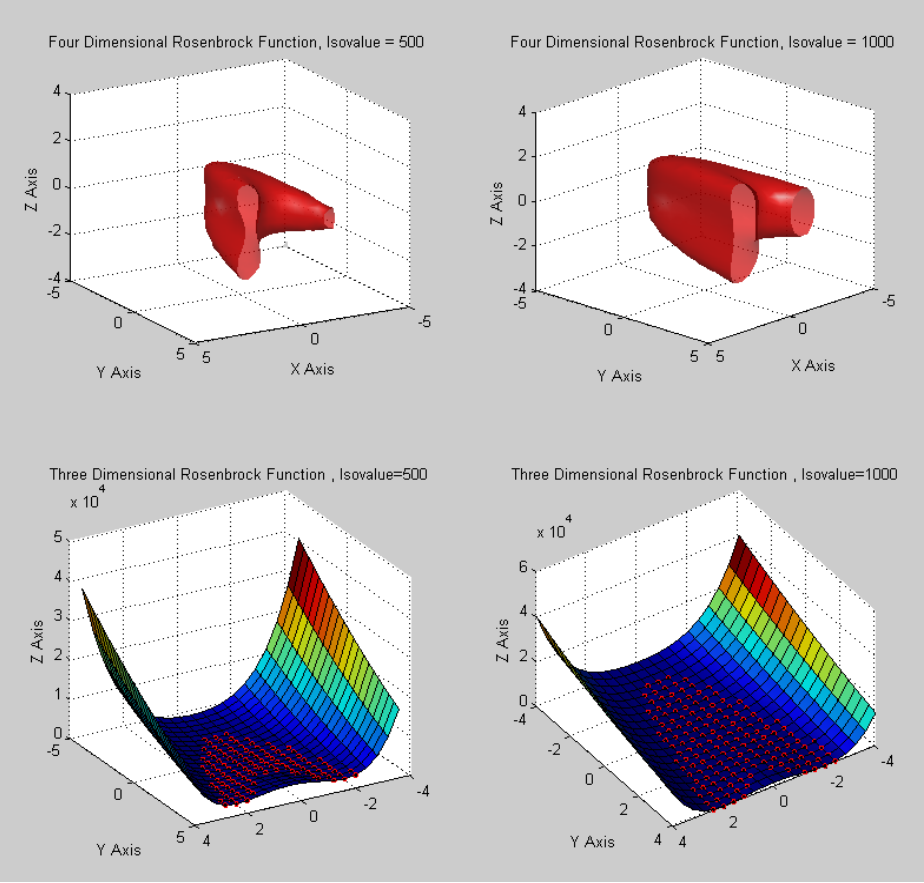

Figure 4.1: Three and four dimensional plots of the Rosenbrock function

In Figure 4.1, three and four dimensional plots of the Rosenbrock functions are shown. The design parameters are regularly sampled in the interval [-4, 4]. Four dimensional Rosenbrock function is plotted for the isovalues between 500 and 1000.

Another test environment is the offline generated four dimensional data set, generated by simulation trials of a special military scenario including a guided missile, sea platform and flare.

In those scenarios, sea platform tries to guarantee an acceptable miss distance value against a guided missile coming towards itself. For that reason, maneuvering with different heading rates and using a flare with different release and ignition times are the applicable ways for increasing the surviving chance of the sea platform. Simulations of that scenario for different values of design parameters and resulting miss distance values are recorded to generate an offline data set.

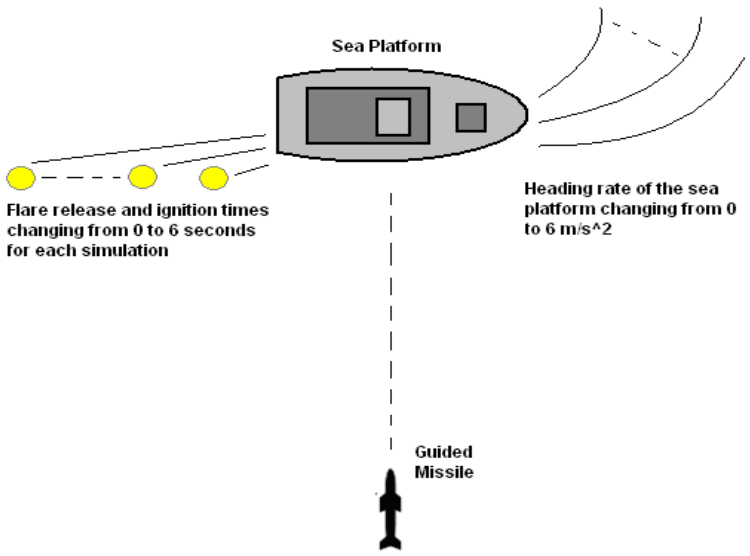

Figure 4.2: A military scenario example

Heading rate of the sea platform, flare release time and flare ignition times are selected as design parameters for miss distance value. All design parameter values

are changed in the interval [0, 6] with the step size of 0.2. So the offline data set includes 29791 simulation results.

Simulation trials are executed on a special super-computer with 8 Dual Core AMD Opteron Processor 880 at 2.40 GHz and 15.5 GB of RAM, which is provided by TÜBİTAK-İLTAREN Research Group and each simulation trial in that super-computer takes 0.5 sec. on the average. Four dimensional plot of that data set is shown in Figure 4.3 for isovalues (miss distances) 200, 220, 240, 260, 280 and 300.

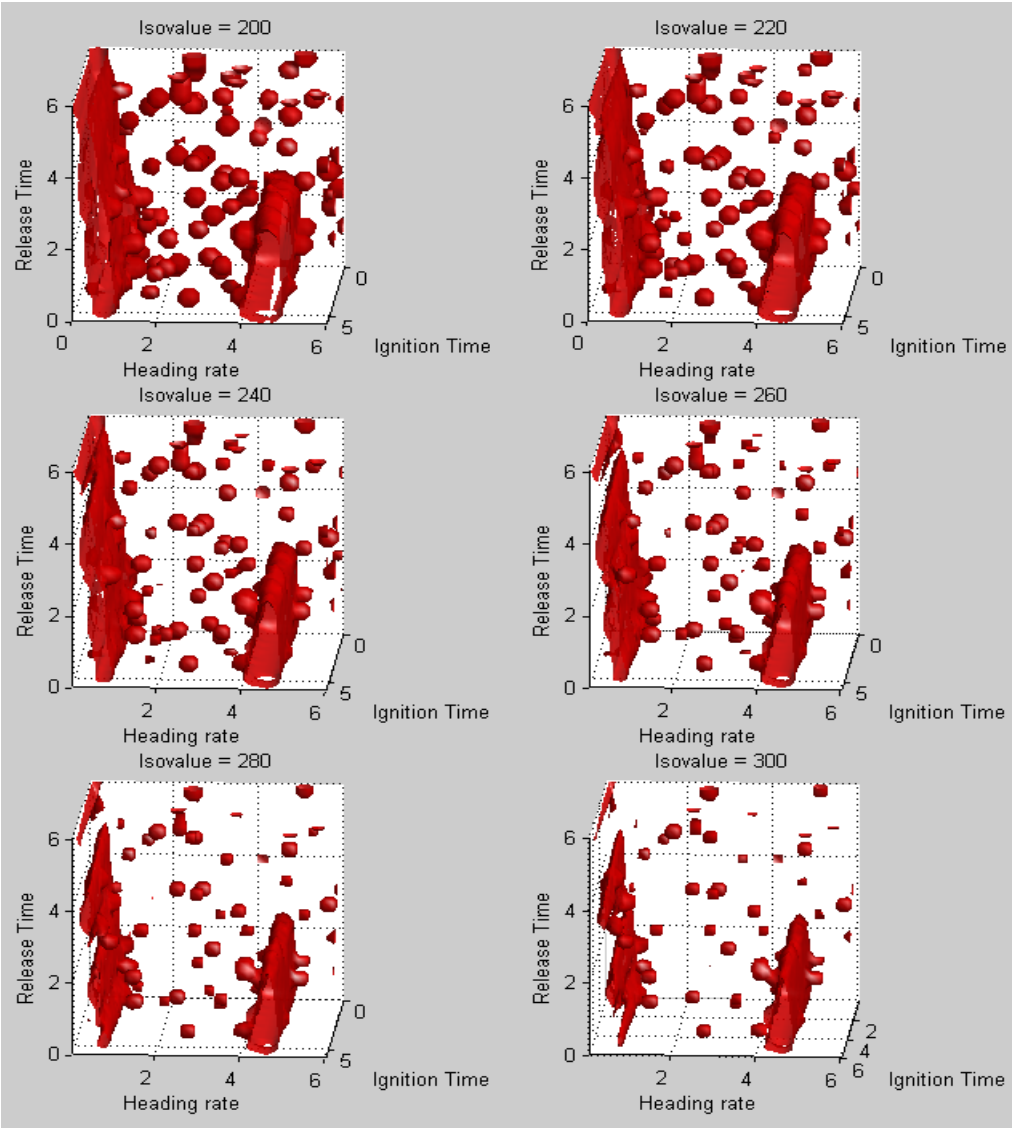

Figure 4.3 Simulation trials plot of offline data set for different isovalues.

The last test problem is the four dimensional data of trading strategy originated from EURUSD parity which is recorded for a specific time period. Three design parameters are the band period, deviation and take profit. The response parameter is the total profit. For a determined band period and deviation from the parity, trading strategy proposes a sell or buy decision considering the third parameter take profit.

Figure 4.4 shows response parameter, total profit for values of 1000, 750, and 500 respectively symbolized with colors red, green and yellow.

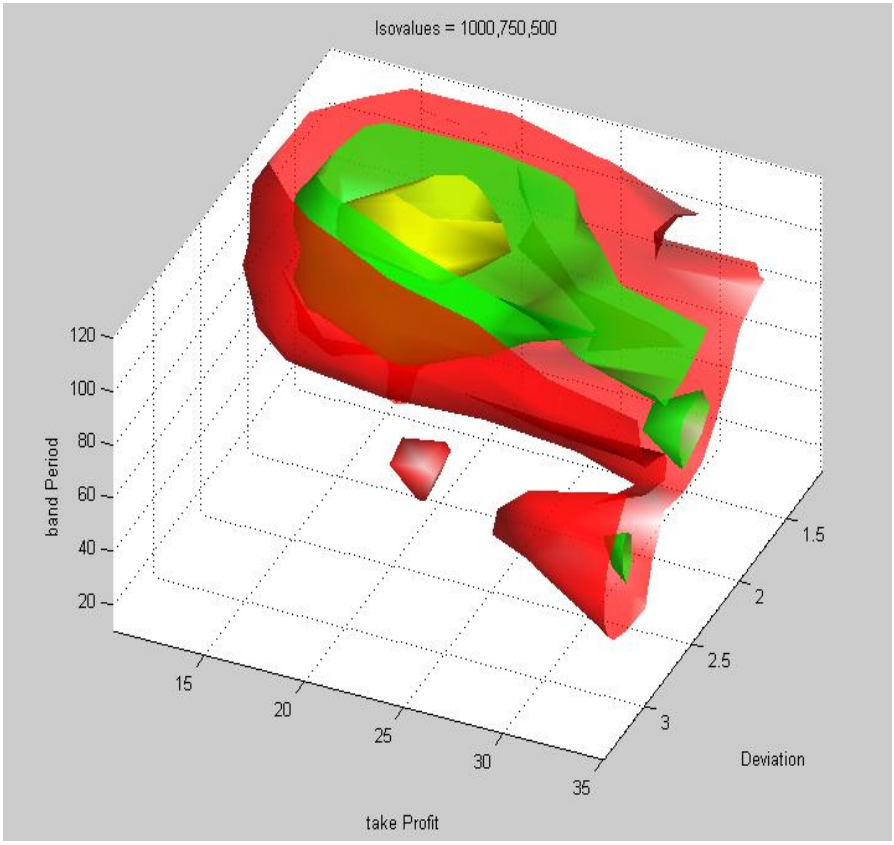

Figure 4.4: Plot of three values, 1000, 750 and 500 for total profit

Isosurfaces for different constraint values are used for the representation of the results of these four dimensional problems. MATLAB is selected as the main tool for the development process of the algorithms.

4.2. NUMERICAL RESULTS

Applications of the algorithms to test problems defined in the previous section is realized. Obtained results are presented separately for each test problem.

4.2.1. OFFLINE DATA SET OF İLTAREN

The application results of convex hull based and covariance analysis based search direction estimation algorithms to the four dimensional offline data set provided by TÜBİTAK-İLTAREN with two different direct search algorithms, Nelder-Mead simplex method and Hooke-Jeeves pattern search method are reported in this section. Required number of function evaluations and number of sample points obtained during the execution of the algorithms are given. Selected stopping criterion of the algorithm is obtaining a considerable improvement for the response parameter.

For the application of covariance analysis based direction estimation algorithm, four dimensional original data set of İLTAREN first passes from some preprocessing steps to obtain three dimensional cross sections for some specific values of the design parameter, heading rate. This preprocessing step is required because of the restriction that the direction estimation algorithm works only for three or smaller dimensional problems.

Results for the application of simultaneous perturbation stochastic approximation and finite difference stochastic approximation algorithms based gradient estimates to obtain a point satisfying constraints are also reported as required number of function evaluations and number of iterations needed for three and four dimensional problems of İLTAREN data set separately.

Application of the algorithm with Nelder-Mead and Hooke-Jeeves direct search algorithms are shown in Figure 4.5 and Figure 4.6, respectively. Search direction estimation algorithm is convex hull based.

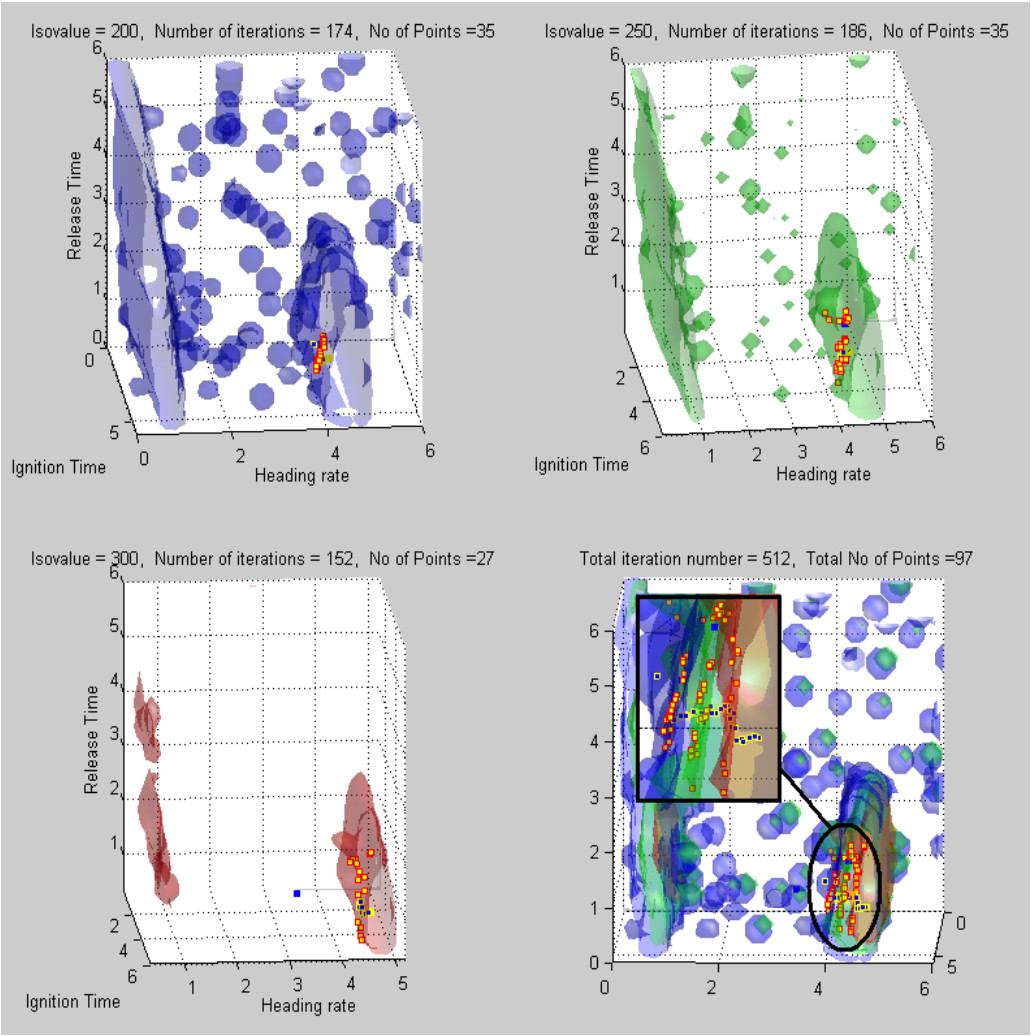

Figure 4.5: Application of the algorithm to four dimensional İLTAREN data with Nelder-Mead simplex algorithm and convex hull based search direction estimation.

In Figure 4.5 and Figure 4.6, first three plots show the isovalues (miss distance) 200, 250 and 300. These three plots are shown together in the last. Samples from the three constrained surfaces are yellow points with red edges. Blue points with yellow edges are the evaluation points required for searching local optima. Starting location for two of the applications is [3.8, 4, 1]. (Heading rate, ignition time and release time) Algorithm stops if the response parameter (miss distance) takes the value 350.

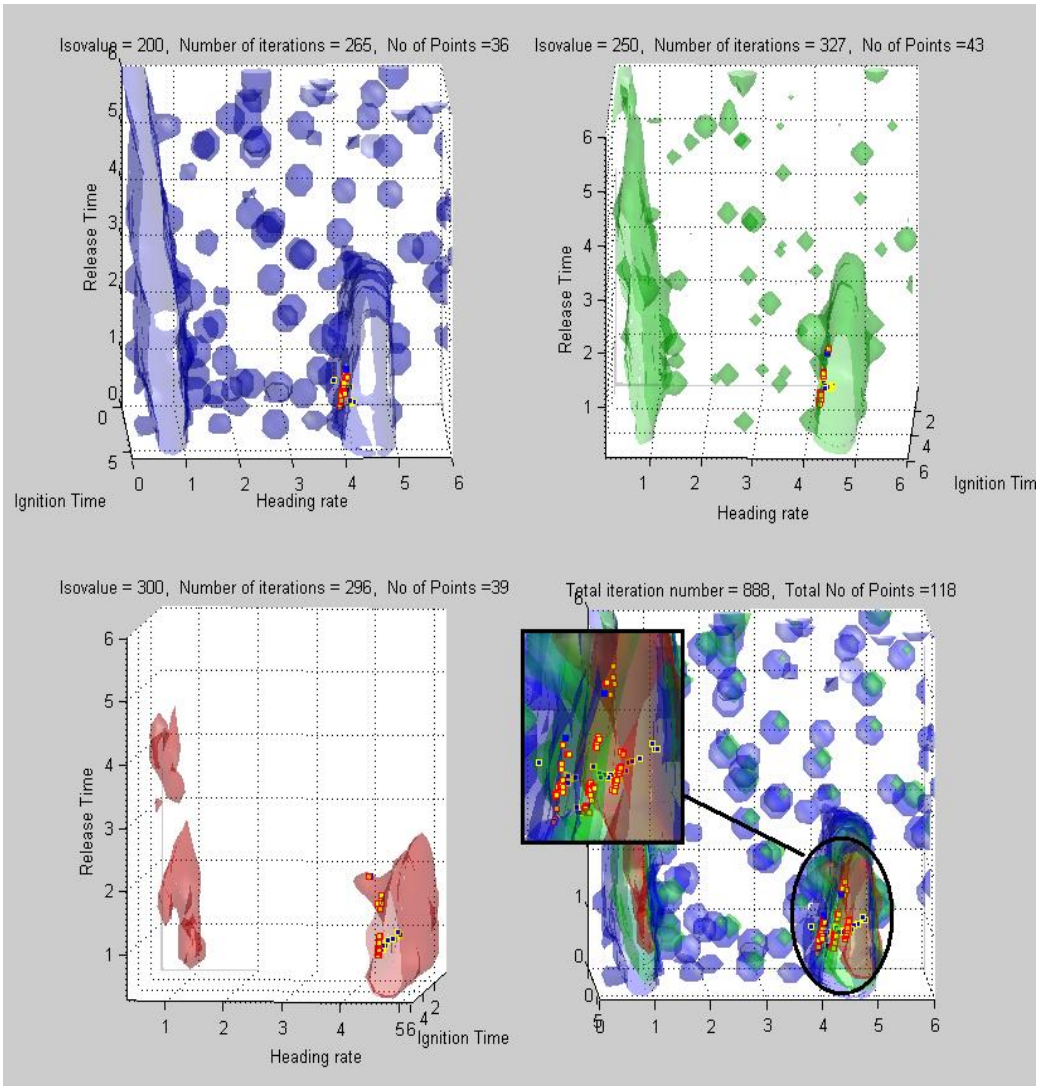

Figure 4.6: Application of the algorithm to four dimensional İLTAREN data with Hooke-Jeeves pattern search algorithm and convex hull based search direction estimation.

Table 4-1 : Results of four dimensional İLTAREN data for two direct search methods

Direct Search Method	Number of Iterations Required	Number of Points Obtained
Nelder-Mead	512	97
Hooke-Jeeves	888	118

Application of the algorithm with Nelder-Mead and Hooke-Jeeves direct search algorithms are shown in Figure 4.7 and Figure 4.8, respectively. But this time search direction estimation is covariance analysis based. So four dimensional original data set is considered for three dimensional cross section surfaces for some specific values of heading rate design parameter.

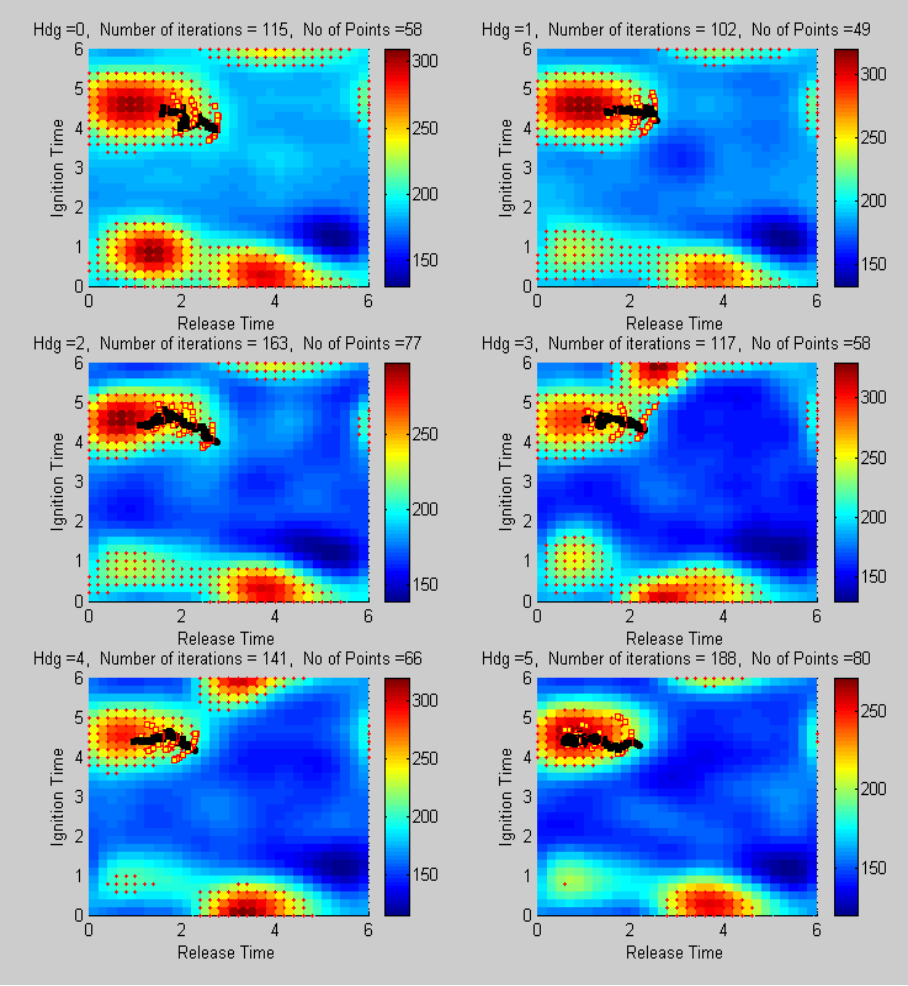

Figure 4.7: Application of the algorithm to three dimensional İLTAREN data with Nelder-Mead simplex algorithm and covariance analysis based search direction estimation

In Figure 4.7 and Figure 4.8, yellow points with red edges show the samples from constrained surfaces. Black points are evaluations for seeking local optima. Algorithm stops if response parameter miss distance gets the value of 280 or higher.

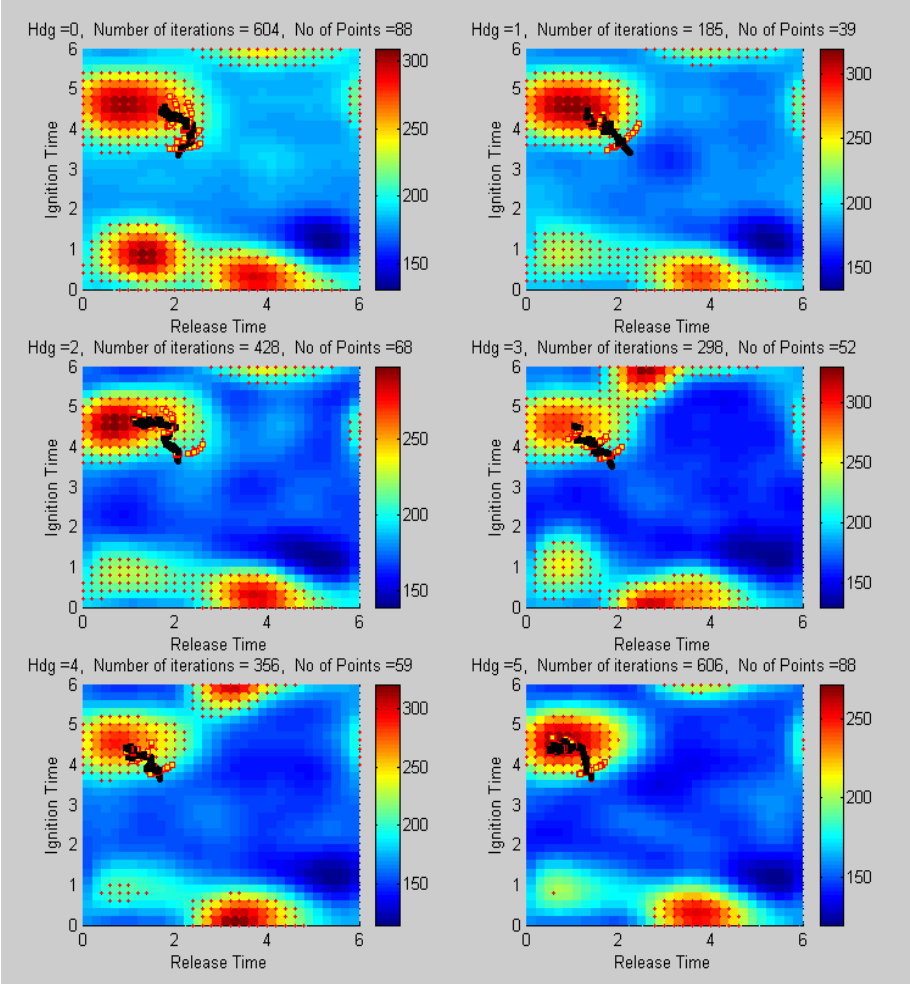

Figure 4.8: Application of the algorithm to three dimensional İLTAREN data with Hooke-Jeeves pattern search algorithm and covariance analysis based search direction estimation

Table 4-2 : Results of three dimensional İLTAREN data for two direct search methods

Direct Search Method	Number of Iterations Required	Number of Points Obtained
Nelder-Mead	826	388
Hooke-Jeeves	2477	394

Nelder-Mead simplex method requires less number of function evaluations when compared with Hooke-Jeeves pattern search method for three and four dimensional

İltaren data application. Furthermore, obtained number of points is nearly the same. It is obvious that, Nelder-Mead simplex method as a direct search algorithm is more favourable for that application.

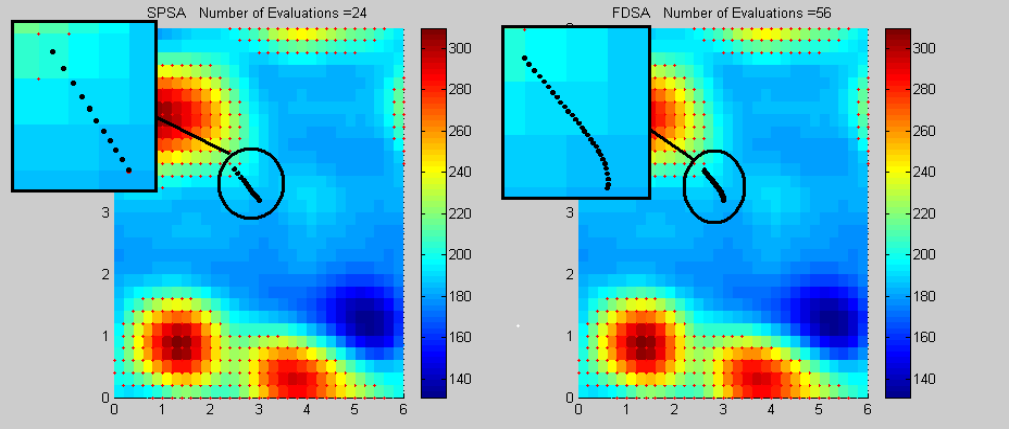

Figure 4.9: FDSA and SPSA algorithms for three dimensional İLTAREN data.

Simultaneous perturbation and finite difference based gradient estimates are utilized to obtain a point satisfying the constraints, starting from an initial arbitrary location. The application results of these algorithms to three and four dimensional data of İLTAREN are also presented in this section.

Table 4-3 : Results of FDSA and SPSA methods for three dimensional İLTAREN data

Gradient Estimate Method	Average No of Evaluations	Average No of Points	Number of Executions
FDSA	224	56	20
SPSA	72	38	20

In Figure 4.9, an example of execution of the SPSA and FDSA algorithms is shown for three dimensional İLTAREN data. Two of the algorithms start from the same initial point and executed 20 times. Average number of points required for

convergence and number of function evaluations needed are given in Table 4-3. FDSA algorithm requires four evaluations for gradient estimate, however SPSA requires only two.

As it is seen from the application results given in Table 4-3 and Table 4-4, using SPSA algorithm for three and four dimensional İltaren data applications is more efficient with respect to the FDSA algorithm. The search pattern for FDSA looks more consistent with respect to SPSA, however both of the algorithms converge nearly to the same point.

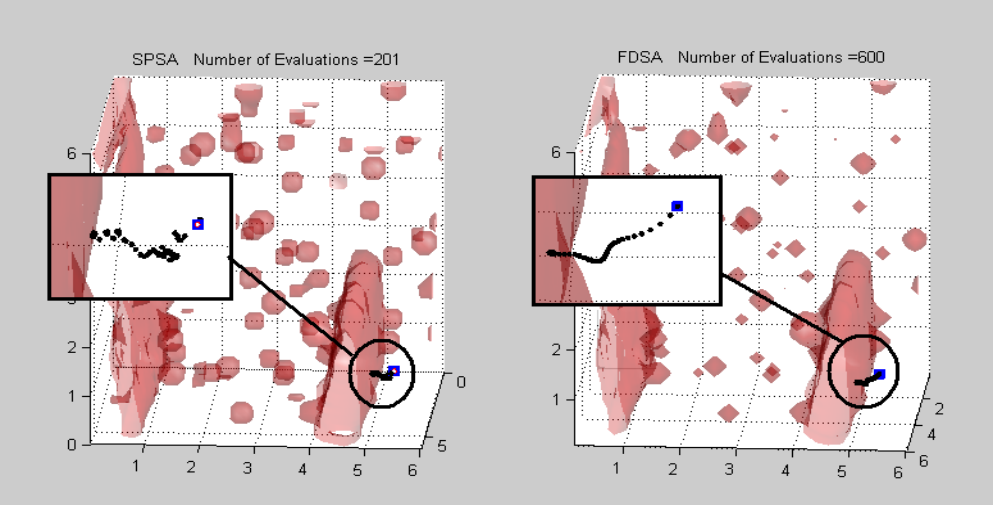

Figure 4.10: FDSA and SPSA algorithms for four dimensional İLTAREN data

Table 4-4 : Results of FDSA and SPSA methods for four dimensional İLTAREN data

Gradient Estimate Method	Number of Iterations	Number of Points
FDSA	600	100
SPSA	201	100

In Figure 4.10 an example of execution of the SPSA and FDSA algorithms is shown for four dimensional İLTAREN data. Both of the algorithms start from the same

initial point. Number of points required for convergence and number of function evaluations needed are given in Table 4-4.

4.2.2. ROSENBROCK FUNCTION

Rosenbrock function is generally used to test the performance of optimization algorithms. Three and four dimensional Rosenbrock function application results with two direct search algorithms Nelder-Mead simplex method, Hooke-Jeeves pattern search method and two search direction estimation algorithms, which are covariance analysis based and convex hull analysis based are reported in this part. Required number of function evaluations and number of sample points obtained during the execution of the algorithms are given. Selected stopping criterion of the algorithm is getting considerable improvement for the response parameter.

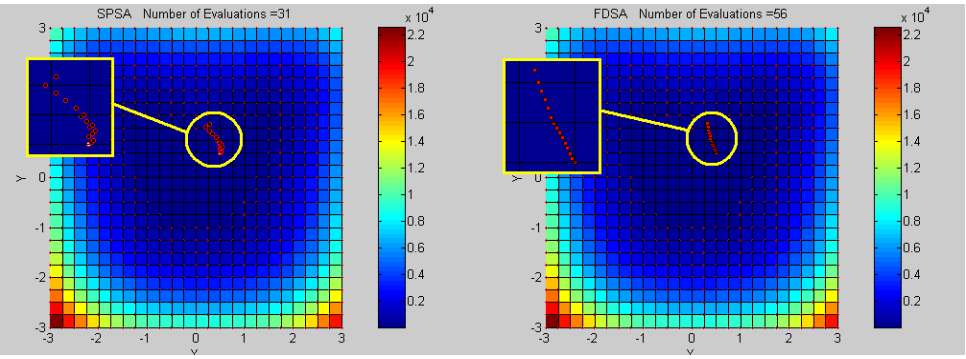

Figure 4.11: FDSA and SPSA algorithms for three dimensional Rosenbrock function

Table 4-5 : Results of FDSA and SPSA methods for three dimensional Rosenbrock function

Gradient Estimate Method	Average No of Evaluations	Average No of Points	Number of Executions
FDSA	56	14	20
SPSA	32	16	20

Results for the application of simultaneous perturbation stochastic approximation and finite difference stochastic approximation algorithms based gradient estimates to obtain a point satisfying constraints are also reported.

In Figure 4.11, an example of execution of the SPSA and FDSA algorithms is shown for the three dimensional Rosenbrock function. Two of the algorithms start from the same initial point and are executed 20 times. Average number of points required for convergence and number of function evaluations needed are given in Table 4-5.

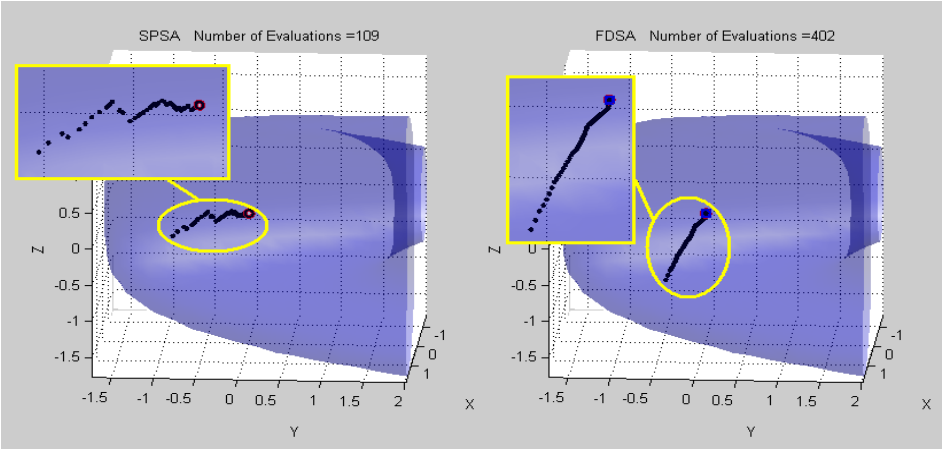

Figure 4.12: FDSA and SPSA algorithms for four dimensional Rosenbrock function

Table 4-6 : Results of FDSA and SPSA methods for four dimensional Rosenbrock function

Gradient Estimate Method	Number of Iterations	Number of Points
FDSA	402	67
SPSA	109	54

In Figure 4.12, an example of execution of the SPSA and FDSA algorithms is shown for four dimensional Rosenbrock function. Two of the algorithms start from the same initial point. Number of points required for convergence and number of function evaluations needed are given in Table 4-6.

SPSA algorithm for both three and four dimensional Rosenbrock function applications is more preferable as it is seen from the results given in Table 4-5 and Table 4-6. As the dimension of the problem increases, using FDSA algorithm for gradient estimation becomes more inefficient.

Application of the algorithm with Nelder-Mead and Hooke-Jeeves direct search methods are shown in Figure 4.13 and Figure 4.14, respectively. Covariance analysis based sampling strategy is utilized.

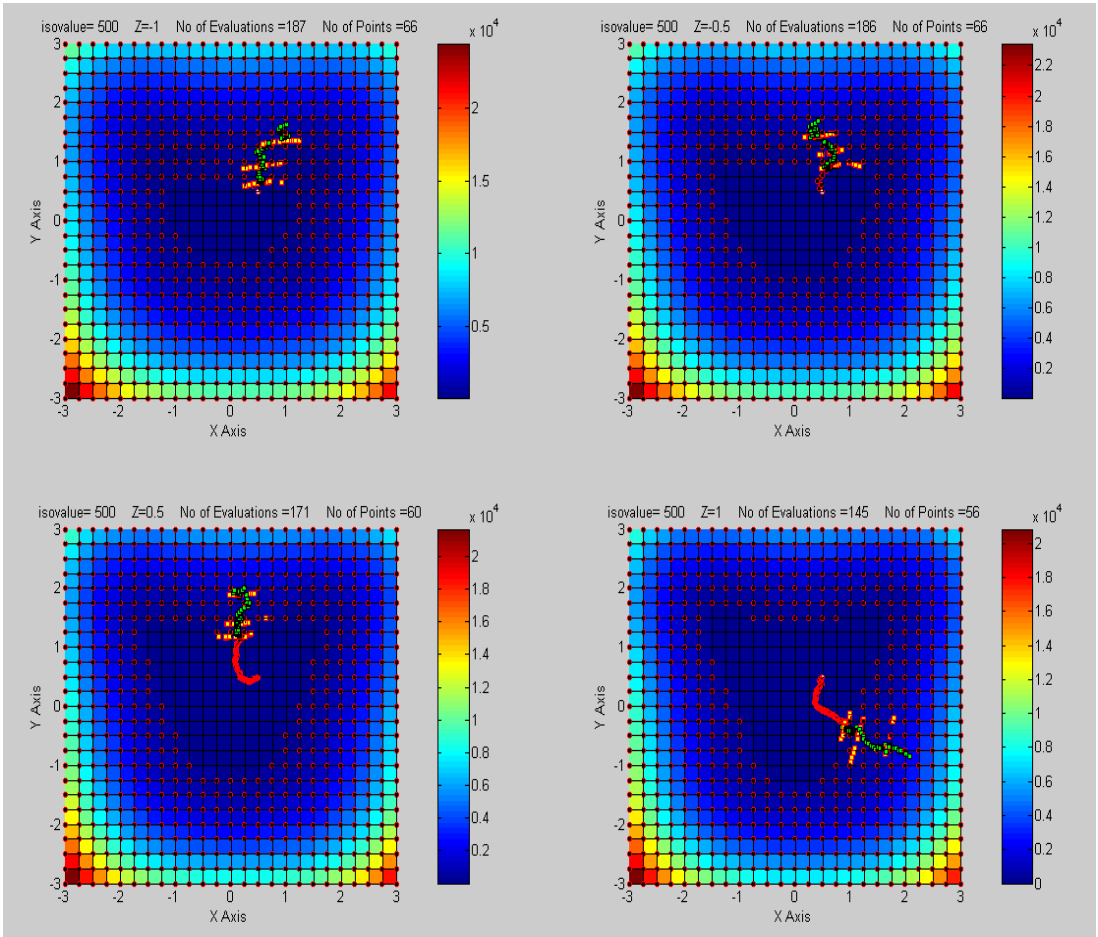

Figure 4.13: Application of the algorithm to three dimensional Rosenbrock function with Nelder-Mead simplex algorithm and covariance analysis based direction estimation

In Figure 4.13 and Figure 4.14, yellow points with red edges show the samples from constrained surfaces. Green points are evaluations for seeking local optima. Algorithm stops if response parameter gets the value of 750 or higher.

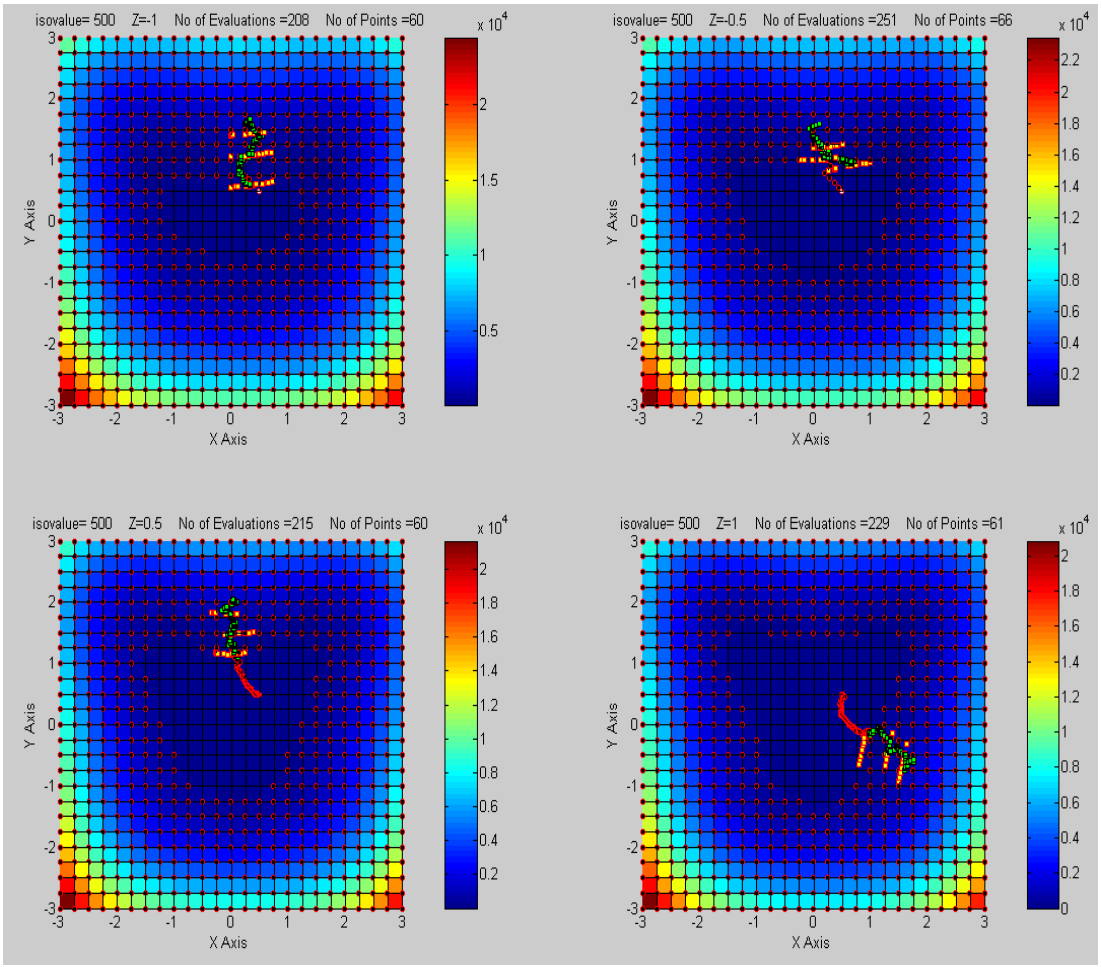

Figure 4.14: Application of the algorithm to three dimensional Rosenbrock function with Hooke-Jeeves pattern search algorithm and covariance analysis based search direction estimation

Table 4-7 : Results of three dimensional Rosenbrock function for two direct search methods

Direct Search Method	Number of Iterations Required	Number of Points Obtained
Nelder-Mead	689	248
Hooke-Jeeves	903	247

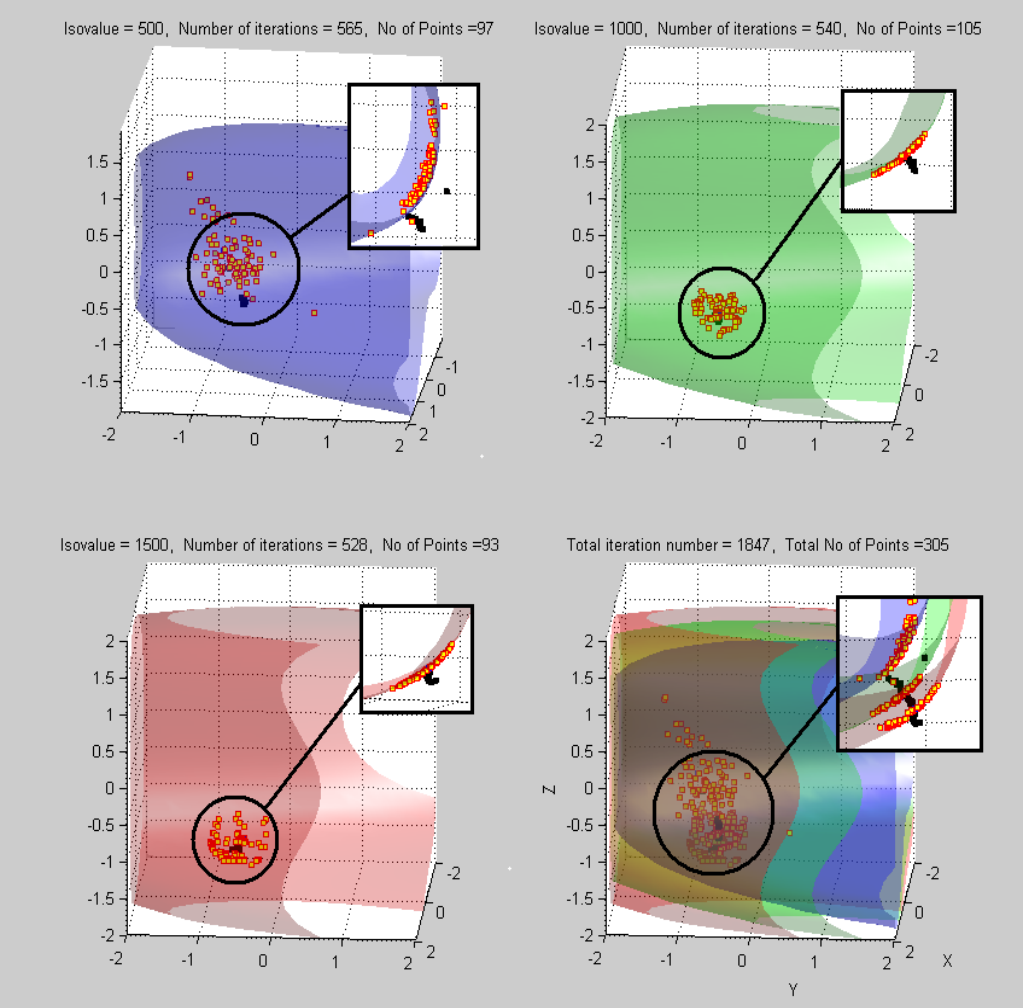

Figure 4.15: Application of the algorithm to four dimensional Rosenbrock function with Nelder-Mead simplex algorithm and convex hull based search direction estimation.

Table 4-8 : Results of four dimensional Rosenbrock function for two direct search methods

Direct Search Method	Number of Iterations Required	Number of Points Obtained
Nelder-Mead	1847	305
Hooke-Jeeves	2388	250

In Figure 4.15 and Figure 4.16, first three plots are for the isovalues 500, 1000 and 1500. These three plots are plotted together in the last. Samples from the three constrained surfaces are yellow points with red edges and black points are evaluation

points required for searching local optima. Start location for two of the applications is same. Algorithm stops if response parameter takes the value of 1800.

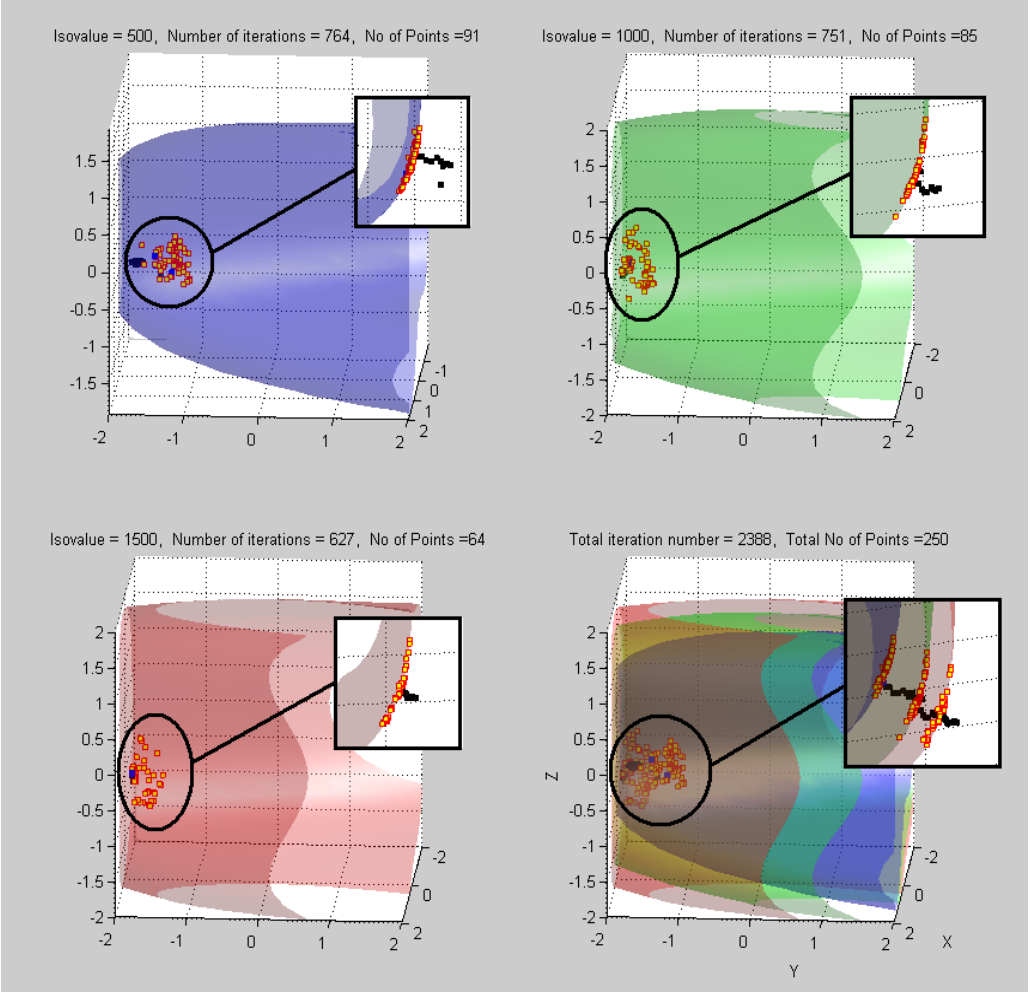

Figure 4.16: Application of the algorithm to four dimensional Rosenbrock function with Hooke-Jeeves pattern search algorithm and convex hull based search direction estimation.

As it is seen from the results given in Table 4.7 and 4.8, Nelder-Mead simplex method is more efficient according to the Hooke-Jeeves pattern search algorithm as a direct search method. The number of points obtained is close for two of the algorithms but Hooke-Jeeves algorithm requires more number of function evaluations.

4.2.3. STOCK EXCHANGE DATA

In Figure 4.17 an example of execution of the SPSA and FDSA algorithms is shown for four dimensional stock exchange data. Two of the algorithms start from the same initial point. Number of points required for convergence and number of function evaluations needed are given in Table 4-9.

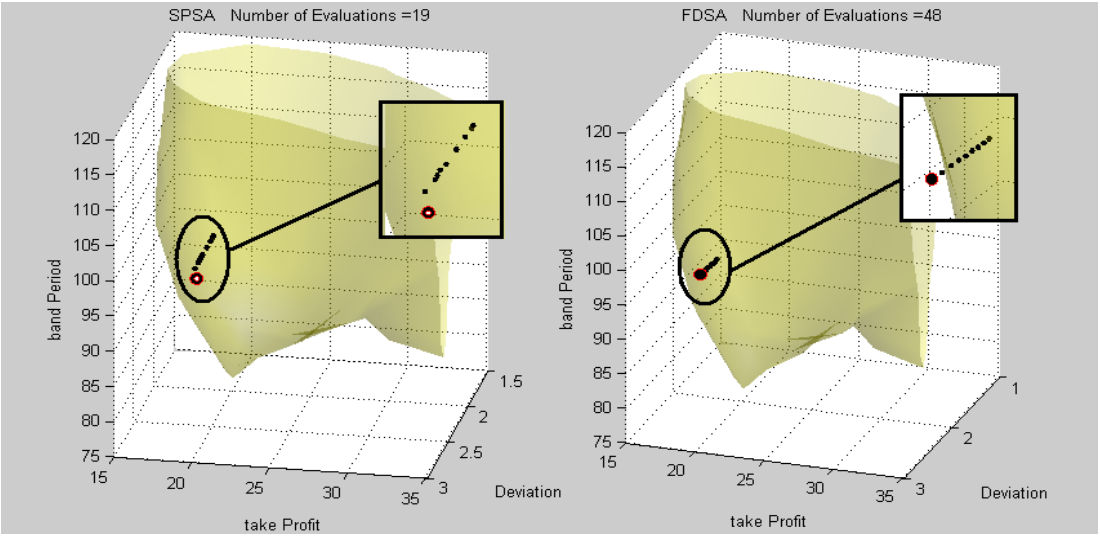

Figure 4.17: FDSA and SPSA algorithms for four dimensional stock exchange data

Table 4-9 shows the results obtained for four dimensional stock exchange data. The number of points obtained is nearly the same for FDSA and SPSA but the number of required function evaluations is approximately three times higher for FDSA.

Table 4-9 : Results of FDSA and SPSA methods for four dimensional stock exchange data

Gradient Estimate Method	Number of Iterations	Number of Points
FDSA	48	8
SPSA	19	9

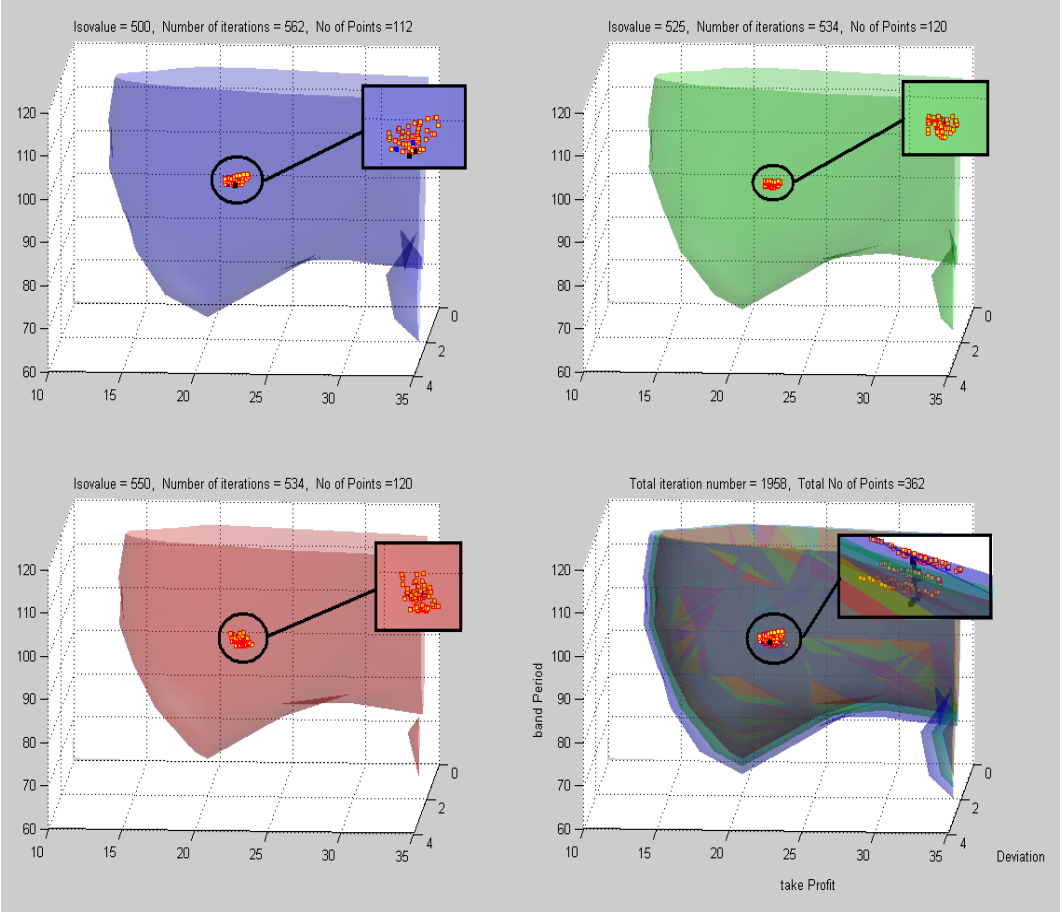

Figure 4.18: Application of the algorithm to four dimensional stock exchange data with Nelder-Mead simplex algorithm and convex hull based search direction estimation.

Table 4-10 : Results of four dimensional stock exchange data for two direct search methods

Direct Search Method	Number of Iterations Required	Number of Points Obtained
Nelder-Mead	1958	362
Hooke-Jeeves	4204	920

In Figure 4.18 and Figure 4.19, first three plots are plotted for the isovalues 500, 525 and 550. These three plots are plotted together in the last. Samples from the three constrained surfaces are yellow points with red edges and black points are evaluation

points required for searching local optima. Starting location for two of the applications is same. Algorithm stops if response parameter takes the value 600.

Selecting Nelder-Mead simplex algorithm as a direct search method for stock exchange data is more advantageous with respect to Hooke-Jeeves simplex search method from the results given in Table 4-10. Two of the algorithms converge to local optima but Nelder-Mead method requires smaller number of function evaluations.

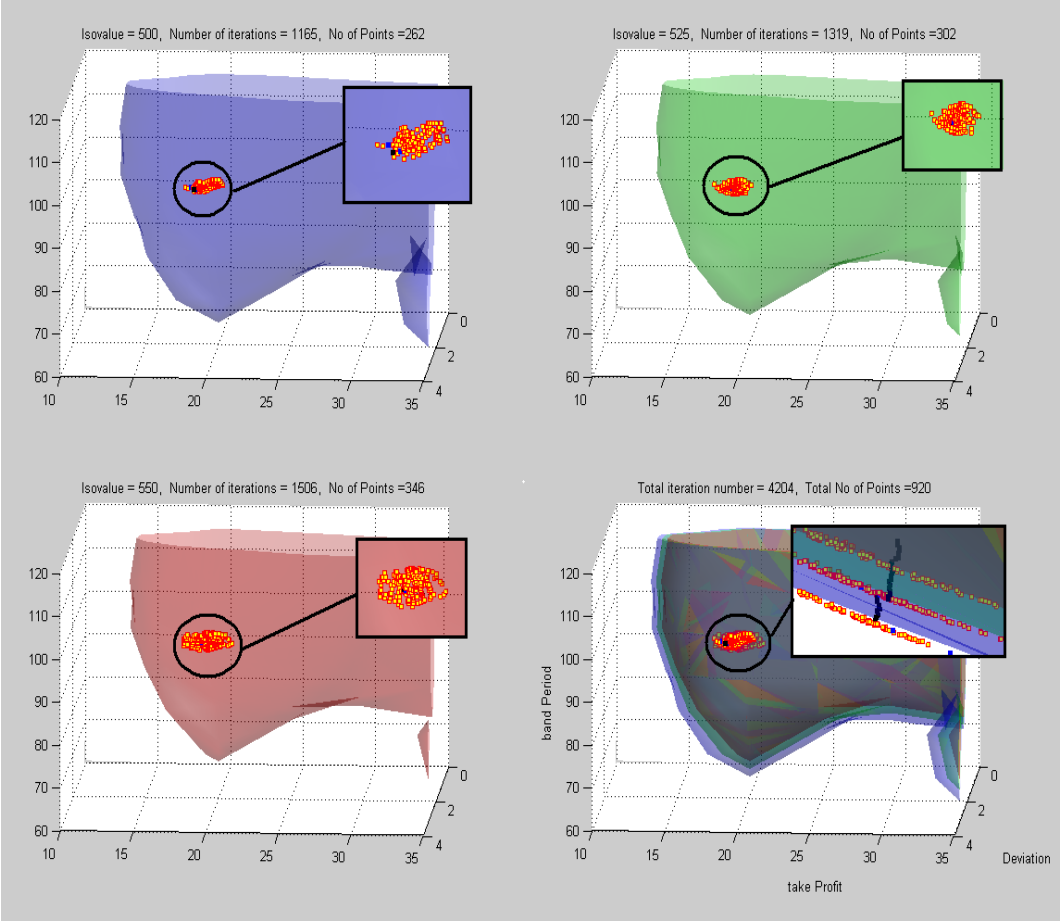

Figure 4.19: Application of the algorithm to four dimensional stock exchange data with Hooke-Jeeves pattern search algorithm and convex hull based search direction estimation.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. CONCLUSIONS

For the approach investigated in this study, we have in mind problems where the only information available is the possibility to evaluate the computational or simulation model, and each evaluation is computationally expensive and no other additional information is available. Moreover, the problem can be extended with some costly or non-costly constraints. The problem of finding an optimum setting for possibly large number of design variables requiring minimum number of evaluations is considered in this study.

These types of problems are common in black-box design optimization. Classical optimization methods, based on derivatives, are not applicable because the derivative information is not available or will be quite costly to approximate. The aim is to require as few function evaluations as possible to obtain convergence to a local minimizer or maximizer subject to given constraints.

The algorithm that is presented in this study is based on direct search methods and stochastic approximation algorithms available in the literature. Penalty and barrier methods have also been utilized to handle costly constraints in the problem.

In this context, stochastic approximation algorithms, simultaneous perturbations and finite differences, have been examined for the evaluation of the cost function. These methods utilize the approximation of the derivative. Some derivative free, direct search algorithms are also examined. These are Nelder-Mead simplex algorithm and Hooke-Jeeves pattern search algorithm.

Represention of a local approximate surface model, which is both inexpensive and accurate, is also one of the objectives of this study. That approximate model captures the relationship between input and output. Such an approximation function is called a response surface model (also known as metamodels, or surrogate models). The response surface model is an interpolation based Radial Basis Functions (RBFs). Radial basis functions as interpolants are preferred since their interpolation properties are very suitable. Specifically, it is trivial to construct local interpolations using RBF. The Orthogonal Least Squares (OLS) algorithm is a greedy algorithm which chooses the suitable centers for radial basis functions systematically considering the individual contribution to error reduction.

Consequently, this study presents an iterative optimization methodology utilizing direct search methods and stochastic approximation algorithms for solving optimization problems with expensive-to-evaluate cost functions and/or constraints. Applications of this algorithm to three test problems are presented and some numerical results have been obtained. Rosenbrock function, a real problem provided by TÜBİTAK-İLTAREN and data set of stock exchange strategy are the selected test problems for the algorithm.

5.2. FUTURE WORK

Although the algorithms used for sampling the constrained surfaces have promising results, some extensions may be necessary to improve the performance of the sampling.

Convex hull based sampling algorithm works efficiently for convex regions of the surface to be sampled, however it is observed that sampling density for non-convex regions of the surface may not be satisfactory for some applications. Detecting these regions and executing the algorithm again for these parts to increase sampling density is left as a future work.

The covariance analysis based sampling algorithm performs more satisfactory results if the smoothness of the surface to be sampled is high. The application of that algorithm to relatively non-smooth surfaces may require some modifications. Sharp edges have to be detected and direction at these regions has to be calculated rigorously. A new direction estimation method may be developed to execute at sharp edges.

REFERENCES

[1] J. E. Dennis, Jr and V. Torczon, “Direct search methods on parallel machines”, SIAM J. Optimization. Vol. 1, No. 4, pp. 448–474, 1991.

[2] V. Torczon, “Pattern search methods for nonlinear optimization”, SIAG/OPT Views and News, Vol. 6, pp. 7–11, 1995.

[3] V. Torczon, and M.W. Trosset, “From evolutionary operation to parallel direct search: pattern search algorithms for numerical optimization”, Computing Science and Statistics, Vol. 29, pp. 396–401, 1998.

[4] M.W. Trosset, “I know it when I see it: toward a definition of direct search methods”, SIAG/OPT Views and News, Vol. 9, pp. 7–10, 1997.

[5] H.J. Kushner and D.S. Clark, “Stochastic Approximation Methods for Constrained and Unconstrained Systems”, Springer-Verlag, New York, 1978.

[6] B.T. Polyak, “Introduction to Optimization”, Optimization Software, Inc., New York, 1987.

[7] F. Glover, J.P. Kelly, and M. Laguna, “New advances and applications of combining simulation and optimization”, Proceedings of the 1996 Winter Simulation Conference, pp. 144–152, 1996.

[8] J.E. Dennis, Jr. and H.F. Walker, “Inaccuracy in quasi-Newton methods: local improvement theorems. Mathematical Programming Study”, Vol. 22, pp. 70–85, 1984.

[9] J.A. Nelder and R.Mead, “A simplex method for function minimization”, Computer Journal, Vol. 7, No. 4, pp. 308-313, 1 January1965.

[10] R. Hooke and T.A. Jeeves. “Direct search solution of numerical and statistical problems” Journal of the Association of Computing Machinery, Vol. 8, pp. 221–224, 1961.

[11] W. Spendley, G. R. Hext, and F. R. Himsworth, “Sequential application of simplex designs in optimization and evolutionary operation”, Vol.4, pp. 441–461, 1962.

[12] J. Kiefer, J. Wolfowitz, “Stochastic estimation of a regression function”, Ann. Math. Stat., vol. 23, pp. 462-466, 1952.

[13] J.C. Spall, “An Overview of the Simultaneous Perturbation Method for Efficient Optimization”, Johns Hopkins APL Technical Digest, Vol. 19, No. 4, 1998.

[14] J.C. Spall, “Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation”, IEEE Trans. on Automatic Control, Vol. 37, No. 3, 1992.

[15] A. Isaacs, “Direct-search methods and DACE”, Seminar Report, Department of Aerospace Engineering, Indian Institute of Technology, Bombay, 2003.

[16] J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, “Convergence properties of the Nelder-Mead simplex method in low dimensions”, SIAM J. Optim. , Vol. 9, No. 1, pp. 112-147, 1998.

[17] R.M. Lewis, V. Torczon, M.W. Trosset, “Direct Search Methods: Then and Now”, NASA/CR-2000-210125, ICASE Report No. 2000-26, 2000.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Surface reconstruction from unorganized points”, In Proceedings of ACM SIGGRAPH 92, pp.71-78, 1992.

[19] E. Shaffer, M. Garland, “Efficient Adaptive Simplification of Massive Meshes”, In VIS '01: Proceedings of the conference on Visualization '01, pp. 127-134, 2001

[20] R.H. Myers, D.H. Montgomery, “Response Surface Methodology”, Wiley, New York, 1995.

[21] K.M. Carley, N.Y. Kamneva, J. Reminga, “Response Surface Methodology”, CASOS Technical Report, October, 2004.

[22] C. Shen, J.F. O'Brien, and J.R. Shewchuk, "Interpolating and approximating implicit surfaces from polygon soup", In Proceedings of ACM SIGGRAPH, pp. 896-904, 2004.

[23] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.P. Seidel, "Multi-level partition of unity implicits", ACM Transactions on Graphics, Proceedings of SIGGRAPH 2003, Vol. 22, No. 3, pp. 463-470, 2003.

[24] G. Turk, and J. O'Brien, "Variational implicit surfaces", Technical Report GITGVU-99-15, Georgia Institute of Technology, 1998.

[25] G. Turk, and J. O'brien, "Modelling with implicit surfaces that interpolate", Transactions on Graphics, Vol.21, pp. 855-873, 2002.

[26] B. Morse, T.S. Yoo, P. Rheingans, et al. "Interpolating implicit surfaces from scattered surfaces data using compactly supported radial basis functions", In Proceedings of Shape Modeling International, pp. 89-98, 2001.

[27] S. Chen, C.F.N. Cowan, and P.M. Grant, "Orthogonal least square learning algorithm for radial basis function networks", IEEE Transactions on Neural Network, Vol. 2, No. 2, pp. 302-309, 1991.

[28] N. Amenta, M. Bern, and M. Kamvysselis, "A new Voronoi-based surface reconstruction algorithm", In Proceedings of SIGGRAPH'98, pp. 415-421, 1998.

[29] N. Amenta, S. Choi, and R. Kolluri, "The power crust", In Proceedings of 6th ACM Symposium on Solid Modeling, pp. 249-260, 2001.

[30] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans, "Reconstruction and representation of 3D objects with radial basis functions", In Proceedings of ACM SIGGRAPH 2001, pp. 67-76, 2001.

[31] P. Tobor, P. Reuter, and C. Schilck, "Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions", Journal of WSCG 2004, Vol. 12, pp. 467-474, 2004.

[32] J.R. Blum, "Multidimensional stochastic approximation methods", Ann. Math. Stat., vol. 25, pp. 737-744, 1954.

[33] R.M. Freud, “Penalty and Barrier Methods for Constrained Optimization”, Massachusetts Instititute of Technology, 2004.

[34] J. Wang, J.C. Spall, “A Constrained Simultaneous Perturbation Stochastic Approximation Algorithm Based on Penalty Functions”, Proceedings of the American Control Conference, San Diego, California, Vol. 1, pp.393-399, 1999.

[35] D.G. Humphrey, J.R. Wilson, “A Revised Simplex Search Procedure for Stochastic Simulation Response-Surface Optimization”, Proceedings of the 1998 Winter Simulation Conference, Vol.1, pp. 751-759, 1998.

[36] MathWorks, www.mathworks.com, last access time is 17.08.2009.

[37] Ö. Yeniay, “Penalty Function Methods for Constrained Optimization with Genetic Algorithms”, Mathematical and Computational Applications, Vol. 10, No. 1, pp. 45-56, 2005.

[38] J.E. Kack, “Constrained Global Optimization with Radial Basis Functions”, Research Report MdH-IMa-2004, Department of Mathematics and Physics, Malardalen University, September 2004.

[39] A.E. Smith, D.W. Coit, “Penalty Functions”, Section C 5.2 of Handbook of Evolutionary Computation, Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania, September 1995.

[40] http://en.wikipedia.org , last access date is 27.08.2009.

[41] http://www.answers.com , last access date is 12.10.2009.

[42] R. Brekelmans, L. Driessen, H. Hamers, D. den Hertog, “Constrained Optimization Involving Expensive Function Evaluations: A Sequential Approach”, ISNN 0924-7815, November 2001.

[43] Advance Design and Optimization Technologies, http://www.adoptech.com, last access date is 09.10.2009.

[44] E.J. Anderson, M.C. Ferris, “A Direct Search Algorithm for Optimization with Noisy Function Evaluations”, SIAM J. Optim., Vol. 11, pp. 837-857, 2001.

[45] J.L. Maryak, D.C. Chin, “Global Random Optimization by Simultaneous Perturbation Stochastic Approximation”, Proceedings of the American Control Conference, Vol. 34, pp. 441-466, June 2001.

[46] J.C. Spall, “Stochastic Optimization and the Simultaneous Perturbation Method”, Proceedings of the 1999 Winter Simulation Conference, pp. 101-109, 1999.

APPENDIX A

GRAM SCHMIDT ALGORITHM

In mathematics, particularly in linear algebra and numerical analysis, the Gram–Schmidt process is a method for orthonormalizing a set of vectors in an inner product space, most commonly the Euclidean space R^n. The Gram–Schmidt process takes a finite, linearly independent set $S = \{v_1, v_2, ..., v_k\}$ for $k \leq n$ and generates an orthogonal set $S' = \{u_1, u_2, ..., u_k\}$ that spans the same k-dimensional subspace of R^n as S.

We define the projection operator by:

$$proj_u(v) = u_2 - \frac{\langle u, v \rangle}{\langle u, u \rangle} \cdot u \qquad \text{(A.1)}$$

where $\langle u, v \rangle$ denotes the inner product of the vectors *u* and *v*. This operator projects the vector *v* orthogonally onto the vector *u*. The Gram–Schmidt process then works as follows:

Step1: Let $u_1 = v_1$

Step2: $u_2 = v_2 - proj_{u_1}(v_2)$

Step3: $u_3 = v_3 - proj_{u_1}(v_3) - proj_{u_2}(v_3)$

Step4: $u_4 = v_4 - proj_{u_1}(v_4) - proj_{u_2}(v_4) - proj_{u_3}(v_4)$

⋮

Stepk: $u_k = v_k - \sum_{j=1}^{k-1} proj_{u_j}(v_k)$

The resulting orthogonal set $\{u_1, u_2, ..., u_k\}$ forms an orthogonal basis. The material of this part is primarily based on [40].

APPENDIX B

PENALTY METHODS

Penalty functions have been a part of the literature on constrained optimization for decades [39]. In general, a penalty function approach is as follows: given an optimization problem,

$$\begin{aligned} &\min f(\theta) \\ &s.t.\ g_1(\theta) \le 0 \\ &\quad g_2(\theta) \le 0 \\ &\quad\ \vdots \\ &\quad g_p(\theta) \le 0 \end{aligned} \tag{B.1}$$

where $f : R^n \to R, \quad g_i : R^n \to R,\ i = 1,..., p$. Considering only inequality constraints is not restrictive, because an equality constraint of the form $h(\theta) = 0$ is equivalent to two inequality constraints $h(\theta) \le 0, \quad and \quad -h(\theta) \le 0$. We now discuss a method for solving the above constrained optimization problem using techniques from unconstrained optimization. Specifically, we approximate the constrained optimization problem above by an unconstrained optimization problem

$$\min\ f(\theta) + \gamma P(\theta) \tag{B.2}$$

where $\gamma \in R$ is a positive constant, and $P : R^n \to R$ is a given function.

We then solve the associated unconstrained optimization problem, and use the solution as an approximation to the minimizer of the original problem. The constant γ is called the penalty parameter, and the function P is called the penalty function. Formally, a function $P : R^n \to R$ is called a penalty function for the above constrained optimization problem if it satisfies the following three conditions;

- P is continuous;
- $P(\theta) \geq 0$ for all $\theta \in R^n$;
- $P(\theta) = 0$ iff θ is feasible, that is, $g_1(\theta) \leq 0, ..., g_p(\theta) \leq 0$.

Clearly, for the above unconstrained problem to be a good approximation to the original problem, the penalty function P must be appropriately chosen. The role of the penalty function is to "penalize" points that are outside the feasible set. Therefore, it is natural that the penalty function be defined in terms of the constraint functions $g_1, ..., g_p$. A possible choice for P is

$$P(\theta) = \sum_{i=1}^{p} g_i^+(\theta) \tag{B.3}$$

where

$$g_i^+(\theta) = \max(0, g_i(\theta)) = \begin{Bmatrix} 0 & if \ \ g_i(\theta) \leq 0 \\ g_i(\theta) & if \ \ g_i(\theta) > 0 \end{Bmatrix} \tag{B.4}$$

The penalty function method for solving constrained optimization problems involves constructing and solving an associated unconstrained optimization problem, and using the solution to the unconstrained problem as the solution to the original constrained problem. Of course, the solution to the unconstrained problem (the approximated solution) may not be exactly equal to the solution to the constrained problem (the true solution). Whether or not the solution to the unconstrained problem

is a good approximation to the true solution depends on the penalty parameter γ and the penalty function P. We would expect that larger the value of the penalty parameter γ, the closer the approximated solution will be to the true solution, because points that violate the constraints are penalized more heavily. Ideally, in the limit as $\gamma \to \infty$, the penalty method should yield the true solution to the constrained problem. The material of this part is based on the references; [33], [37], [39].