TRUST – REGION METHOD FORDERIVATIVE FREE CASE

ADDIS ABABA UNIVERSITY
COLLEGE OF COMPUTATIONAL AND NATURAL SCIENCES
DEPARTMENT OF MATHEMATICS

In partial fulfillment of the requirement of the degree of Master of Science in mathematics

By: TewodrosNegash

Stream: Optimization

Advisor: Semu Mitiku (Ph.D)

November 2015
Addis Ababa

Trust Region Method for Derivative Free Case

By: Tewodros Negash

A Project Submitted to Addis Ababa University
College of Computational and Natural Science
Department of Mathematics

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Mathematics

November 2015
Addis Ababa

Addis Ababa University

Department of Mathematics

The undersigned hereby certify that they have read and recommend to graduate studies for acceptance of a project entitled: **trust-region method for derivative free case** by Tewodros Negash in partial fulfillment of the requirements for the degree of Master of Science.

November, 2015

Addis Ababa, Ethiopia

Advisor: _____

Semu Mitiku (Ph.D)

Chairman: Tesfa B.

Examining committee 1. Berhanu G. (Ph.D) _____

2. Taddese A. (Ph.D) _____

Acknowledgement

First and for most I thank the almighty God for helping me to realize this project from beginning to the end .I also would like to acknowledge all the people who were directly or indirectly involved in supporting and encouraging me to finish my project.

I am strongly indebted to my project advisor, Dr. Semu Mitiku, who welcomed me open heartedly and for his unreserved advice and skillful contribution to this project and his meticulous comments I received throughout this project work. I have special respect and appreciation to him, for his free and family like approach and advice.

My heart-felt appreciation also goes to my families for their moral and financial support and follows up. Finally I am indebted to so many other people whose names are not listed here, but who have contributed unreserved support for the success of this project for which I am very thankful.

Abstract

In this project a derivative-free (DF) surrogate-based trust region optimization approach is proposed. In the proposed approach, quadratic surrogate models are constructed and successively updated. The generated surrogate model is then optimized instead of the underlined objective function over trust regions. Truncated conjugate gradients are employed to find the optimal point within each trust region. The approach constructs the initial quadratic surrogate model using few data points of order O(n), where n is the number of design variables. The proposed approach adopts weighted least squares fitting for updating the surrogate model instead of interpolation which is commonly used in DF optimization. This makes the approach more suitable for stochastic optimization and for functions subject to numerical error. The weights are assigned to give more emphasis to points close to the current center point.

Keywords

- *Derivative-free optimization;*
- *Quadratic surrogate model;*
- *Trust region;*

Contents

CHAPTER ONE

BASIC CONCEPTS AND DEFINITION

1.1 PRELIMINARY

Mathematical Optimization is often also called *Nonlinear Programming, Mathematical Programming* or *Numerical Optimization.* In more general terms Mathematical Optimization may be described as the science of determining the *best* solutions to mathematically defined problems, which may arise from models of physical reality or of manufacturing and management systems.

Optimization problems are common in many disciplines and various domains. In optimization problems, we have to find solutions which are optimal or near-optimal with respect to some goals. Usually, we are not able to solve problems in one step, but we follow some process which guides us through problem solving. Often, the solution process is separated into different steps which are executed one after the other. Commonly used steps are recognizing and defining problems, constructing and solving models, and evaluating and implementing solutions.

Optimization problems can be categorized in to two groups depending on whether or not constraints exist in the problem, those are constrained and unconstrained. In which the optimum value is sought of an objective function of many variable without any constraints is called unconstrained optimization.

Unconstrained optimization involves finding the vector of variables x such that an objective function $f(x)$ achieves its minimum or maximum value. Thus, unconstrained optimization problems are usually categorized by the characteristics of the function f. This project focuses on techniques for solving unconstrained optimization problems using derivative free case of Trust-region method.

As minimizing $f(x)$ is mathematically equivalent to maximizing$-f(x)$, all problems are formulated as minimization problem. In the rest part of this chapter, we will define some preliminary concepts and recall their properties which are useful in our later discussion.

1.2 Convexity and Minimization

Definition: A set $S \subseteq R^n$ is called a convex set if for all $x, y \in S$ and for all $\lambda \in [0,1]$ it holds $\lambda x + (1-\lambda)y \in S$.

Definition: A function $f: S \to R$,where S is a nonempty convex set, is a convex function if $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ for all $\lambda \in [0,1]$ and $x, y \in S$.

Definition: A function $f(x)$ is called a strictly convex function if the inequality above is strict for all $x \neq y$ and $\lambda \in (0,1)$.

Definition: A function $f: S \to R$,where S is a nonempty convex set, is a concave function if $f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda)f(y)$ for all $\lambda \in [0,1]$ and $x, y \in S$.

Definition: A function $f(x)$ is called a strictly concave function if the inequality above is strict for all $x \neq y$ and $\lambda \in (0,1)$.

1.3 Eigenvalues and Eigenvectors

Given an n by n matrix A the solution $(u, \lambda)(u \neq 0)$ to the nonlinear equation $Au = \lambda u$ are known as eigenpairs. The scalar λ is eigenvalue and the vector u is an eigenvector.

There are many related characterization of eigenvalues, of which the best known is that the eigenvalues necessarily satisfy the equation

$$\det(A - \lambda I) = 0$$

If all the eigenvalues of a real square matrix $A$are nonnegative real numbers then it is called positive semi definite matrix and if all the eigenvalues are negative real numbers, then A is called negative-definite matrix. If such a matrix has both positive and negative eigenvalues we shall say that it is indefinite. Moreover, we have also the following equivalent definition.

Definition: An nxn symmetric matrix G is called

- Positive definite if $s^T Gs > 0$ for all $s \in R^n, s \neq 0$.
- Positive semi-definite if $s^T Gs \geq 0$ for all $s \in R^n$.
- Negative definite if $s^T Gs < 0$ for all $s \in R^n, s \neq 0$.

- Negative semi-definite if $s^T Gs \leq 0$ for all $s \in R^n$.
- indefinite if there exists $s_1, s_2 \in R^n$ for which ${s_1}^T G s_1 > 0$ and ${s_2}^T G s_2 < 0$.

1.4 Optimality Condition

We shall be concerned with the optimization problem of minimizing an objective function $f(x)$ of n real variables$x$, where x is constrained to lie within a closed region C of feasible points. This feasible region may be the whole of R^n , in this case the problems is effectively unconstrained or it may be a subset ofR^n, in this case the problem is constrained.

A feasible point x^* is a local minimizer of f if there is an open neighborhood Ω of x^* such that $f(x^*) \leq f(x)$ for all$x \in C \cap \Omega$. The minimizer is strong or strict if there is an open neighborhood Ω of x^* such that $f(x^*) < f(x)$ for all $x \neq x^* \in C \cap \Omega$ and it is global minimizer if $f(x^*) \leq f(x)$ for all$x \in C$.

The value of f corresponding to a minimizer is a minimum.

Necessary Condition

Suppose the function f has a local minimum in$x_o \in \dot{C}$, that is, in an interior point of$C$. Then

1. If f is differentiable in x_o then $\nabla f(x_o) = 0$ holds.
2. If f is twice continuously differentiable in a neighborhoodofx_o, then the Hessian $G_f(x_o) = \nabla^2 f(x_o) = \left(\partial^2 f(x_o) \big/ \partial x_i \partial x_j\right)_{n \times n}$ is positive semi definite.

Theorem 1: Let f be twice continuously differentiable and let x^* be a local minimizer of f. Then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semi definite.

Proof: Let $u \in R^n$ be given. Taylor's theorem states that for all real t sufficiently small

$$f(x^* + tu) = f(x^*) + t\nabla f(x^*)^{\mathrm{T}} u + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t^2)$$

Since x^* is a local minimize we must have for t sufficiently small $0 \leq f(x^* + tu) - f(x^*)$ and hence $\nabla f(x^*)^{\mathrm{T}} u + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t) \geq 0$ for all t sufficiently small and all $u \in R^n$. So, if we set $t = 0$ and $u = -\nabla f(x^*)$ we obtain $\|\nabla f(x^*)\|^2 = 0$

Setting $\nabla f(x^*) = 0$ dividing by t and settingt = 0, we obtain $\frac{1}{2} u^T \nabla^2 f(x^*) u \geq 0$ for all$u \in R^n$.

Theorem 2: Let f be twice continuously differentiable in neighborhood ofx^*. Assume that $\nabla f(x^*) = 0$ and that $\nabla^2 f(x^*)$ is positive semi definite. Then x^* is a local minimizer of$f$.

Proof: Let $0 \neq u \in R^n$. for sufficiently small t we have

$$f(x^* + tu) = f(x^*) + t\nabla f(x^*)^{\mathrm{T}} u + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t^2)$$

$$= f(x^*) + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t^2)$$

Hence, if $\lambda > 0$ is the smallest eigenvalue of $\nabla^2 f(x^*)$, we have

$$f(x^* + tu) = f(x^*) + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t^2)$$

$$\geq f(x^*) + \frac{\lambda}{2} \|\mathrm{tu}\|^2 + o(t^2) \geq 0$$

$f(x^* + tu) - f(x^*) \geq \frac{\lambda}{2} \|\mathrm{tu}\|^2 + o(t^2) \geq 0$ For t sufficiently small.

$$f(x^* + tu) - f(x^*) \geq 0$$

Hence x^* is a local minimizer of f.

Definition: Iteration is a process by which an estimate x^k of the solution to the problem is replaced by a better estimate x^{k+1}.

1.5 Vector and Functional Norm

We use norms to measure the size of vectors, matrices and functions. A norm is a function I.I: S $\longrightarrow$R which required satisfying three basic properties

1. $\|x\| \geq 0$ and $\|x\| = 0 \Leftrightarrow x = 0$.
2. $\|\alpha x\| \leq |\alpha|\|x\|$ for all $\alpha \in R$.
3. $\|x + y\| \leq \|x\| + \|y\|$ for all x and y in the relevant space S.

The third property is known as the triangular inequality.

When $s \in R$ the most common norms are the l_p vector norms $(p \geq 1)$ defined by

$$\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

The most common of those are the vector l_1, l_2 and l_∞ norms, overR^n.

$\|x\|_1 = \sum_{i=1}^{n}|x_i|$, $\|x\|_2 = \sqrt{\langle x, x\rangle} and \|x\|_\infty = \max_{1\leq i\leq n}|x_i|$, where$\langle x, y\rangle$ denote the inner product

$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i$ on R^n.

The l_2 norm is often known as the Euclidean norm.

Suppose I.I is a norm R^n and we consider a subset S of R^n. We define an open ball of radius $x \in S$ to be the set

$$\Omega_\varepsilon(x) = \{ y : \|y - x\| < \varepsilon\}$$

The set S is said to be open in R^n if for every x in S, there is a scalar $\varepsilon(x) > 0$ such that $\Omega_\varepsilon(x) \subseteq S$.

1.6 Taylor Serious and Quadratic Models

Let $f(x)$ be a function from R^n to$R$. We say that f is in $\mathbb{C}^{(k)}$ if it has continuous k^{th} order partial derivatives. If f is differentiable $(f \in \mathbb{C}^1)$, we define the gradient of $f(x)$ to be the vector valued function $\nabla_x f(x)$ whose i^{th} component is $\partial f(x)/\partial x_i$. If in addition,

f is twice continuously differentiable($f \in \mathbb{C}^2$), we define the Hessian matrix of f to be the n by n matrix-valued function $\nabla_{xx} f(x)$ whose $(i,j)^{th}$ entry is $\partial^2 f(x) / \partial x_i \partial x_j$. If the derivatives are continuous the Hessian is necessarily symmetric.

Let f has derivatives of all orders at a, and then we call

$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$,to be the Taylor series of f about the number a. The n^{th} Taylor polynomial p_n of f about a is defined by

$$p_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f''(a)}{n!}(x-a)^n$$

The first-order Taylor approximation is: $f(x+\alpha) \approx f(x) + \langle \nabla_x f(x), \alpha \rangle$ and

The Second-order Taylor approximation is given by:

$$f(x+\alpha) \approx f(x) + \langle \nabla_x f(x), \alpha \rangle + \frac{1}{2}\langle \alpha, \nabla_{xx} f(x)\alpha \rangle.$$

The quadratic model is obtained from a truncated Taylor series expansion of $f(x)$ aboutx^k, which can be written as

$f(x^{(k)} + \alpha) \approx M^{(k)}(\alpha) = f^{(k)} + g^{(k)^T}\alpha + \frac{1}{2}\delta^T G^{(k)}\alpha$,Where$\alpha = x - x^{(k)}$ and $M^{(k)}(\alpha)$ is the resulting quadratic approximation at the point x^k.

Some tentative reasons to select quadratic model are

1) A quadratic function is one of the simplest smooth functions with a well determined minimum.
2) A general function expanded about a local minimize x^* is approximated well by quadratic function.
3) Methods based on quadratic models can be made invariant under a linear transformation of variables and quadratic models are easy to manipulate.

1.7 What Is a Trust Region?

Let us consider the problem,

$$\min f(x)$$
$$x \in R^n$$

Where f is a smooth nonlinear objective function from R^n to R. Assume we have a current guess of the solution of the optimization problem say x_0. We now will try to find the optimal point x^* where $f(x^*)$ is lowest by applying numerical method, an *algorithm*. Based on this information, we may try to guess the shape of the objective function in a neighborhood of x_0. In other words, we build around x_0 a *model* of the objective function that is "easier" to work with and decide, admittedly with some degree of arbitrariness, on a region containing x_0 in which we believe the model represents the objective function more or less adequately. This region is called the *trust region* because this is where we trust the model to be a faithful representation of the objective function.

A trust region is normally a neighborhood $\Omega^{(k)}$ at the current iterate $x^{(k)}$ and the region is adjusted from iteration to iteration. The neighborhood $\Omega^{(k)}$ of $x^{(k)}$ is defined in which $q^{(k)}(\alpha)$ agree with $f(x^{(k)} + \alpha)$ in some sense. Then it would be appropriate to choose at the k^{th} iteration $x^{(k+1)} = x^{(k)} + \alpha^{(k)}$ where the correction $\alpha^{(k)}$ minimizes $q^{(k)}(\alpha)$ for all $x^{(k)} + \alpha$ in $\Omega^{(k)}$. The step is restricted by the region of validity of the Taylor series. It will be shown that these methods retain rapid rate of convergence of Newton's method but are also generally applicable and globally convergent.

In the trust region method first we define a region around the current iterate

$$\Omega^{(k)} = \{x: \|x - x^{(k)}\| \leq h^{(k)}\} \quad \text{Where } h^{(k)} > 0 \text{ is the radius}$$

of $\Omega^{(k)}$, which the model is trusted to be adequate to the objective function? And then choose a step to be the approximate minimizer of the quadratic model in the trust region method.

The model sub problem of the trust-region method

$$\min \quad q^{(k)}(\alpha) = f^{(k)} + g^{(k)^T}\alpha + \frac{1}{2}\delta^T G^{(k)}\alpha \qquad s.t\|\alpha\| \leq h^{(k)}$$

,where $g^{(k)} = \nabla f\left(x^{(k)}\right)$ is the gradient at the current iterate $x^{(k)}$and $G^{(k)}$ is an n by n symmetric matrix approximates the Hessian of $f(x)$ and $h^{(k)} > 0$ is trust region radius.

How to choose the radius $h^{(k)}$ at each iteration?

In general, when there is good agreement between the model $q^{(k)}(\alpha)$ and the objective function value $f\left(x^{(k)} + \alpha\right)$ one should select $h^{(k)}$ as large as possible. This can be quantified by defining the actual reduction $(Aredk)$ in$f$ and on the k^{th} step as $\Delta f^{(k)} = f^{(k)} - f\left(x^{(k)} + \alpha^{(k)}\right)$ and

The corresponding predicted reduction $(Predk)$ as

$$\Delta q^{(k)} = q^{(k)}(0) - q^{(k)}\left(\alpha^{(k)}\right) = f^{(k)} - q^{(k)}\left(\alpha^{(k)}\right)$$

Define the ratio $r^k = \frac{\text{Actual reduction}}{\text{Predicted reduction}} = \frac{Aredk}{Predk}$ which measures the agreement between the model function $q^{(k)}$ and the objective function$f$.

The ratio $r^{(k)}$ plays an important role in selecting new iterate $x^{(k+1)}$ and updating the trust-region radius$h^{(k)}$ [9]. If $r^{(k)}$ is close to unity it means there is good agreement and we can expand the trust-region for the next iteration. If $r^{(k)}$ is close to zero or negative we shrink the trust-region, otherwise we do not alter the trust-region.

Steps of a trust region method algorithm

Step1. Given a current iterate build a good local approximation model (e.g., based on a second order Taylor series approximation).

Step2. Choose a neighborhood around the current iterate where the model is "trusted" to be accurate. Minimize the model in this neighborhood.

Step3. Determine if the step is successful by evaluating the true function at the new point and comparing the true reduction in value of the objective with the reduction predicted by the model.

Step4. If the step is successful, accept the new point as the next iterate and proceed (possibly increasing the size of the trust region if the success is really significant). If the step is unsuccessful, reject the new point and reduce the size of the trust region.

Step5. Repeat until convergence.

CHAPTER TWO

DERIVATIVE FREE OPTIMIZATION

2.1 INTRODUCTION

Derivative free optimization methods are designed for solving nonlinear optimization without using the derivative of the involved functions. The derivative free optimization method which we are going to use in this project approximates the objective function explicitly without employing its derivative.

We consider formally the problem,

$$\min f(x)$$
$$x \in R^n \qquad (2.1)$$

Where f is a smooth nonlinear objective function from R^n to R and is bounded. Since the derivatives of the objective function are not available for derivative free optimization, we assume that the gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$ cannot be computed for any x. The numerous applications of derivative free optimization can be found in engineering design, geological modeling, finance, manufacturing, biomedical applications and many other fields. As the available computational power grows the simulation processes become routine and using optimization of complex systems becomes possible and desirable. Thus the number of applications of derivative free optimization grows continuously, which partially explains the continuing growth of the field itself. Another reason for the growth of the field is the recent development of relatively sophisticated algorithms and theory which address the specific needs of the derivative free problems.

The majority of the existing derivative-free techniques have the following features:

•They require a relatively large number of function evaluations, $O(n^2)$ (where n is the number of system design variables) to construct the initial quadratic model.

•The quadratic models are constructed via interpolating the objective function at a constant number of points; when a point is obtained a previous point is dropped. In

addition, these algorithms usually ignore the valuable information contained in all previously evaluated expensive function value

2.2 Derivative Free Optimization methods.

This part of the chapter only focuses on different methods of solving unconstrained minimization problem whose derivative of the objective function is not available. To use trust region framework in the derivative free case we use an alternative approximation technique, which does not use derivative estimates, which will be discussed in Chapter Three. There are many methods of solving derivative free optimization problems. Direct search method and quadratic interpolation are the most widely used methods. These methods were introduced by Powell [5] .

2.2.1 Quadratic Interpolation

The main idea of these methods is building a polynomial model, interpolating the objective function at all points at which its value is known. The model is then minimized over the trust region and a new point, is computed. The objective function evaluated at this new point thus possibly enlarging the interpolation set. This newly computed point is checked as to whether the objective function is improved and the whole process is repeated until convergence is achieved. The basic concepts for constructing a quadratic model function from a sample points are introduced. For trust-region methods building a model by polynomial interpolation have been developed by a number of authors. Many interpolation-based trust-region methods construct local polynomial interpolation-based models of the objective function and compute steps by minimizing these models inside a region using the standard trust-region methodology. The models are built so as to interpolate previously computed function values at a subset of past iterates or at specially constructed points. For the model to be well-defined, the interpolation points must be poised, meaning that the geometry of this set of points has to "span the space" sufficiently well to stay safely away from degeneracy. To make things more clear some necessary information about used notations, the general trust-region framework, interpolation, and the definition of poisedness and well-poisedness are given. In our derivative-free context, the model (2.2) will be determined by interpolating known

objective function values at a given set X_k of *interpolation points*, meaning that the *interpolation conditions*

$$M_k(x) = f(x) \text{ for all } x \in X_k \qquad (2.2)$$

must hold. The set X_k is known as the *interpolation set*. We will say that a set of points can be interpolated by a polynomial of certain degree if for any function f there exists a polynomial M (of this degree) such that (1.1) hold for all points in the set.

Definition:-A set of points X is called POISED, with respect to a given subspace of polynomials, if it can be interpolated by polynomials from this subspace.

Definition:- A set of points X is called well-poised, if it remains poised under small perturbations.

Suppose $\{\phi_i(.)\}_{i=1}^{q}$ is a basis in the space of quadratic polynomials, then any quadratic polynomial $\sum_{i=1}^{q} \alpha_i B_i(x^j)$ for some $\alpha = (\alpha_1, \ldots, \alpha_q)$. The interpolation condition can be written as a system of linear equation in α.

$$\sum_{i=1}^{q} \alpha_i B_i(x^j) = f(x^j) \; j = 1, \ldots, p. \qquad (2.3)$$

The coefficient matrix of this system is

$$B(X) = \begin{pmatrix} B_1(x^1) & \ldots. & B_q(x^1) \\ \vdots & \vdots & \vdots \\ B_1(x^p) & \ldots. & B_q(x^p) \end{pmatrix} \qquad (2.4)$$

For a given set of points and a set of function values an interpolation polynomial (from a given space of polynomials) exists and is unique if and only if $B(X)$ is square and nonsingular.

If the set X is poised, then theoretically, we can solve the linear system and find the interpolation polynomial. However, numerically the matrix $B(X)$ is often ill – conditioned even when it is nonsingular. Conditioning of $B(X)$, clearly, depends of the choice of the basis $\{B_i(x)\}$. For example, one can choose a basis such that is an $B(X)$ identity matrix. Such basis is called Lagrange fundamental polynomial basis.

Algorithm

Step 0: Initialization

Let a starting point x^s and the value $f(x^s)$ be given

Choose an initial trust region radius $\Delta_0 > 0$

Choose at least one additional point not further than Δ_0 away from x^s to create an initial well-poised interpolation set Y and initial basis on Newton fundamental polynomials.

Determine $x^0 \in Y$ which has the best objective function value;

i.e $f(x^0) = \min_{y^i \in Y} f(y^i)$.

Set k=0

Set parameters η_0, η_1 to measure progress; $0 < \eta_0 < \eta_1 < 1$.

Step 1:Build the model.

Using the interpolation set Y and the basis of NFP, build an interpolation model $Q_k(x)$.

Step 2:Minimize the model within the trust region.

Set $B_k = \{x : \|x - x^k\| \leq \Delta_k\}$. Compute the point such that

$$Q_k(x) = \min_{x \in B_k} Q_k(x)$$

Compute $f(\hat{x}^k)$ and the ratio

$$\rho_k \equiv \frac{f(x^k) - f(\hat{x}^k)}{Q_k(x^k) - Q_k(\hat{x}^k)}$$

Step 3: Update the interpolation set.

- If $\rho_k \geq \eta_0$, include $\hat{x}^k$ in Y, dropping one of the existing interpolation points if necessary.
- If $\rho_k < \eta_0$ include $\hat{x}^k$ in Y, if it improves the quality of the model
- If $\rho_k < \eta_0$ and there are less that n+1 points in the intersection of Y and $\boldsymbol{B}_k$ generate new interpolation point in $\boldsymbol{B}_k$, while preserving/ improving well-poisedness.
- Update the basis of the Newton Fundamental polynomials.

Step 4: Update the trust region radius.

- If $\rho_k \geq \eta_1$ increase the trust region radius

$\Delta_{k+1} \in [\Delta_k, \gamma_2 \Delta_k]$

- If $\rho_k < \eta_0$ and cardinality of Y $\cap \boldsymbol{B}_k$ was less than n+1 when $\hat{x}^k$ was computed reduce the trust region

$\Delta_{k+1} \in [\gamma_0 \Delta_k, \gamma_1 \Delta_k]$

- Otherwise, set $\Delta_{k+1} = \Delta_k$

Step 5: Update the Current iterate

Determine x^k with the best objective function value

$$f(\bar{x}^k) = \min_{\substack{y^j \in Y \\ y^j \neq xk}} f(y^j)$$

If improvement is sufficient (w.r.t. predicted improvement)

$$\bar{\rho}_k \equiv \frac{f(x^k) - f(\bar{x}^k)}{Q_k(x^k) - Q_k(\hat{x}^k)} \geq \eta_0,$$

Set $x^{k+1} = \bar{x}^k$. Otherwise, set $x^{k+1} = x^k$. Increment k by one and go to step 1.

End of algorithm

2.2.2 DIRECT (SIMPLEX) SEARCH METHODS

The Nelder-Mead direct search is also called simplex search algorithm. This method requires only function evaluations, but not derivatives. As such the method is useful when

- The derivative of the objective function is expensive to compute;
- Exact first derivatives of f are difficult to compute or f has discontinuities;
- The values of f are 'noisy'.

There are many practical optimization problems which exhibit some or all of the above difficult properties. In particular, if the objective function is a result of some experimental (sampled) data, this might usually be the case.

Let the objective function f be from R^n to R. A simplex S in R^n is a polyhyderal set with n+1 vertices $x_1, x_2, \ldots, x_{n+1} \in R^n$ such that $\{x_k - x_i: k \in \{1,\ldots,n+1\} \setminus \{i\}\}$ is linearly independent in R^n. A simplex S is non – degenerate if none of its three vertices lie on a line or if none of its four points lie on a hyperplane, etc.

Thus the Nelder-Mead simplex algorithms search the approximate minimum of a function by comparing the values of the function on the vertices of a simplex. Accordingly, Sk is a simplex at k^{th} step of the algorithm with ordered vertices $\{x_1^k, x_2^k \ldots, x_{n+1}^k\}$ in such a way that

$$f(x_1^k) \le f(x_2^k) \le \ldots \le f(x_{n+1}^k)$$

Hence, x_{n+1}^k is termed the 'worst' vertex, x_n^k the next 'worst' vertex, etc. for the minimization, while x_1^k is the best vertex. Thus a new simplex S^{k+1} will be determined by dropping vertices which yield larger function values and including new vertices which yield reduced function values, but all the time keeping the number of vertices n+1 (1 plus problem dimension). This is achieved through reflection expansion, contraction or shrinking of the simplex S^k. In each step it is expected that $S^k \neq S^{k+1}$ and the resulting simplices remain non-degenerate.

Algorithm

Step 0: Start with a non-degenerate simplex S0 with vertices $\{x_1^0, x_2^0, \ldots, x_{n+1}^0\}$. Choose the constraints

$$\rho>0,\ \chi>1\ (\chi>\rho),\ 0<\gamma<1, \text{and } 0<\sigma<1$$

known as reflection, expansion, contraction and shrinkage parameters ,respectively.

While(1)

Step1: Set $k \leftarrow k+1$ and label the vertices of the simplex S^k so that x_1^k, $x_2^k \ldots, x_{n+1}^k$ according to

$$f(x_1^k) \le f(x_2^k) \le \ldots \le f(x_{n+1}^k)$$

Step2: Reflect

- Compute the reflection point x_r^k

  $$x_r^k = \bar{x}^k + \rho(\bar{x}^k - x_{n+1}^k) = (1+\rho)\bar{x}^k - \rho x_{n+1}^k$$

  where $\bar{x}^k = \sum_{i=1}^{n} x_i^k$ (here the worest point x_{n+1}^k will not be used)
- Compute $f(x_r^k)$
- **If** $f(x_1^k) \le f(x_r^k) < f(x_n^k)$, **then** accept x_r^k and reject x_{n+1}^k and GOTO Step 1. **Otherwise** GOTO **Step 3**.

Step 3: Expand

(a). If $f(x_r^k) < f(x_1^k)$, **then** calculate the expansion point x_e^k :

$$x_e^k = \bar{x}^k + \chi(x_r^k - \bar{x}^k) = \bar{x}^k + \rho(\bar{x}^k - x_{n+1}^k) = (1+\rho\chi)\bar{x}^k - \rho\chi x_{n+1}^k$$

- Compute $f(x_e^k)$
  - **If** $f(x_e^k) < f(x_r^k)$, **then** accept x_e^k and reject x_{n+1}^k and GOTO Step 1.
  - **Else** accept x_r^k and reject x_{n+1}^k and GOTO Step 1.

(b). **Otherwise** GOTO Step 4.

Step 4: Contract

If $f(x_r^k) \geq f(x_n^k)$, then perform a contraction between $\bar{x}^k$ and the better of x_{n+1}^k and x_r^k

(a). **Outside Contraction:** If $f(x_n^k) \leq f(x_r^k) < f(x_{n+1}^k)$ (i.e. $\bar{x}^k$ is better than x_{n+1}^k) ,then perform outside contraction, i.e. calculate $x_c^k = \bar{x}k - \gamma(x_r^k - \bar{x}^k)$ and evaluate $f(x_c^k)$.

- If $f(x_c^k) \leq f(x_r^k)$, then accept x_c^k and reject x_{n+1}^k and GOTO Step 1.
- Otherwise GOTO Step 5. (Perform shrink)

(b). **Inside Contraction:** If $f(x_r^k) \geq f(x_{n+1}^k)$,(i.e. x_{n+1}^k is better than x_r^k), then perform an inside contraction; i.e. calculate

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

- **If** $f(x_{cc}^k) < f(x_{n+1}^k)$,**then** accept x_{cc}^k and reject x_{n+1}^k and GOTO Step 1.
- **Otherwise** GOTO Step 5.(perform Shrink)

Step 5: Shrink

Define n new points

$v_i = x_1 + \sigma(x_i^k - x_1), i = 2, \ldots, n+1$

so that the $n+1$ points

$x_1, v_2, \ldots, v_{n+1}$

form the vertices of a simplex. GOTO Step 1.

END

Unfortunately, to date, there is no concrete convergence property that has been proved of the original Nelder-Mead algorithm. The algorithm might even converge to a non-stationary point of the objective function. However, in general, it has been tested to provide rapid reduction in function values and successful implementations of the algorithm usually terminate with bounded level sets that contain possible minimum points. Recently there are several attempts to modify the Nelder-Mead algorithm to come up with convergent variants. Among these: the fortified-descent simplical search method and a multidimensional search algorithm are two of the most successful ones [3].

CHAPTER THREE

THE TRUST REGION APPROACH

3.1 TRUST REGION ALGORITHM

In this chapter we consider formally the problem,

$$\min f(x)$$
$$x \in R^n$$

Where f is a smooth non linear objective function from R^n to R and is bounded. Since the derivatives of the objective function are not available, the computationally expensive objective function is locally approximated around a current iterate x^k by a computationally cheaper quadratic surrogate model $M(x)$ which can be placed in the form:

$$M(x) = a + b^T(x - x_k) + \frac{1}{2}(x - x_k)^T B(x - x_k) \qquad (1)$$

Where $a \in R$, the vector $b \in R^n$, and the symmetric matrix $B \in R^{n \times n}$ are the unknown parameters of $M(x)$. The total number of the model parameters is $q = (n + 1)(n + 2)/2$. These parameters can be evaluated by interpolating the objective function at q points.

Initial model

Let x_0 be the initial point. Initially, assuming that B is a diagonal matrix, then the number of points required to construct the initial model is $m = 2n + 1$. The initial m points x_i, $i = 1, 2, \ldots, m$, can be chosen as follows

$$x_1 = x_0 \text{ and } \begin{cases} x_{i+1} = x_0 + \Delta_1 e_i, & i = 1,2,\ldots,n \\ x_{i+n+1} = x_0 - \Delta_1 e_i, & i = 1,2,\ldots,m-n-1 \end{cases} \qquad (2)$$

where Δ_1 is the initial trust region radius that is provided by the user, and e_i is the i^{th} coordinate vector in R^n.

The initial quadratic model $M^{(1)}(\boldsymbol{x})$ will have the parameters $a^{(1)}$, the vector $\boldsymbol{b}^{(1)}$, and the n diagonal elements of the model Hessian matrix $\boldsymbol{B}^{(1)}$. These parameters are computed by requiring that the initial model interpolates the objective function $f(x)$ at the initial m points given in (2). Therefore the initial model parameters are obtained by satisfying the matching conditions:

$$M^{(1)}(x_i) = f(x_i), \qquad i = 1, 2, \ldots, m \qquad (3)$$

Model optimization

At the k^{th} iteration, assume that $\boldsymbol{x}_k$ is the current solution point. The model $M^{(k)}(\boldsymbol{x})$ is then minimized, in place of the objective function, over the current trust region and a new point is produced by solving the trust region sub-problem:

$$\min_s M^{(k)}(s), \quad \text{subject to } \|s\| \leq \Delta_k \qquad (4)$$

Where $\boldsymbol{s} = \boldsymbol{x} - \boldsymbol{x}_k$, Δ_k is the current trust region radius, and $\|\cdot\|$ throughout is the l_2-norm. This problem is solved by the method of truncated conjugate gradient by Steihaug [8]. It is identical as the standard conjugate gradient method as long as the iterates are inside the trust region. If the conjugate gradient method terminates at a point within the trust region, this point is a global minimizer of the objective function. If the new iterate is outside the trust region, a truncated step which is on the region boundary is considered. Also, the method treats the case where the minimum is in the opposite direction of the conjugate direction which is due to the non convexity of the model [7]. One good property of this method is that the solution computed has a sufficient reduction property, which was proved by Bandler and Abdel-Malek [8].

Let $\boldsymbol{s}^*$ denotes the solution of (4), and then a new point $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}^*$ is obtained. The achieved actual reduction in the objective function is compared to that predicted reduction using the model by computing the reduction ratio which is given by:

$$r_k = \frac{actual\ reduction}{predicted\ reduction} = \frac{f(x_k) - f(x_n)}{M^{(k)}(x_k) - M^{(k)}(x_n)} \qquad (5)$$

This ratio reflects how much the surrogate model agrees with the objective function within the trust region. The trust region radius and the current iterate will be updated such that, if r_k is sufficiently high, i.e., $r_k \geq 0.7$, there is a good agreement between the model and the objective function over this step. Hence, it is beneficial to expand the trust region for the next iteration, and to use $\boldsymbol{x}_n$ as the new center of the trust region. If r_k is positive but not close to 1, i.e., $0.1 \leq r_k < 0.7$, the trust region radius is not altered. On the other hand, if r_k is smaller than a certain threshold, $r_k < 0.1$, the trust region radius is reduced. The updating formula used for updating Δ_k and $\boldsymbol{x}_k$ can be expressed as follows:

$$r_k \begin{cases} r_k < 0.1: & \Delta_{k+1} = \frac{1}{2}\Delta_k \\ 0.1 \leq r_k \leq 0.7: & \Delta_{k+1} = \Delta_k \\ r_k \geq 0.7 & \begin{cases} \|s^*\| < \Delta_k: \Delta_{k+1} = \Delta_k \\ \|s^*\| \geq \Delta_{k+1} = 1.5\Delta_k \end{cases} \end{cases} \qquad (6)$$

$$x_{k+1} = \begin{cases} x_k + s^*, & if\ r_k > 0 \\ x_k & otherwise \end{cases} \qquad (7)$$

It is to be mentioned that the current center is the point of least function value achieved so far.

Model update

When a new point is available, the current quadratic model $M^{(k)}(\boldsymbol{x})$ is updated so that the point of lowest objective function value $\boldsymbol{x}_k$ is now the center of the k^{th} trust region. The model will take the form:

$$M^{(k)}(s) = a^{(k)} + s^T b^{(k)} + \frac{1}{2} s^T B^{(k)} s, \quad s = x - x_k \text{ and } s \in \mathrm{R}^n \qquad (8)$$

The parameters: $a^{(k)}$, $\boldsymbol{b}^{(k)}$ and $\boldsymbol{B}^{(k)}$ are evaluated employing the parameter values of the previous model $M^{k-1}(\boldsymbol{x})$ in addition to all available function values. The constant $a^{(k)}$ is assigned the value of $f(\boldsymbol{x})_k$, i.e., $a^{(k)} = f(\boldsymbol{x})_k$. The model will be updated in two steps. First, the vector $\boldsymbol{b}^{(k)}$ is updated then the Hessian matrix $\boldsymbol{B}^{(k)}$ is updated as follows:

Step1: Updating the vector $\boldsymbol{b}^{(k)}$

The vector $\boldsymbol{b}^{(k)}$ can be obtained using only n points. However, using the n recent points may result in ill-conditioned system of linear equations. In order to avoid this, it is proposed to use the least squares approximation with the most recent $2n$ points. So, the vector $\boldsymbol{b}^{(k)}$ is evaluated such that the model $M^k(\boldsymbol{x})$ fits the last $2n$ points obtained, $\boldsymbol{x}_i$, $i = 1, 2, \ldots, 2n$, i.e., the following condition should be satisfied:

$$M^{(k)}(s_i) = f(s_i), \quad \text{where } s_i = x_i - x_k \qquad i = 1, 2, \ldots, 2n \qquad (9)$$

When computing the vector $\boldsymbol{b}^{(k)}$, the matrix $\boldsymbol{B}^{(k)}$ is assigned temporarily the value of the previous model Hessian matrix, $\boldsymbol{B}^{(k-1)}$, hence the vector $\boldsymbol{b}^{(k)}$ is obtained by solving the following system of linear equations:

$$\boldsymbol{A}\boldsymbol{b}^{(k)} = \boldsymbol{v}, \qquad (10)$$

where

$$A = \begin{bmatrix} s_1^T \\ s_2^T \\ \cdot \\ \cdot \\ \cdot \\ s_{2n}^T \end{bmatrix} \text{ and } v = \begin{bmatrix} f(s_1) - a^{(k)} - \frac{1}{2} s_1^T B^{(k-1)} s_1 \\ f(s_2) - a^{(k)} - \frac{1}{2} s_2^T B^{(k-1)} s_2 \\ \cdot \\ \cdot \\ \cdot \\ f(s_{2n}) - a^{(k)} - \frac{1}{2} s_{2n}^T B^{(k-1)} s_{2n} \end{bmatrix} \qquad (11)$$

The previous system is an over-determined system. The least squares approximation for $\boldsymbol{b}^{(k)}$ is

$$\boldsymbol{b}^{(k)} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{v}. \qquad (12)$$

Step2: Updating the matrix $\boldsymbol{B}^{(k)}$

The model Hessian matrix $\boldsymbol{B}^{(k)}$ is evaluated using the following updating formula:

$$\boldsymbol{B}^{(k)} = c\boldsymbol{B}^{(k-1)} + \boldsymbol{q}\boldsymbol{p}^T \qquad (13)$$

where c is a positive constant, $0.5 < c < 1$, and the vector $\boldsymbol{p} \in R^n$,

$$q = [\text{sign}(\text{diag}(B^{(k-1)}))]\sqrt{(1-c)|\text{diag}(B^{(k-1)})|} \qquad (14)$$

This choice of $\boldsymbol{q}$, ensures that changes in $\boldsymbol{B}^{(k)}$ occur gradually. The vector $\boldsymbol{p}$ is evaluated such that the model $M^{(k)}(\boldsymbol{x})$ tries to fit all the available m points obtained so far, $\boldsymbol{x}_i$, $i = 1, 2, \ldots, m$, i.e., the following condition should be satisfied

$$M^{(k)}(s_i) = f(s_i), \quad \text{where } s_i = x_i - x_k \qquad i = 1, 2, \ldots, \text{m} \qquad (15)$$

i.e., the vector $\boldsymbol{p}$ is obtained by solving the weighted system of linear equations

$$\boldsymbol{Ap}=\boldsymbol{v}, \qquad (16)$$

Where

$$A=\begin{bmatrix} \frac{1}{2}s_1^T q s_1^T w_1 \\ \frac{1}{2}s_2^T q s_2^T w_2 \\ \cdot \\ \cdot \\ \cdot \\ \frac{1}{2}s_m^T q s_m^T w_m \end{bmatrix} \text{ and } v=\begin{bmatrix} w_1(f(s_1) - a^{(k)} - s_1^T b^{(k)} - \frac{1}{2}s_1^T cB^{(k-1)}s_1) \\ w_2(f(s_2) - a^{(k)} - s_2^T b^{(k)} - \frac{1}{2}s_2^T cB^{(k-1)}s_2) \\ \cdot \\ \cdot \\ \cdot \\ w_m(f(s_m) - a^{(k)} - s_m^T b^{(k)} - \frac{1}{2}s_m^T cB^{(k-1)}s_m) \end{bmatrix} \qquad (17)$$

To obtain more accurate model in the neighborhood of the current center, the available points are assigned different weights w_i, $i = 1, 2, \ldots, m$ according to their distances from the trust region center. In the proposed approach the weight w_i, associated with each equation, takes the form:

$$w_i = \begin{cases} 1 & if\ ||s_i|| \le c_1\Delta \\ \frac{c_1\Delta}{||s_i||} & if\ ||s_i|| > c_1\Delta \end{cases} \quad ,i = 1,2,..,m, \qquad (18)$$

where c_1 is a positive constant, $c_1 \geqslant 1$.

The previous system in (16) is an over-determined system ($m > n$). The least squares approximation for $\boldsymbol{p}$ is

$$\boldsymbol{p} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\,\boldsymbol{A}^T\boldsymbol{v}. \qquad (19)$$

After getting the vector $\boldsymbol{p}$, the term $\boldsymbol{qp}^T$ is calculated and the matrix is made symmetric by resetting the off-diagonal elements to their average values, i.e., $b_{ij} = b_{ji} \leftarrow (b_{ij} + b_{ji})/2$, then the new Hessian matrix $\boldsymbol{B}^{(k)}$ is updated according to Eq. (13).

The model can be improved by generating a new point $s_{new} = x_{new} - x_k$, which is chosen to be on the boundary of the trust region so that it improves the distribution of points around the center of the trust region. A suggested solution to find s_{new} is to solve the following problem:

$$\max_s P = \sum_{i=1}^{n_s} (s_i^T s)^2 \text{ , such that } s_s^T s < 0 \quad \forall i \text{ and } \|s\| \leq \Delta \qquad (20)$$

Where s_{new} is selected to maximize the sum of squares of the projections of the vector s_{new} on the other s_i, $i = 1, 2, \ldots n_s$ vectors, where n_s is the available set of points. After generating s_{new}, the function value $f(x_{new})$ is computed. If $f(x_{new})$ is found to be less than $f(x_k)$, then x_{new} will be considered as the new trust region center of the subsequent iteration, otherwise, x_{new} will just be added to the available set of points.

Algorithm

A complete algorithm for the proposed method is given below (see also an illustrative flowchart in Fig. 1).

Step1 Set $N = 0$ (the number of function evaluations), given $x_0 \in R^n, \Delta_1 > 0, 0.5 < c < 1, c_1 \geqslant 1$, N_{max}, $\delta x_0 \in R^n$, $\Delta_1 > 0$, $0.5 < c < 1$, $c_1 \geqslant 1$, N_{max}, δ (a termination criterion).

Step 2 Find the initial m points using (2), letting x_1 be the initial trust region center, then construct the initial quadratic model using (3), Set $k = 1$.

Step 3 Solve the trust region sub-problem (4) using the truncated conjugate gradient method to obtain $s^* = x_n - x_k$ of the model $M^{(k)}(x)$ over the trust region.

Step 4. Evaluate $f(x_n)$ and compute the reduction ratio by substituting in (5).

Step 5 Update the trust region radius to obtain Δ_{k+1} using (6).

Step 6 Determine the trust region center of the next iteration x_{k+1} based on x_k and r_k using (7). If $\|f(x_{k+1}) - f(x_k)\| < \delta$, the algorithm will be terminated with $x_{opt} = x_{k+1}$ and $f_{opt} = f(x_{k+1})$. If for two successive iterations, r_k is negative go to

Step 9, else continue.

Step 7 Add the point $\boldsymbol{x}_n$ to the set of available points S, if the number of points in S exceeds $N_{\max}$, remove the farthest point from $\boldsymbol{x}_{k+1}$.

Comment. To avoid severe computational and storage overhead, a bound $N_{\max}$ is put to limit the uncontrollable increase in the number of stored points. Specifically, when the number of available points reaches $N_{\max}$ the farthest point from the trust region center is removed.

Step 8 Construct the quadratic model $M^{(k+1)}(\boldsymbol{x})$ around $\boldsymbol{x}_{k+1}$ based on $M^{(k)}(\boldsymbol{x})$ and the set of available points S using the updating procedures in Eqs. (9), (10), (11), (12), (13), (14), (15), (16), (17), (18) and (19), then set $k = k + 1$ and go to Step 3.

Step 9 Generate a new point $\boldsymbol{s}_{new}$ using (20), add it to the set of points S, then go to Step 8.

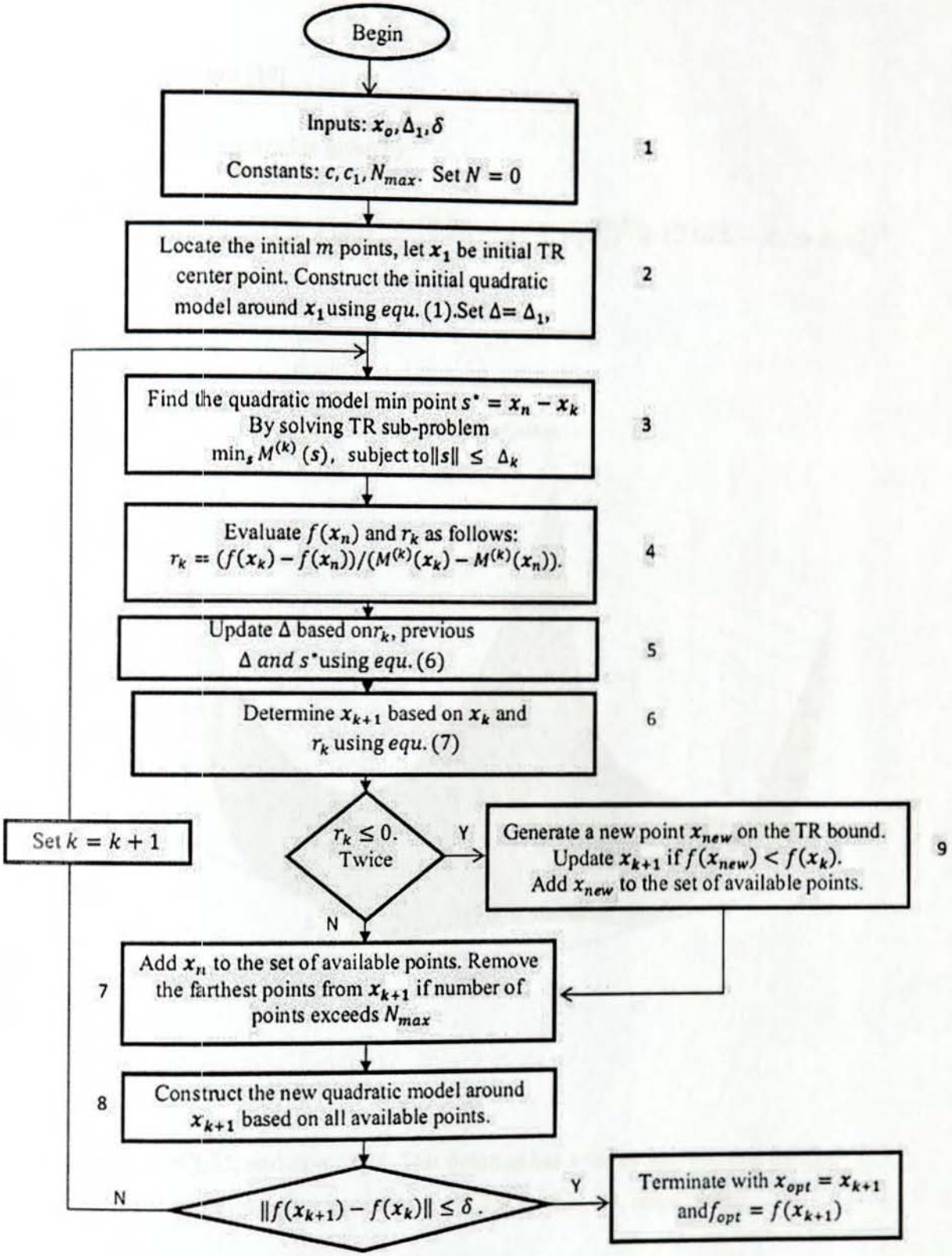


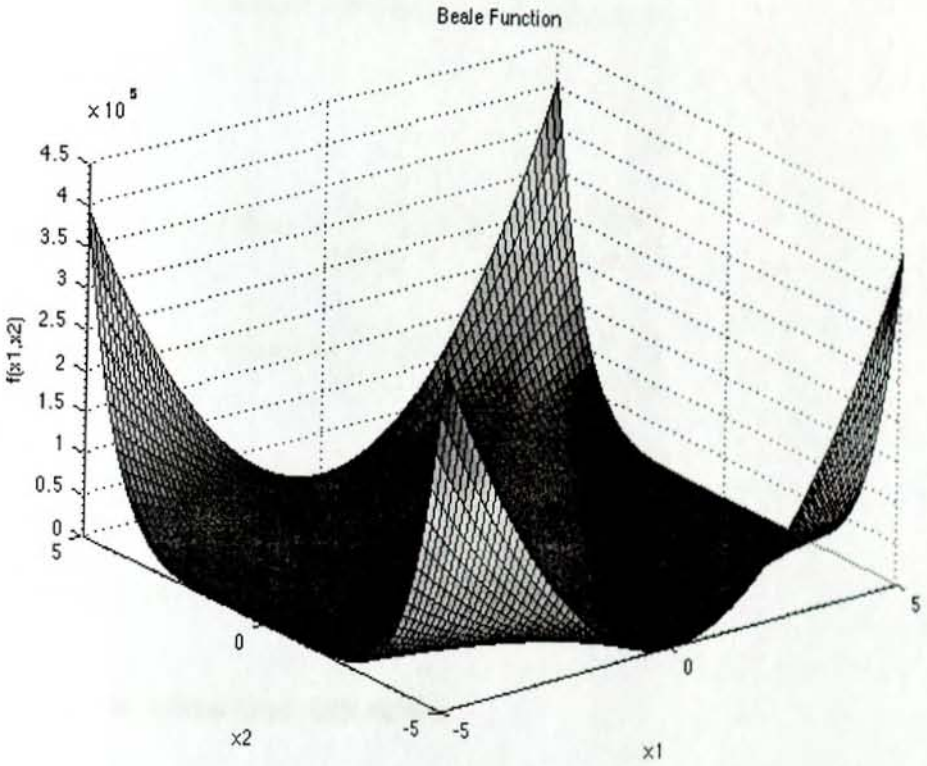
Fig 1.A flowchart for the proposed optimization algorithm.

3.2 EXAMPLE

The 2D Beale function [10]

The function to be minimized is given by

$$f(\mathbf{x}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$



where $a_1 = 1.5$, $a_2 = 2.25$, and $a_3 = 2.625$. This function has a valley approaching the line $x_2 = 1$. The initial values used for $\boldsymbol{x}_0$ and Δ_1 are $(0.1\ 0.1)^T$ and 0.8, respectively.

$min\ f(\mathrm{x}) = (1.5 - \mathrm{x}_1 + \mathrm{x}_1\mathrm{x}_2)^2 + (2.25 - \mathrm{x}_1 + \mathrm{x}_1\,\mathrm{x}_2)^2 + (2.625 - \mathrm{x}_1 + \mathrm{x}_1\mathrm{x}_2^3)^2$

$x \in R^2$

Given $x_0 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$, $\Delta_1 = 0.8$

Step 1 $N = 0$

Since $n = 2$, we need $m = 2n + 1 = 2(2) + 1 = 5$ points

The initial m points $x_i = 1, 2, \ldots 5$ can be chosen as follows

$x_1 = x_0 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$ and $\begin{cases} x_{i+1} = x_0 + \Delta_1 e_i \ldots., & i = 1, 2 \ldots n. \\ x_{i+2+1} = x_0 + \Delta_1 e_i \ldots., & i = 1, 2 \ldots n-1. \end{cases}$

Hence, $x_1 = x_0 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$

$$i = 1, \quad x_{1+1} = x_2 = x_0 + \Delta_1 e_1 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} + 0.8 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

$$i = 2, \quad x_{2+1} = x_3 = x_0 + \Delta_1 e_2 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} + 0.8 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix}$$

$$i = 1, \quad x_{1+2+1} = x_4 = x_0 - \Delta_1 e_1 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} - \begin{pmatrix} 0.8 \\ 0. \end{pmatrix} = \begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix}$$

$$i = 2, \quad x_{2+2+1} = x_5 = x_0 - \Delta_1 e_2 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} - 0.8 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0.7 \end{pmatrix}$$

Step 2:- Construct the initial quadratic model

let $x_1 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$ be the initial trust region center. Set $k = 1$

To construct the model we use

$$M^1(x_i) = f(x_i) \ldots \quad i = 1,2, \ldots, 5.$$

Where $M^1(x) = a_1 + b^T(x - x_1) + \frac{1}{2}(x - x_1)^T B (x - x_1)$ assuming B is a diagonal matrix and $b \in R^2$.

$i = 1, \quad x_i = x_1 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$

$$M^1(x_1) = f(x_1) = a_1 + b^T(0) + \frac{1}{2}(0)^T B(0) = f\begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$$

$$a_1 = (1.5 - 0.1 + (0.1)(0.1)^2 + (2.25 - 0.1 + (0.1)(0.1)^2)^2 + (2.625 - 0.1 + 0.1(0.1)^3)^2$$

$$a_1 = 1.9881 + 2.151 + (2.5251)^2$$

$$a_1 = 12.99103101$$

$i = 2, \quad x_i = x_2 = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$

$$M^1(x_2) = a_1 + (b_1 b_2) + \left(\begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}\right) + \frac{1}{2}\left(\begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}\right)^T \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}\left(\begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}\right)$$

$$= f\left(\begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}\right)$$

$$M^1(x_2) = 12.99103101 + (b_1 b_2)\begin{pmatrix} 0.8 \\ 0 \end{pmatrix} + \frac{1}{2}(0.8 \;\; 0)\begin{pmatrix} 0.8 \\ 0.1 \end{pmatrix} = f\begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

$$\Rightarrow 12.99103101 + 0.8\, b_1 + 0.32a + f\begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

$$\Rightarrow f = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}(1.5 - 0.9 + 0.9 \times 0.1)^2 + (2.25 - 0.9 + 0.9(0.1)^2)^2 + (2.625 - 0.9 + 0.9\,(0.1)3)^2$$

$$= 0.4761 + 1.846881 + 2.97873081$$

$$= 5.30171181$$

Thus, $12.99103101 + 0.8b_1 + 0.32a = 5.30171181$

$$\Rightarrow 0.8b1 + 0.32a = -7.6893192 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(1)}$$

$i = 3, \quad x_i = x_3 = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$

$$M^1(x_3) = 12.99103101 + (b_1 b_2) \begin{pmatrix} 0 \\ 0.8 \end{pmatrix} + \frac{1}{2} (0.08) \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} 0 \\ 0.8 \end{pmatrix} = f \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix}$$

$$\Rightarrow 12.99103101 + 0.8\, b_2 + 0.32b = f \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix} = 6.3189$$

$$\Rightarrow 0.8b2 + 0.32b = -6.67213101 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(2)}$$

$i = 4, \quad x_i = x_4 = \begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix}$

$$M^1(x_4) = 12.99103101 + (b_1 \;\; b_2) \left(\begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \right) +$$

$$\frac{1}{2} \left(\begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \right)^T B \left(\begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \right) = f \begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix}$$

$$\Rightarrow 12.99103101 + (b_1\, b_2) \begin{pmatrix} -0.8 \\ -0.2 \end{pmatrix} + \frac{1}{2} (-0.8 - 0.2)\, B \begin{pmatrix} -0.8 \\ -0.2 \end{pmatrix} = f \begin{pmatrix} -0.8 \\ -0.2 \end{pmatrix}$$

$$\Rightarrow 12.99103101 - 0.8b_1 - 0.2b_2 + (-0.4 - 0.1) \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} -0.8 \\ -0.2 \end{pmatrix} = f \begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix}$$

$$\Rightarrow 12.99103101 - 0.8b_1 - 0.2b_2 + 0.32a + 0.02b = f \begin{pmatrix} -0.7 \\ -0.1 \end{pmatrix}$$

$$\Rightarrow 12.99103101 - 0.8b_1 - 0.2b_2 + \; 0.32a + 0.02b = 24.87442949$$

$$\Rightarrow -0.8b_1 - 0.2b_2 + 0.32a + 0.02b = 11.88339848 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(3)}$$

$i = 5, \quad x_i = x_5 = \begin{pmatrix} -0.1 \\ -0.7 \end{pmatrix}$

$$M^1(x_5) = 12.99103101 + (b_1 \;\; b_2) \left(\begin{pmatrix} -0.1 \\ -0.7 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \right) + \frac{1}{2} \left(\begin{pmatrix} -0.1 \\ -0.7 \end{pmatrix} - \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \right)^T$$

$$B\left(\begin{pmatrix}-0.1\\-0.7\end{pmatrix}-\begin{pmatrix}0.1\\0.1\end{pmatrix}\right)=f\begin{pmatrix}-0.7\\-0.1\end{pmatrix}$$

$$\Rightarrow 12.99103101+(b_1+b_2)\begin{pmatrix}-0.2\\-0.8\end{pmatrix}+\frac{1}{2}(-0.2-0.8)\,B\begin{pmatrix}-0.2\\-0.8\end{pmatrix}=f\begin{pmatrix}-0.1\\-0.7\end{pmatrix}$$

$$\Rightarrow 12.99103101-0.2b_1-0.8b_2+(-0.1-0.4)\begin{pmatrix}a&0\\0&b\end{pmatrix}\begin{pmatrix}-0.2\\-0.8\end{pmatrix}=f\begin{pmatrix}-0.1\\-0.7\end{pmatrix}$$

$$\Rightarrow 12.99103101-0.2b_1-0.8b_2+0.02a+0.32b=f\begin{pmatrix}-0.1\\-0.7\end{pmatrix}$$

$$\Rightarrow 12.99103101-0.2b_1-0.8b_2+0.02a+0.32b=f\begin{pmatrix}-0.1\\-0.7\end{pmatrix}$$

$$\Rightarrow 12.99103101-0.2b_1-0.8b_2+0.02a+0.32b=15.32336749$$

$$-\quad 0.2b_1-0.8b_2+0.02a+0.32b=2.33233648\ \ldots\ldots\ldots\ldots\ldots\ldots\textbf{(4)}$$

Using the above four equations we have

$$\begin{pmatrix}0.8&0&0.32&0\\0&0.8&0&0.32\\-0.8&0.2&0.32&0.02\\-0.2&-0.8&0.02&0.32\end{pmatrix}\begin{pmatrix}b_1\\b_2\\a\\b\end{pmatrix}=\begin{pmatrix}-7.6893192\\-6.67213101\\11.88339848\\2.33233648\end{pmatrix}$$

Using Cramer's Rule on the above, we get

$a = 1.972$, $b = -10.095$, $b_1 = -10.4$, $b_2 = -4.302$

Thus,

$$M^1(x) = a_1 + b^T(x) + \frac{1}{2}(x)^T B(x)$$

Where $a_1 = 12.991$, $b=\begin{pmatrix}-10.4\\-4.302\end{pmatrix}\in R^2$, $B=\begin{pmatrix}a&0\\0&b\end{pmatrix}=\begin{pmatrix}1.972&0\\0&-10.095\end{pmatrix}$

Step 3:- Minimize the model

$$\min M^1(s) \qquad \text{Where } s = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \in \mathrm{R}^2,$$
$$s.t. \parallel s \parallel \leq 0.8$$

This implies

$$M^1\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = 12.991 + (-10.4 \quad -4.302)\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \frac{1}{2}(s_1 \quad s_2)\begin{pmatrix} 1.972 & 0 \\ 0 & -10.095 \end{pmatrix}\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$$

$$\left\| \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \right\| \leq 0.8$$

$$\Rightarrow M^1\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = 12.991 - 10.4s_1 - 4.302s_2 + 0.988(s_1)^2 - 5.048(s_2)^2$$

$$\sqrt{(s_1)^2 + (s_2)^2} \leq 0.8$$

The Lagrange function defined by

$$L(s_1, s_2, \mu) = 12.991 - 10.4s_1 - 4.302s_2 + 0.988(s_1)^2 - 5.048(s_2)^2 + \mu((s_1)^2 + (s_2)^2 - 0.64)$$

To solve this we use KKT condition.

1. $L_{s_1} = -10.4 + 1.976s_1 + 2\mu s_1 = 0$
2. $L_{s_2} = -4.302 - 10.096s_2 + 2\mu s_2 = 0$
3. $\mu((s_1)^2 + (s_2)^2 - 0.64) = 0$
4. $((s_1)^2 + (s_2)^2 - 0.64 \leq 0$
5. $\mu \geq 0$

Case1: let $\mu = 0$

-10.4 + 1.976s_1== and $-4.302 - 10.096s_2 = 0$

$$\Rightarrow s_1 = \frac{10.4}{1.976} = 5.279 \text{ and } s_2 = \frac{4.302}{10.096} = 0.426$$

For s1 = 5.279 and s2 = 0.426 condition 4 is not satisfied.

Case 2: let $\mu > 0$ then $(s_1)^2 + (s_2)^2 = 0.64$

$$L_{s_1} = -10.4 + 1.976s_1 + 2\mu s_1 = 0 \Rightarrow \mu = \frac{10.4 - 1.976s_1}{2s_1}$$

$$L_{s_2} = -4.302 - 10.096s_2 + 2\mu s_2 = 0 \Rightarrow \mu = \frac{4.302 + 10.096s_2}{2s_2}$$

Then $\frac{10.4 - 1.976s_1}{2s_1} = \frac{4.302 + 10.096s_2}{2s_2}$

From the above we have $s = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 0.523 \\ 0.605 \end{pmatrix}$

Step 4: Compute the reduction ratio.

The new point $x_n = x_k + s$ where $x_k = x_1 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}$ and $s = \begin{pmatrix} 0.523 \\ 0.605 \end{pmatrix}$

$$\Rightarrow x_n = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \backslash + \begin{pmatrix} 0.523 \\ 0.605 \end{pmatrix} = \begin{pmatrix} 0.623 \\ 0.705 \end{pmatrix}$$

$$f(x_k) = f(x_1) = f\begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} = 12.991$$

$$f(x_n) = f\begin{pmatrix} 0.623 \\ 0.705 \end{pmatrix} = 10.121$$

$$M^1(x_1) = M^1\begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} = 11.48$$

$$M^1(xn) = M^1\begin{pmatrix} 0.625 \\ 0.705 \end{pmatrix} = 2.0938$$

Now the reduction ratio will be

$$r_k = \frac{f(x_k) - f(x_n)}{M^1(x_k) - M^1(x_n)} \Rightarrow r_1 = \frac{12.991 - 10.121}{11.48 - 2.0938} = 0.306$$

Step 5: Update the trust region radius

$r_1 = 0.306$. Since $0.1 \le r_1 \le 0.7$, $\Delta_{k+1} = \Delta_2 = \Delta_1 = 0.8$

Step 6: Determine the trust region center

Since $r_1 = 0.306 > 0$, then the new center is

$$x_{k+1} = x_2 = x_1 + s = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} + \begin{pmatrix} 0.523 \\ 0.705 \end{pmatrix} = \begin{pmatrix} 0.623 \\ 0.705 \end{pmatrix}$$

Thus the next step is to update the model and again to minimize over the new trust region and to follow the same procedure. Thus the function has a minimum of 0 at $(3\ 0.5)^T$.The result is obtained by applying the proposed technique.

3.3 LIMITATIONS OF DERIVATIVE-FREE OPTIMIZATION.

Perhaps foremost among the limitations of derivative-free methods is that, on a serial machine it is usually not reasonable to try and optimize problems with more than a few of variables, although some of the most recent techniques can handle unconstrained problems in hundred of variables. Also, even on relatively simple and well-conditioned problems it is usually not realistic to expect accurate solutions, convergence is typically rather slow. For example, in order to eventually achieve something like quadratic rate of local convergence, one needs either implicit or explicit local models that are reasonable approximations for a second-order Taylor series model in the current neighborhood. With 100 variables, if one was using interpolation, a local quadratic function would require 5151 function evaluations as shown in Table 3.1[2]. Just as one can achieve good convergence using approximations to the Hessian matrix (such as quasi-Newton methods in the derivative- based case, it is reasonable to assume that one can achieve similar fast convergence in the derivative-free cases by using incomplete quadratic mode or quadratic models based on only first-order approximations. However, unless the functions look very similar to a quadratic in the neighborhood of the optimal solution, in order to progress the models have to be recomputed frequently as the step size and the radius of sampling converge to zero. Even in the case of linear models this can be prohibitive when functions evaluations are expensive. Thus typically one can expect a local convergence rate that is closer to linear than quadratic, and one may prefer early termination.

As for being able to tackle only problems in around 20 or moderately more variables (currently the largest unconstrained problems that can be tackled on a serial machine appears to be in several hundred variables), the usual remedy is to use statistical methods like analysis of variance to determine, say, the most critical 20 variables and optimization only over them. It may also be reasonable to take advantage of the relative simplicity of some of the derivative-free algorithms, like directional direct-search methods, and compute in a parallel, or even massively parallel, environment.

n	10	20	50	100	200
$(n+1)(n+2)/2$	66	231	1326	5151	20301

Table 3.1 Number of points needed to build a "fully quadratic' polynomial model [2].

Another limitation of derivative-free optimization may occur when minimizing non convex functions. However, and although nothing has been proved to support this statement, it is generally accepted the derivative-free optimization methods have the ability find "good" local optima in the following sense. If one has a function with an apparent large number of local optimizers, perhaps because of noise, then derivative-free approach given their relative crudeness, have a tendency to initially go to generally low regions in the early iterations (because of their myopia, or one might even say near-blindness) and in late iterations they still tend to smooth the function, whereas a more sophisticated method may well find the closest local optima to the starting point. The tendency to "Smooth" functions is also why they are effective for moderately noisy functions. There are many situations where derivative-free methods are the only suitable approach, capable of doing better than heuristic or other "last-resort" algorithms and providing a supporting convergence theory.

There are however, classes of problems for which the use of derivative-free methods that we address here are not suitable. Typically, rigorous methods for such problems would require an inordinate amount of work that growth exponentially with the size of the problem. This category of problems includes medium and large scale general global optimization problems with or without derivatives, problems which not only do not have available derivatives but which are not remotely like smooth problems, general large non-linear problems with discrete variable including many combinational optimization ones (so-called NP hard problems) and stochastic optimization problems

CONCLUSIONS

In this project, a trust region optimization method that does not require any derivative information has been proposed. In this method, the objective function is approximated via quadratic surrogates, and using few number of initial data points than the exact number of surrogate parameters. Classical benchmark test problems were used to demonstrate the accuracy and efficiency of the proposed method. The results obtained showed the ability of the proposed method to rapidly converge to the final region containing the optimum solution when only a limited number of function evaluations is permissible and when a high accuracy is not really necessary. Thus, the proposed method is suitable for stochastic optimization or objectives that suffer from numerical inaccuracy

REFERENCE

1) *Abdel-KarimS.O.Hassen, Hany L abdel-Malek, Ahmed S.A. Mohamed, Tamer M. Abdulfadel, Ahmed E. Elqenawy. RF Cavity design expoiting a new derivative-free trust region optimization approach; J. Adv Res(2014), pp. 4 - 11*

2) *A. R. Conn, K. Schinberg, L. N. Vicente. Introduction to Derivative free Optimization; SIAM Books(2009)*

3) *Abebe Geletu. Solving Optimization Problems using the Matlab Optimization Toolbox – a Tutorial, Tu-Ilmenau, Fakultaet fuer Mathematik and Naturwissenschaften (December 13, 2007)*

4) *A.S.O. Hassan, H.L. Abdel-Malek, A.A. RabieNon-derivative design centering algorithm using trust region optimization and variance reduction Eng Opt, 38 (2006), pp. 37–51*

5) *M. J. D. Powell. A direct search optimization method that models the objective by quadratic and constraint functions by linear interpolation. In Advances in Optimization and Numerical Analysis, Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico, volume 275, pages 51–67, Dordrecht, NL, 1994. Kluwer Academic Publishers.*

6) *M. J. D. Powell. A view of algorithms for optimization without derivatives. Technical report NA 2007/03. University of Cambridge, Department of Applied Mathematics and Theoretical Physics, Cambridge, England; 2007.*

7) *T. Steihaug. The conjugate gradient method and trust regions in large scale optimization SIAM J Numer Anal, 20 (1983), pp. 626–637*

8) *JW Bandler, HL Abdel-Malek. Optimal centering, tolerancing and yield determination using multidimensional approximation. In: Proc IEEE international symposium on circuits and systems (Phoenix, Arizona); April 1977. p. 219–22.*

9) *EML Beale. On an iterative method of finding a local minimum of a function of more than one variable. Tech. Rep. No. 25, Statistical Techniques Research Group, Princeton Univ., Princeton, N.J.; 1958.*

Declaration

I the undersigned, declare that this project is my original work and has not been presented of submitted partially of in full by any other person for a degree in any university, and that all sources of materials used for the purpose of this project have been duly acknowledged

Declared by

Name Tewodros Negash

Sign _____

Date _____