Jean Bosco NDIKUBWIMANA

Finding local optima for costly objective functions (using Radial Basis Functions)

UNIVERSITY OF RWANDA-HUYE CAMPUS
College of Science and Technology
School of Pure and Applied Science
Department of Applied Mathematics

A Thesis Submitted in Partial Fulfillment of the Requirements for
The degree of Master of Science in Applied Mathematics
At the University of Rwanda-Huye Campus
In Collaboration with Linköping University, Sweden

Supervisory Committee:
Supervisor: Dr. Nils-Hassan Quttineh

Master Degree Program in Applied Mathematics
Specialization: Statistical modeling and actuarial science
Department of Applied Mathematics
Faculty of Science
University of Rwanda-Huye Campus

This master degree program thesis is funded by UR/SIDA

Huye, January 2014

Declaration

This is to certify that this thesis is my own work, that due reference has been made in the text to all other material used, that it is less than 20, 000 words in length, exclusive of figures, tables and bibliographies, and that it has not been previously submitted for any comparable academic award.

Student's names :NDIKUBWIMANA Jean Bosco.....
Date :.....

Dedication

I dedicate this thesis To
My Lord and Redeemer Jesus Christ,
My Parents, my brothers and sisters,
My beloved wife Angélique DUKUNDE,
My daughter Cartel INEZA GANZA,
My classmates

Acknowledgements

I would like to thank my supervisor Dr. Nils-Hassan Quttineh for his warm encouragement and thoughtful guidance. He is responsible for involving me in the field of optimization and in particular MATLAB computations which has been a tough part of this thesis. My special thanks go to the government of Sweden for sponsoring our program to this level; we would also like to extend our appreciation to all lectures of Linköping University (LIU) and University of Rwanda (UR) and more specifically those of the Departments of Applied Mathematics, Statistics and Pure Mathematics and the lectures from Lund University and Stockholm University.

Abstract

The present work deals with optimization of costly objective functions. The thesis objective is to minimize a function $f(x)$ which is costly to evaluate. In this context costly means time consuming, and one single function evaluation could require several hours of computation time. We assume that neither analytical expression of $f(x)$ can be found, nor can we find derivatives. Such functions are often referred to as 'black-box'. One way of dealing with such optimization problems is to sample the variable space and build surrogate models based on Radial Basis function(RBF), which are sufficient to predict the output of an expensive computer code. In the present thesis we discuss two different iterative methods that are often used in optimization to find the approximate local optima to black box function. Thereafter we show that surrogate models based on RBF contribute more and more accurately to the problem in terms of time than the two analytic methods.

Keywords: Radial Basis Functions, local optima, Costly objective functions.

List of Figures

3.1 *The costly objective function $f(\mathbf{x})$ on the left and the surrogate model based on the number of sampled points $N = 30$ to the right.* 13
3.2 *The figures representing the true local optima (TL) marked on the true costly function on the left and the interpolated local optima (IL) marked on the interpolation surface with $N = 100$ to the right.* 16
3.3 *The figures representing the true local optima (TL) marked on the true costly function on the left and the interpolated local optima (IL) marked on the interpolation surface with $N = 70$ to the right.* 17

List of Tables

1.1 *Different choice of radial basis functions* 5

2.1 *Convergence of iterative methods namely SD and NM applied to the same convex objective function.* 11

3.1 *Distance measure between true and interpolated local optima using the starting point* $\mathbf{x}^0 = [0.5, 0.5]^T$. 15
3.2 *Distance measure between true and interpolated local optima with starting point* $\mathbf{x}^0 = [0; 0]^T$. 15
3.3 Distance measure between true (**TL**) and interpolated local optima (**IL**). 16
3.4 *Distance measure between true and interpolated local optima with starting point* $\mathbf{x}^0 = [0.5; 0.5]$. 17

Contents

Declaration **i**

Dedication **ii**

Acknowledgements **ii**

Abstract **iii**

list of figures **iv**

list of tables **1**

table of contents **1**

Basic concepts **3**

1 Basic concepts **4**
1.1 Introduction 4
1.1.1 The interpolation problem 4
1.2 Interpolation using RBF 5
1.3 The derivatives of the interpolant 6
1.3.1 The first derivative of $s_n(x)$ or (Gradient) 6
1.3.2 The second derivative of $s_n(\mathbf{x})$ or (Hessian) 6

2 Iterative methods in optimization **7**
2.1 Necessary and sufficient conditions of optimality 7
2.1.1 Necessary conditions 7
2.1.2 Sufficient conditions 8
2.2 The steepest descent method (SD) 9
2.3 Newton's method (NM) 9
2.3.1 Steepest descent and Newton algorithms 10
2.4 Numerical tests for the NM and SD for finding optima 11

3 Finding local optima for costly objective functions using RBF. 13
3.1 The surrogate modeling 13
3.2 Finding local optima using RBF 14
3.3 Conclusions 18

APPENDICES 19

A MATLAB codes 20

Chapter 1

Basic concepts

1.1 Introduction

In current days, several applications in science and engineering are related to optimization with so called costly problems, where costly means time consuming. Such a problem can be viewed as a problem of optimizing costly objective functions and one single function evaluation could take several hours of computation time. In many applications, the objective function typically is non-convex and non-linear. This implies that the number of local optima to a given optimization problem could be large. The optimization problem can be formulated as:

$$\min f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d. \tag{1.1}$$

We assume that neither analytical expressions of $f(\mathbf{x})$ can be found nor we can find derivatives, which lead to the fact that standard non-linear programming methods have a hard time to find the local optima. One way of handling this kind of problem, is to sample the variable space and build an interpolation surface $s(x)$ using so called "**Radial Basis Functions (RBF)**." Using this surface as guidance, new sample points are selected which (hopefully) contribute to a more and more accurate interpolation surface. The optimization is then performed on $s(\mathbf{x})$ instead of $f(\mathbf{x})$, and since $s(\mathbf{x})$ has an analytical representation, such function evaluations are not costly.

1.1.1 The interpolation problem

The problem considered in this thesis is described as follows:

i. Given is a set of data points whose elements consist of paired values of the independent variables (a given vector $\mathbf{x}$) and the dependent variable (a scalar y) denoted as $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{n}$, where n is the number of input points and where the vector $\mathbf{x}$ is $\mathbf{x} = [x_1, \cdots, x_d]$ with d the dimension.

ii. Find a closed form approximate function $s(\mathbf{x})$ of the dependent variable $y^{(i)} = f(\mathbf{x}^i)$ and its closed form approximate derivative functions.

iii. Use the analytical expression of $s(\mathbf{x})$ and its derivatives in standard non-linear programming methods to find a local optima of the given costly objective function $f(\mathbf{x})$.

1.2 Interpolation using RBF

The "**Radial Basis Functions (RBF)**" are means to get an approximation of a function of several variables, for instance $\mathbf{x} \in \mathbb{R}^d$. That is, given n distinct points $\mathbf{x} \in \Omega$, with the evaluated function values $\mathbf{y}$, the radial basis function interpolant $s_n(\mathbf{x})$ has the form (see in [1]):

$$s_n(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi(\|\mathbf{x}^i - \mathbf{x}\|_2) + b^T\mathbf{x} + a. \tag{1.2}$$

where $\lambda \in \mathbb{R}^n, b \in \mathbb{R}^d, a \in \mathbb{R}$ where $\phi(r), r \geq 0$ is some radial function. Parameters λ, b and a are found by solving the symmetric linear system

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix} \tag{1.3}$$

where Φ is a $n \times n$ matrix with entries $\Phi_{ij} = \phi(\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2)$ and

$$P = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_n^T & 1 \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}, c = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \\ a \end{pmatrix}, y = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

In the following table we have commonly used radial basis functions:

RBF	$\phi(r), r \geq 0$
Gaussian	$\phi(r) = e^{(-\varepsilon r)^2}, \varepsilon > 0$
Inverse multiquadric	$\phi(r) = \frac{1}{(r^2+\varepsilon^2)^{\frac{1}{2}}}, \varepsilon > 0$
multiquadric	$\phi(r) = (r^2 + \varepsilon^2)^{\frac{1}{2}}, \varepsilon > 0$
Cubic	$\phi(r) = r^3$
Linear	$\phi(r) = r$
thin plate spline	$\phi(r) = r^2 \log r$

Table 1.1: *Different choice of radial basis functions*

The choice of ε influences both accuracy and numerical stability of the solution to our interpolation problem and we need to choose a basis function so that the system (2.3) is non-singular. Once this is satisfied we have a unique radial basis function interpolant to our objective function f at the points $\{x_1, \cdots, x_n\}$. For the thesis we choose to use the cubic spline $\phi(r) = r^3$. The method of interpolation using RBF is based on general techniques proposed by Jones [3]. In [1, 2] it is again well discussed and used to find some results to optimization problems. However, in [2] it is also tested on some optimization functions such as the Dixon-Szegö test function, while in [1] we can find a good comparison of the method to for instance DACE interpolation.

1.3 The derivatives of the interpolant

We have assumed that neither analytical expression of $f(\mathbf{x})$ can be found nor can we find its derivatives. But since the objective function can be interpolated, and the interpolant has an analytic representation, we can also find the derivatives of the interpolant. In this section we will derive first and the second derivatives of the interpolant.

1.3.1 The first derivative of $s_n(x)$ or (Gradient)

The interpolant is

$$s_n(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi(\|\mathbf{x}^i - \mathbf{x}\|_2) + b^T\mathbf{x} + a. \tag{1.4}$$

Since it is continuously differentiable and since we chose $\phi(r) = r^3$, then its gradient is derived from

$$\nabla s_n = (\frac{\partial s_n}{\partial x_1}, \cdots, \frac{\partial s_n}{\partial x_d})^T \tag{1.5}$$

and the gradient in the vector form is given by

$$\nabla s_n(\mathbf{x}) = -3\sum_{i=1}^{n} \lambda_i \cdot \|\mathbf{x}^i - \mathbf{x}\|_2 \cdot (\mathbf{x}^i - \mathbf{x}) + b. \tag{1.6}$$

1.3.2 The second derivative of $s_n(\mathbf{x})$ or (Hessian)

From the first derivative we have that

$$\nabla^2 s_n(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \cdot \Theta(\mathbf{x}^i - \mathbf{x}) \tag{1.7}$$

where

$$\Theta(r) = \begin{cases} \sum_{i=1}^{n} \lambda_i \cdot \frac{\phi'(\|r\|_2)}{\|r\|_2} I_n + (\phi''(r) - \frac{\phi'(\|r\|_2)}{\|r\|_2}) \cdot \frac{rr^T}{\|r\|_2^2}, & r \neq 0 \\ \phi''(0) I_n, & r = 0 \end{cases} \tag{1.8}$$

and where I_n is the $n \times n$ identity matrix. Replacing $\phi(r)$ by the chosen basis function we have that (1.7) becomes

$$\nabla^2 s_n(\mathbf{x}) = 3\sum_{i=1}^{n} \lambda_i \cdot (\|\mathbf{x}^i - \mathbf{x}\|_2 + \frac{(\mathbf{x}^i - \mathbf{x})(\mathbf{x}^i - \mathbf{x})^T}{\|\mathbf{x}^i - \mathbf{x}\|_2}) \tag{1.9}$$

assuming that $\|\mathbf{x}^i - \mathbf{x}\|_2$ is not equal to zero. We have now derived the derivatives of the interpolant, now let us introduce some iterative methods where these expressions can be used.

Chapter 2

Iterative methods in optimization

The iterative methods that we will use in this work are "Newton's method" and "the steepest descent" method. These methods will first be discussed and tested to a given convex function having a known local optima for a given starting point and analyse the convergence. Next we will use these methods and the expressions of the interpolant derived in the previous sections. But first, let us recall some definitions. Let us assume that the function f is defined over $\Omega \subset \mathbb{R}^d$.

Definition 2.1 *A point $\boldsymbol{x}^* \in \Omega$ is said to be a relative minimum point or a local minimum point of the function f, if there is an $\varepsilon > 0$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$ such that $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon$. If the inequality is strict for $\boldsymbol{x} \neq \boldsymbol{x}^*$, then $\boldsymbol{x}^*$ is said to be a strict local minimum point.*

Definition 2.2 *A point $\boldsymbol{x}^* \in \Omega$ is said to be a global minimum point of the function f, if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$. If the inequality is strict for $\boldsymbol{x} \neq \boldsymbol{x}^*$, then $\boldsymbol{x}^*$ is said to be a strict global minimum point.*

Definition 2.3 *A direction $\boldsymbol{d}^k$, such that $\nabla f(\boldsymbol{x}^k)^T \boldsymbol{d}^k < 0$ is called a descent direction and is a direction in which $f(\boldsymbol{x})$ decreases.*

The two algorithms we will consider in this part of the thesis converges to a local minimum.

2.1 Necessary and sufficient conditions of optimality

2.1.1 Necessary conditions

Let f be twice continuously differentiable. We will use Taylor's theorem in a simple way to show that the gradient of f vanishes at a local minimizer and the Hessian is positive semi-definite. These *are the necessary conditions* for optimality.

Theorem 2.1 *(Necessary condition) Let f be twice continously differentiable and let $\boldsymbol{x}$ be a local minimizer of f. Then*

$$\nabla f(\boldsymbol{x}^*) = 0. \tag{2.1}$$

Moreover, $\nabla^2 f(\boldsymbol{x})$ is positive semi definite.

Proof. Given $u \in \mathbb{R}^d$. By Taylor's theorem we have that for sufficiently small t, that $0 \leq f(\mathbf{x}^* + tu) - f(\mathbf{x})$ and hence

$$\nabla f(\mathbf{x}^*)u + \frac{t}{2}u^T\nabla^2 f(\mathbf{x}^*)u + o(t) \geq 0. \tag{2.2}$$

For sufficiently small t and for all $u \in \mathbb{R}^d$. Setting $t = 0$ and $u = -\nabla f(\mathbf{x}^*)$ we get

$$\|\nabla f(\mathbf{x}^*)\| = 0. \tag{2.3}$$

Include (2.3) in (2.2) we get

$$\frac{t}{2}u^T\nabla^2 f(\mathbf{x}^*)u + o(t) \geq 0 \tag{2.4}$$

and finally dividing (2.4) by t we obtain

$$\frac{1}{2}u^T\nabla^2 f(\mathbf{x}^*)u \geq 0.$$

for all $u \in \mathbb{R}^d$. ■

2.1.2 Sufficient conditions

A stationary point need not be a minimizer. To obtain a minimizer we must require the second derivative to be nonnegative. These are the *sufficient conditions* for optimality.

Theorem 2.2 *(Sufficient condition) Let f be twice continuously differentiable in a neighborhood of $\boldsymbol{x}^*$. Assume that $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite. Then $\boldsymbol{x}^*$ is a local minimizer of f.*

Proof. Let $u \neq 0, u \in \mathbb{R}^d$. Now, for sufficiently small t we have

$$\begin{aligned} f(\mathbf{x}^* + tu) &= f(\mathbf{x}^*) + t\nabla f(\mathbf{x}^*)^T u + \frac{t^2}{2}u^T\nabla^2 f(\mathbf{x}^*)u + o(t^2) \\ &= f(\mathbf{x}^*) + \frac{t^2}{2}u^T\nabla^2 f(\mathbf{x}^*)u + o(t^2). \end{aligned}$$

Hence, for the smallest eigenvalue $\lambda > 0$ of $\nabla^2 f(\mathbf{x}^*)$ we have:

$$f(\mathbf{x}^* + tu) - f(\mathbf{x}^*) \geq \frac{\lambda}{2}\|tu\|^2 + o(t^2) > 0. \tag{2.5}$$

for t sufficiently small. Thus $\mathbf{x}$ is a local minimizer of f. ■

For solving unconstrained optimization problems we always use search methods. The main difference between these methods is the selection of the search direction. Let us study the

first order Taylor's series approximation of the function $f(\mathbf{x})$ in the points $\mathbf{x}^k$. We get the approximation

$$f_1(\mathbf{x}) = f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) \tag{2.6}$$

In a minimization problem we are interested in each iteration to find a point $\mathbf{x}$ such that $f(\mathbf{x}) < f(\mathbf{x}^k)$. From the first order approximation we observe that we can find such points in search directions $\mathbf{d}^k = (\mathbf{x} - \mathbf{x}^k)$ where $\nabla f(\mathbf{x}^k)^T\mathbf{d}^k < 0$. Generally, any descent direction can be used as a search direction in a search method.

2.2 The steepest descent method (SD)

The steepest descent method is a method where the search direction is computed only by using the gradient $\nabla f(\mathbf{x}^k)$ in the point $\mathbf{x}^k$. The negative gradient $-\nabla f(\mathbf{x}^k)$ defines the direction in which the function $f(\mathbf{x})$ decreases most locally. To show that $-\nabla f(\mathbf{x}^k)$ is a descent direction of SD we need only to show that $\nabla f(\mathbf{x}^k)^T\mathbf{d}^k < 0$ as it is defined in definition (2.3). Now let $d^k = -\nabla f(\mathbf{x}^k)$.

$$\begin{aligned}\nabla f(\mathbf{x}^k)^T d^k &= -(\nabla f(\mathbf{x}^k))^T \nabla f(\mathbf{x}^k) \\ &= -\|\nabla f(\mathbf{x}^k)\|^2 < 0.\end{aligned}$$

This shows that $\nabla f(\mathbf{x}^k)^T d^k < 0$ and we can conclude that $d^k = -\nabla f(\mathbf{x}^k)$ is a descent direction of SD.

2.3 Newton's method (NM)

In this method, we use a second order Taylor series approximation to determine a search direction. A second order Taylor series approximation of the function $f(\mathbf{x})$ around the point $\mathbf{x}^k$ can be written as

$$f_2(\mathbf{x}) = f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^k)^T\nabla^2 f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) \tag{2.7}$$

We should also write $\nabla^2 f(\mathbf{x}^k) = H(\mathbf{x}^k)$ to denote the Hessian matrix evaluated at the points $\mathbf{x}^k$. The main idea is to determine the optimal solution to the approximate function and use this point to determine the search direction. The gradient for the approximate function is

$$\nabla f_2(\mathbf{x}) = \nabla f(\mathbf{x}^k) + H(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) \tag{2.8}$$

We know from the necessary condition that the optimal solution is found in a stationary point. We can therefore set $\nabla f_2(\mathbf{x}) = 0$ and write the search direction as

$$\mathbf{d}^k = (\mathbf{x} - \mathbf{x}^k) = -H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k) \tag{2.9}$$

This search direction is called Newton's search direction. As we did here above for SD, we can also show that $d^k = -H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k)$ is a descent direction of NM i.e. $(\nabla f(x^k))^T d^k < 0$. That is:

$$\begin{aligned}\nabla f(\mathbf{x}^k)^T d^k &= \nabla f(\mathbf{x}^k)^T[-H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k)] \\ &= -\nabla f(\mathbf{x}^k)^T H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k)\end{aligned}$$

Recall that if $H(\mathbf{x}^k)$ is positive definite then we have a minimum at $\mathbf{x}^k$ and $H^{-1}(\mathbf{x}^k)$ is positive definite as well. This implies that

$$\nabla f(\mathbf{x}^k)^T H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k) > 0$$

which implies that

$$-\nabla f(\mathbf{x}^k)^T H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k) < 0.$$

So we can conclude that $\nabla f(\mathbf{x}^k)^T d^k < 0$ which shows that $d^k = -H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k)$ is a descent direction for NM.

Here after we give the description of both of these methods.

2.3.1 Steepest descent and Newton algorithms

1. Choose an arbitrary starting point $\mathbf{x}^0$, set $k = 0$.
2. Determine $\nabla f(\mathbf{x}^k)$ and $H(\mathbf{x}^k)$ and compute the search direction.

   $$\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$$

   for the steepest descent method and

   $$\mathbf{d}^k = -H^{-1}(\mathbf{x}^k)\nabla f(\mathbf{x}^k)$$

   for the Newton's method.
3. Check the convergence criterion. The point $\mathbf{x}^k$ is an optimal solution if

   $$\|\nabla f(\mathbf{x}^k)\| < \varepsilon.$$

4. Determine the steplength t^k. Use the line search for SD and $t^k = 1$ for NM.
5. Compute a new point as $\mathbf{x}^{k+1} = \mathbf{x}^k + t^k\mathbf{d}^k$.
6. Update $k := k + 1$ and go to step 1.

2.4 Numerical tests for the NM and SD for finding optima

In this section we will perform a numerical test using codes implemented in MATLAB for both the Newton's Method and the Steepest Descent method. We present different results obtained when performing numerical tests on two different functions.

Example 2.1 *Consider the convex objective function* $f(\boldsymbol{x}) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$ *with a given starting point. This function has* $\boldsymbol{x} = (2, 1)^T$ *as local optima. In the following table we can observe how fast the methods converges. The starting point is chosen to be* $\boldsymbol{x}^0 = (0.5, 0.5)^T$ *and the tolerance* ϵ *to be* 10^{-10}.

Steepest descent			Newton's method		
Iteration	$\mathbf{x}^k$	$\|\nabla f(\mathbf{x}^k)\|$	*Iteration*	$\mathbf{x}^k$	$\|\nabla f(\mathbf{x}^k)\|$
1	$[1.8050; 0.3200]$	5.1467	1	$[1.000; 0.5000]$	14.6353
8	$[1.6694; 0.8266]$	0.1431	4	$[1.7037; 0.8519]$	0.3512
16	$[1.7310; 0.8613]$	0.0743	8	$[1.9415; 0.9707]$	0.0027
24	$[1.7674; 0.8810]$	0.473	12	$[1.9884; 0.9942]$	2.08×10^{-5}
50	$[1.8274; 0.9126]$	0.0189	16	$[1.9977; 0.9989]$	1.60×10^{-7}
100	$[1.9118; 0.9559]$	0.0073	20	$[1.9995; 0.9998]$	1.23×10^{-9}
1000	$[1.9585; 0.9792]$	2.5565×10^{-4}	22	$[1.9998; 0.9999]$	3.22×10^{-11}
10000	$[1.9951; 0.9975]$	6.5565×10^{-5}	23	$[1.9999; 0.9999]$	3.22×10^{-11}

Table 2.1: *Convergence of iterative methods namely SD and NM applied to the same convex objective function.*

From the Table 2.1 *we can see the convergence of the methods (SD and NM). The iteration started with a chosen value* $\boldsymbol{x}^0 = (0.5, 0.5)^T$. *We stop the iteration if some of the following criteria is satisfied.*

- *The number of iteration exceeds an upper bound.*
- $\|\nabla f\|$ *is below* ϵ.

For the NM we can see that it took 23 *iterations to find a solution with* $\|\nabla f\| = 3.22 \times 10^{-11} < \epsilon = 10^{-10}$. *For the SD it reaches* 10000 *iterations approaching a solution but we can see that we still need iterations to find an approximate solution since the condition* $\|\nabla f\| < 10^{-11}$ *is not yet fulfilled. Obviously, the number of iterations to find an approximate solution is much larger for the SD than for the NM.*

Example 2.2 *Our next example is the function* $f(x) = (x_1^2 - x_2)^2 + (x_1 - 1)^2$. *We will attempt to find an approximate solution to the minimum of this function using the starting point* $\boldsymbol{x}^{(0)} = (0, 0)^T$. *The gradient of our function is given by*

$$\nabla f = \begin{pmatrix} 4x_1(x_1^2 - x_2) + 2(x_1^2 - 1) \\ -2(x_1^2 - x_2) \end{pmatrix}$$

and its Hessian is found to be

$$\nabla^2 f = \begin{pmatrix} 12x_1^2 - 4x_1x_2 + 2 & -4x_1 \\ -4x_1 & 2 \end{pmatrix}$$

Now we use this in SD and NM, setting the tolerance to be 10^{-10} *and taking account of the conditions in Example 2.1, for stopping criteria we found that the SD took* 414 *iterations to find the approximate solution* $[1.0000; 1.0000]$ *with* $\|\nabla f\| = 7.6628 \times 10^{-11} < \epsilon = 10^{-10}$ *while for the NM it took only* 3 *iterations to find the same approximate solution. This shows again how fast the NM is converging than the SD.*

Chapter 3

Finding local optima for costly objective functions using RBF.

In this section we will discuss clearly how to use the iterative methods SD and NM on the RBF surrogate model and test how fast the selected method is converging. This will be done using codes implemented in MATLAB and test the code on an objective function with a given starting point.

3.1 The surrogate modeling

Recall that we are dealing with the problem of finding a local optima for a surrogate model based on Radial Basis Function (RBF), an estimation of a costly objective function. In our introduction, it is said that for costly objective function, one way of finding a local optima is to sample the variable space and build an interpolation surface $s(x)$ using RBF. Further information about surrogate modeling can be found in [1]. In the following figures we show the costly objective function and its RBF surrogate model.

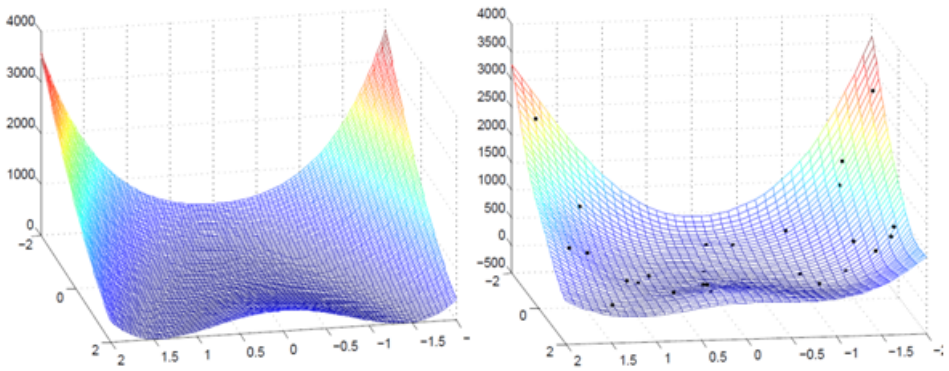

Figure 3.1: *The costly objective function $f(\mathbf{x})$ on the left and the surrogate model based on the number of sampled points $N = 30$ to the right.*

From the above figures we can remark that as we increase the number of sampled points, the surrogate model becomes increasingly more a correct description of our function $f(\mathbf{x})$.

3.2 Finding local optima using RBF

After building an interpolation surface, it is possible to perform optimization on the obtained surface $s(\mathbf{x})$ since it has an analytical representation and an analytical expression of its derivatives can be obtained. In the previous sections we derived both the first and the second derivative of this interpolant of $s(\mathbf{x})$. From the two algorithms we discussed and test the convergence here above, we realized that the NM is converging faster that the SD. Since the NM uses both the gradient and the Hessian to find a local optima, we choose it to try to find the approximate local optima. Here we introduce a **distance measure** between the true local optima (**TL**) and the one found using the interpolation surface say the interpolated local optima (**IL**) and consider the fact that the standard vector norm is a good choice, i.e.

$$\|x^*_{TL} - x^*_{IL}\|. \tag{3.1}$$

To use the above distance measure for concluding results we proceed as follows:

1. We choose randomly the number of sampled points in the variable space $[a, b]\times[a, b]$ where a and b denote respectively the lower and the upper bounds in which we are sampling points.
2. We perform multiple runs for each number of sampled points N.
3. We calculate the mean of these runs for each N which is the distance measure defined in 3.1.

Example 3.1 *Consider again the function* $f(\boldsymbol{x}) = (x_1-2)^4+(x_1-2x_2)^2$ *and the same starting point* $\boldsymbol{x}^0 = [0.5, 0.5]^T$ *as in Example 2.1. Recall that, for different randomly sampled points we can see how the surrogate model is becoming an approximation of our true function. For this example we choose randomly sampled points from* $N = 20$ *up to* $N = 70$ *in the space* $[-2, 4] \times [-2, 4]$. *Then we perform* 300 *runs using each* N *and finally compute the mean for successful runs to get the distance measure. The results have been summarized in the following table.*

Sampled points N	**Distance measure**
20	14.960781
30	0.602976
40	0.527857
50	0.474989
60	0.434048
70	0.410464

Table 3.1: *Distance measure between true and interpolated local optima using the starting point* $\mathbf{x}^0 = [0.5, 0.5]^T$.

From Table 3.1, we remark that as we increase the number of sampled points, the distance measure decreases. This means that the ***IL*** *is becoming closer to the* ***TL****. Here we have that up to* $N = 70$ *the approximate local optima is* $x^* = [1.5589; 0.7530]$ *and its function value* $f^* = -1.7191e + 003$. *Now one is interested in increasing the number of sampled points and change the starting point to see if still the distance measure decreases and probably get very and very small which implies the very closedness of* ***IL*** *to* ***TL****. For this purpose, we choose randomly sampled points form* $N = 20$ *up to* $N = 100$ *in the same space* $[-2, 4] \times [-2, 4]$ *and change the starting point to* $\boldsymbol{x}^0 = [0; 0]^T$. *Performing again* 300 *runs using each* N *and there after compute the mean for successful runs we get the following results.*

Sampled points N	**Distance measure**
20	23.244053
30	0.795023
40	0.582721
50	0.527803
60	0.435320
70	0.414193
80	0.400497
90	0.390038
100	0.386417

Table 3.2: *Distance measure between true and interpolated local optima with starting point* $\mathbf{x}^0 = [0; 0]^T$.

*We can remark that still the distance measure is decreasing and we can see that changing the starting point does not affect the method since still the expected result (****the distance measure to decrease****) is achieved. In the following figure we show the true local optima (****TL****) marked on the true function and (****IL****) marked on the surrogate modeling. The approximate local optima is* $x^* = [1.6877; 0.8805]$.

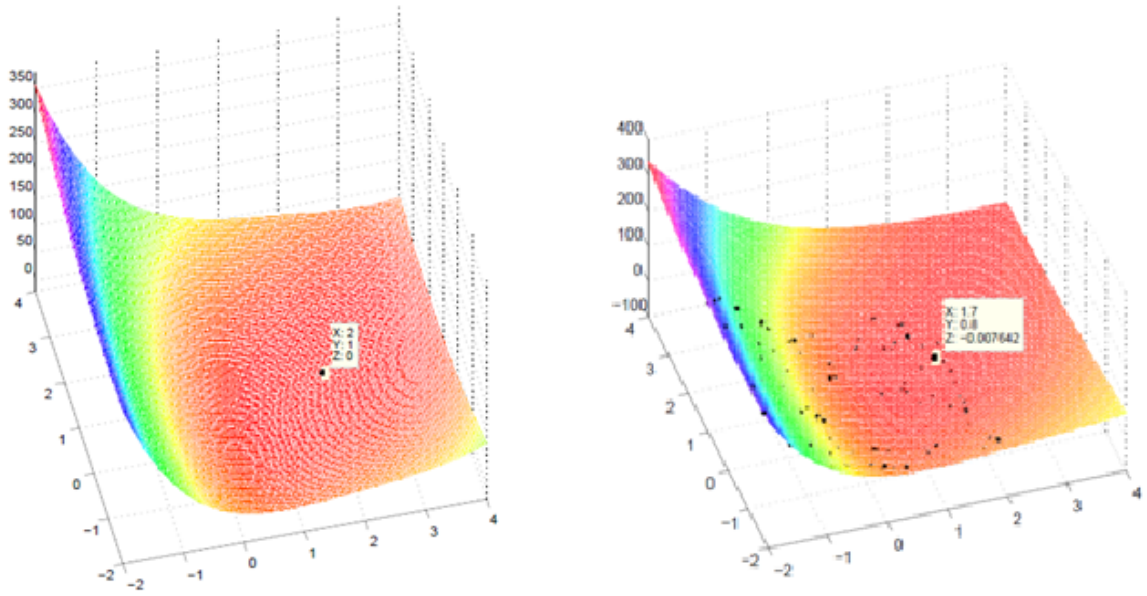

Figure 3.2: *The figures representing the true local optima (TL) marked on the true costly function on the left and the interpolated local optima (IL) marked on the interpolation surface with* $N = 100$ *to the right.*

Example 3.2 *our next test is the test on the Rosenbrock function. We will proceed as in Example3.1. We use the surrogate model of this function as in example3.1 with the starting point* $\boldsymbol{x}^0 = [0; 0]^T$ *and compute the distance measure. We summarize the results as above in a table.*

Sampled points N	**Distance measure**
20	4.211011
30	1.959497
40	29.404015
50	1.715829
60	1.440039
70	1.984533

Table 3.3: Distance measure between true (**TL**) and interpolated local optima (**IL**).

As we remark, the results in this table are somehow confusing but there is an explanation. This is due to the convexity of our function. This means that if our function is not convex enough we will have different optima but one of them is the interpolated local optima. This is not strange since in the introduction it is said that if the objective function is non-convex and non-linear, then the number of local optima to a given optimization problem is large. If we look clearly in our table we can realize that for $N = 20$ *and* $N = 30$ *the distance measure decreases and then*

for $N = 40$ it increases again. This means that already one optima has been found. But for instance using NM this remark can not be observed since only a local optima is found. Observe that again from $N = 40$ up to $N = 60$ the distance decreases again which implies that there is again an other optima. Recall that in optimization, if the function is not convex, then to find the local optima depends on the starting point. So to fix this problem we change the starting point to $[0.5, 0.5]^T$ and repeat the same task. The following are obtained results:

Sampled points N	**Distance measure**
20	24.828213
30	7.005160
40	3.438281
50	3.227534
60	3.166003
70	1.881733

Table 3.4: *Distance measure between true and interpolated local optima with starting point* $\mathbf{x}^0 = [0.5; 0.5]$.

Now we can remark the distance measure decreases as N increases which implies that **IL** is getting closer to **TL**. We can remark also that the interpolated local optima depends on the starting point for a non-convex function. In the following figures we show **TL** and **IL** on their respective surfaces.

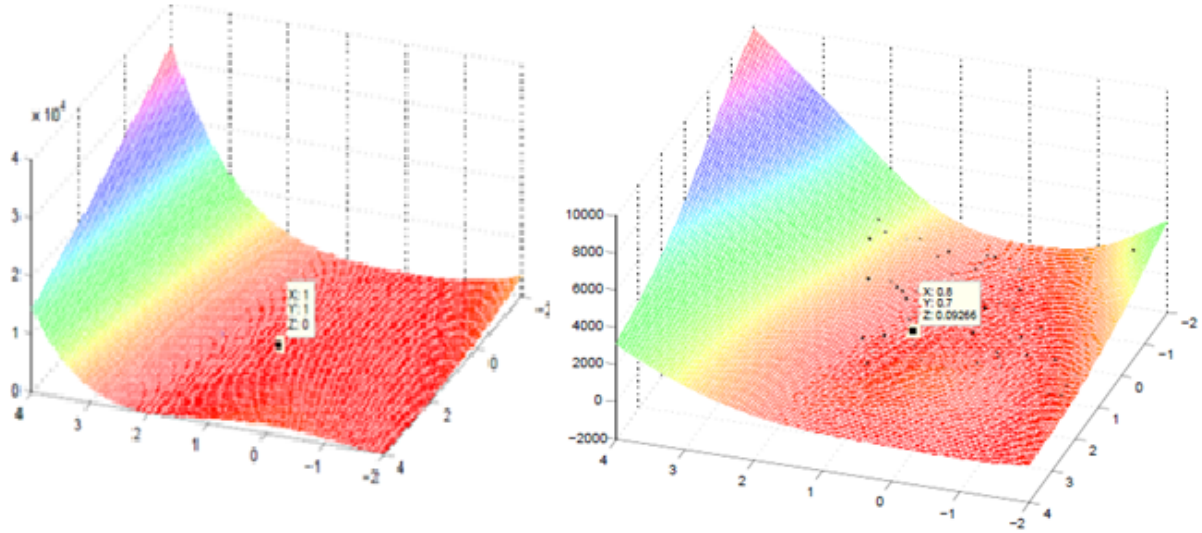

Figure 3.3: *The figures representing the true local optima (TL) marked on the true costly function on the left and the interpolated local optima (IL) marked on the interpolation surface with $N = 70$ to the right.*

3.3 Conclusions

Surrogate models are intended to be used when function evaluations take from several minutes to several hours or more. The above numerical experiments shows that for a black-box objective function, the surrogate model containing a chosen radial basis function can be used to find the interpolated local optima which is close related to the true local optima as the number of sampled points increases. The only challenge that arises is when the objective function is not convex. The method does not fail but shows that there exist a large number of local optima which is also important since for example NM did not show this but only finds a local optima. In the case of a non-convex function we choose to change the starting point and we found that our expectation about the distance measure to decrease is remaining true.

Bibliography

[1] K. Holmström, N-H. Quttineh, M. M. Edvall, An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization, *Optimization and Engineering* (2008).

[2] H.-M. Gutmann: A radial basis function method for global optimization. *Technical Report* DAMTP 1999/NA22, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England (1999).

[3] D.R. Jones. Global optimization with response surfaces. Presented at the fifth SIAM conference on optimization, Victoria Canada 1996.

[4] Juliane Müller. User Guide for Modularized Surrogate Model Toolbox. *Tampere University of technology* october 2, 2012.

[5] www.libgen.info

Appendix A

MATLAB codes

The following MATLAB codes have been used to get results of this thesis.

1. The code used to test SD

```
function [xStar,fStar]=steepestDescent(f,gradf,x0,Tol,MaxIter,TMax)
x=x0;
for k = 1:MaxIter

   g  = gradf(x);

   if norm(g) < Tol
      break
   else

      d =-g/norm(g);
   end
t=fminbnd(@(t) f(x+t*d),0,TMax);

   x1  = x+t*d;

   x = x1;
end
xStar = x1;
fStar = f(x);

end
```

2. The code used to test NM

```
function [xStar,fStar]=newtonMethod(f,gradf,hesf,x0,Tol,MaxIter)
%initialization
%we need as inputs:
%the starting point
%the function
%its gradient
%its hessian
%the maximum number of iterations
x=x0;
%Starting iterations
for i=1:MaxIter

    d =-hesf(x)\gradf(x); %iterate
    if norm(gradf(x))<Tol
        break
    end
    % use the step size to be 1.
     x1=x+d;
     x=x1;      %update x
end
xStar=x;
fStar=f(x);
end
```

3. The code used to generate points

```
  function [X,p] = generatePoints(N)
X = -2+ 4*rand(2,N);
p=zeros(N,1);
%Insert your function here as an inline function
for i=1:N
    p(i)=f(X(1,i),X(2,i));
end
end
```

The following code have been used to find the local optima using RBF surrogate model. The first code is tested using the next one.

1. the first code

```
  function [xStar,fStar] = NMforRBF(X,p,x0,Tol,MaxIter)
%-------------------------------------------------------

% Input arguments:
%
% X         sampled points, used to build RBF surface
% p         function values for sampled points
%
% x0        starting point
% Tol       tolerance for stopping criteria
% MaxIter   maximum number of iterations
%
%
% Output arguments:
%
% xStar     local optimum found
% fStar     function value at xStar
%-------------------------------------------------------


% Calculate interpolation parameters
[lamda,c] = solveSystem(X,p);


% Initialize starting point
x=x0;

% Start iterating
for i=1:MaxIter

   % Calculate the gradient
   g  = gradf(x,X,lamda,c);

   % Check stopping criteria
   if norm(g) < Tol
      break
   end

   % Calculate the Hessian
```

```
   H  = hesf(x,X,lamda);

   % Calculate search direction
   d  = -H\g;

   % Check if search direction is ascent
   if d'*g > 0

      % If so, set xStar and fStar to "infinity" and break
      xStar = inf(2,1);
      fStar = inf;
      break
      %return
   end

   % Newtons Method, use step size t=1.
   % Take the step, find new x
   x  = x + d;
end

% Return best values found (local optimum)
xStar = x;
fStar = EvaluateRBF(x,X,lamda,c);

end




%-------------------------------------------------------------
% This is your old function ThegradientRBF( x,X,lamda,c )
% Now it is used only inside the NMforRBF function.
%-------------------------------------------------------------

function Grads_n = gradf(x,X,lamda,c)
% Calculates the gradient of the RBF surface at xbar

% Dimensions and number of sampled points
[d,N] = size(X);

% Interpolation parameter b
b  = c(1:d);
```

```
Grads_n = zeros(d,1);
for i=1:N
   Grads_n = Grads_n - 3*lamda(i)*((norm(X(:,i)-x))*(X(:,i)-x));
end

Grads_n  = Grads_n+b;
end



%-------------------------------------------------------------
% This is your old function HessianRBF(x,X,lamda)
% Now it is used only inside the NMforRBF function.
%-------------------------------------------------------------

function Hessians_n = hesf(x,X,lamda)
% Calculates the Hessian of the RBF surface at xbar

% Dimensions and number of sampled points
[d,N] = size(X);

Hess  = zeros(d);
for i=1:N
   if norm(X(:,i)-x)>= 1e-10
      Hess  = Hess + 3*lamda(i)*((norm(X(:,i)-x))*eye(d) + ...
              ((X(:,i)-x)*(X(:,i)-x)')/norm(X(:,i)-x));
   end
end

Hessians_n  = Hess;
end
%-------------------------------------------------------------
% This is the old function EvaluteRBF(x,X,lamda)
% Now it is used only inside the NMforRBF function.
%-------------------------------------------------------------
function Sn=EvaluateRBF(x,X,lamda,c)

[d,N]=size(X);
b=c(1:d);
a=c(d+1);
for i=1:N
    s_n=lamda(i)*norm(X(:,i)-x)^3;
end
```

```
Sn=sum(s_n)+b'*x+a;
end
```

2. The code used to test the above code

```
x0=[0;0];
Tol=0.0001;
MaxIter=30;
trueLocalOptima = [1,1]';

M = 300;
N = 20:10:70;

distanceMatrix = zeros(length(N),M);
meanVector     = zeros(length(N),1);

fprintf('Now solving %3i runs using:\n N := ',M);
for n = 1:length(N)
   fprintf('%3i,', N(n));
   for k = 1:M

      [X,p]          = generatePoints(N(n));
      [xStar,fStar]  = NMforRBF(X,p,x0,Tol,MaxIter);

      distanceMatrix(n,k)  = norm(xStar-trueLocalOptima);

   end

end
% Calculate the mean for successful runs
fprintf('\n\nMean distance for:\n');
for k = 1:length(N)
   infIdx         = distanceMatrix(k,:) > 10000;
   meanVector(k)  = mean(distanceMatrix(k,~infIdx));

   % Number of runs that did not find a minima
   %sum(infIdx);

   % Display result
fprintf(' N =%3i:  %8.6f (%4.2f%% fails)\n',N(k), meanVector(k),sum(infIdx)/M);
end
```

Inside the first code here above we can see that there is an other function $[lamda, c] = solveSystem(X, p);$, this is given by the following code.

1. Generating points

```
function [X,p] = generatePoints(N)
X = -2+ 4*rand(2,N);
p=zeros(N,1);
%Insert your function here as an inline function
for i=1:N
    p(i)=f(X(1,i),X(2,i));
end
end
```

2. Solving the system that is describe in the thesis as (1.3)

```
function [lamda,c]=solveSystem(X,p)
[d,N]=size(X);
PHI=calculatePHI(X);
P=[X' ones(N,1)];
A=[PHI,P;P',zeros(d+1,d+1)];
b=[p;zeros(d+1,1)];
s=A\b;

lamda=s(1:N);
c=s(N+1:end);
end
```