Addis Ababa University

School of Graduate Studies

Department of Mathematics

A Graduate project report

On

Derivative Free Optimization

By: Teklebirhan Abraha

A Project submitted to the School of Graduate Studies
of Addis Ababa University in Partial fulfillment of the requirements for the Degree of Master of Science in Mathematics

Advisor: Semu Mitku (Ph.D)

Addis Ababa, Ethiopia

January, 2011

Acknowledgment

With much pleasure, I take the opportunity to thank all the people who have helped me through the course of my graduate study.

First and foremost I would like to thank my advisor and instructor, Dr. Semu Mitku, for his in valuable guidance, a dvice, e ncouragement a nd pr oviding m e pl enty of m aterials related to my project work, without which this well organized (compiled) project would have be en i mpossible. H is c onstant c onfidence, pe rsistent que stioning a nd de ep knowledge of the subject matter have been and will always be inspiring to me.

Finally I would like to express my gratitude to all my; families, friends, and those who supported me in any means to complete this project work.

T/Birhan Abraha

A.A.U

January, 2011

Abstract

We will present derivative free algorithms which optimize non-linear unconstrained optimization problems of the following kind:

$$\min_{x \in \mathbb{R}^n} f(x) \ \ where, f: \mathbb{R}^n \to \mathbb{R}$$

The algorithms developed for this type of problems are categorized as one-dimensional search (golden section and Fibonacci) methods and multidimensional search methods (Powell's method and trust region). These algorithms will, hopefully, find the value of x for which f is the lowest.

The dimension n of the search space must be lower than some number (say 100). We do NOT have to know the derivatives of $f(x)$. We must only have a code which evaluates $f(x)$ for a given value of x. Each component of the vector x must be a continuous real parameter of $f(x)$.

Table of contents

Declaration Letter

I, Teklebirhan Abraha, declare that this project has been composed by me and that no part of the project has formed the basis for the award of any Degree, Diploma, Associate ship, Fellowship or any other similar title to me.

T/Birhan Abraha

_____

Addis Ababa University

January, 2011

Permission Letter

This is to certify that this project is compiled by Mr. Teklebirhan Abraha in the department of Mathematics, College of Mathematics and Computational Sciences, Addis Ababa University, under my supervision.

Semu Mitku (Ph.D)

_____

Addis Ababa University

January, 2011

Introduction

In the process of solving optimization problems, it is well known that expensive useful information is contained in the derivatives of the objective function one wishes to optimize. After all the standard mathematical characterization of a local minimum given by the first order necessary conditions, requires a continuous differentiability of functions that the first order derivatives are zero. However, for a variety of reasons there have always been many instances where (at least some) derivatives are unavailable or unreliable. Nevertheless, under such circumstances, it may still be desirable to carry out optimization [1].

Consequently classes of nonlinear optimization techniques called derivative-free optimization methods are needed. In fact we consider optimization techniques without derivatives as one of the most important and challenging areas in computational science and engineering. Derivative free optimization (DFO) is developed to date for solving small dimensional problems(less than 100 variables) in which the computation of an objective function is relatively expensive and the derivatives of the objective functions are not available. Problems of this nature more and more arise in modern physical, chemical and econometric measurements and in engineering applications where computer simulation is employed for the evaluation of the objective function [6].

There are two important components of derivative free methods, sampling better points in the iteration procedure is the first one of these components. The other one is searching appropriate subspaces where the chance of finding a minimum is relatively high. In order to be able to use the extensive convergence theory for derivative based methods, these derivative free methods need to satisfy some properties. For instance, to guarantee the convergence of a derivative free method, we need to ensure that the error in the gradient converges to zero when the trust region or line search steps are reduced. Hence a descent step will be found if the gradient of the trust function is not zero at the current iterate.

The problem of minimizing a nonlinear function $f: \mathbb{R}^n \to \mathbb{R}$ of several variables when the derivatives of the function are not available is attempted to be solved by the derivative free methods (DFM). The formal statement for the above problem can be written as

$$(P) \qquad \begin{aligned} &\min f(x) \\ &\text{s.t } a_i \le g_i(x) \le b_i \qquad (i = 1,2 \dots, m) \qquad (*) \\ &x \in X \subseteq \mathbb{R}^n \end{aligned}$$

Where $\nabla f(x)$cannot be computed or just does not exist for every x. Here x is an arbitrary subset of $\mathbb{R}^n$, $x \in X$ is an easy constraint. While the functions $g_i(x)$ $(i = 1, 2, \dots, m)$ represent difficult constraints. By easy constraints we mean bound constraints on the variables, linear constraints or more generally nonlinear smooth constraints whose values and the Jacobi matrix can be computed cheaply(easily).

Difficult constraints are only nonlinear constraints whose value is expensive (difficult) to compute and whose derivatives are unavailable for a small repetition and preparation, we include some basic concepts of differentiable optimization in this project.

If the problem function in $(*)$ above is differentiable then x_0 is local minimizer of (P) and hence $\nabla f(x_0) = 0$ and if $\nabla f(x_0) = 0$ and $\nabla^2 f(x_0)$ is positive definite then x_0 is a local minimizer of f. But in most of the cases finding algebraically a point $x_0 \in X \subseteq \mathbb{R}^n$ such that $\nabla f(x) = 0$ may be difficult. Or computing $\nabla f(x)$ or $\nabla^2 f(x)$ may be expensive as the function cannot (may not) be defined explicitly or even the function may not be differentiable at all. In this case numerical methods with derivative free algorithms are required. The methods such as line search and trust region methods will be discussed in this project. Because the line search without derivatives and the trust-region methods algorithms are used to solve Optimization problems without derivatives.

The first chapter of this project deals with nonlinear optimization problems and the methods of solving these problems. The second and the third chapters are on derivative free optimization methods. Particularly in the second chapter we include line search methods which help us to solve one dimensional minimization problems such as the

golden section method and the Fibonacci method, and best known methods for minimization of multidimensional search methods like Powell's method are discussed well. Finally in the last chapter we include the trust region method which is one of the methods for derivative free optimization methods.

Chapter 1
Preliminary concepts

1.1 Nonlinear optimization methods

Optimization methods can be classified as derivative based and derivative free methods depending on their use of derivative (or absence of it) in the process of finding a solution.

Derivative based optimization methods are characterized by: Explicitly uses the derivatives of the objective function, analytical solution is possible, and convergence is faster. And derivative free optimization methods are characterized by:only objective function evaluation, not require derivative information, and handle noisy functions(since methods only rely on function comparisons)

Need for derivative free optimization

Some of the reasons to apply derivative free optimization (DFO) methods are: Growing sophistication of computer hard ware and mathematical algorithms and software(which opens new possibilities for optimization), Derivative evaluations costy and noisy(one can't trust derivatives or approximate them by finite difference), Binary codes (source codes not available or owned by a company) making automatic differentiation impossible to apply, Legacy codes(written in the past and not maintained by the original author), Lack of sophistication of the user(the user need improvement but want to use something simple)

With the current state of the art DFO methods we can successfully address problems where:

- The evaluation of derivatives is expensive and/or computed with noise (and for which accurate finite difference derivative estimation is ruled out).
- The number of variables do not exceed ,say , a hundred(in serial computation)
- The functions are not excessively non smooth.
- Rapid assumption convergence is not of primary importance
- Only a few digits of accuracy are required

It is hard to minimize non convex functions without derivatives, however, it is generally accepted that DFO methods have the ability to find "good" local optimization

1.2 Overview of differentiable nonlinear unconstrained optimization methods

1.2.1 The problem

Optimization problems can be divided into two large classes, namely constrained and unconstrained problems. The basic unconstrained optimization problem can be stated in its standard form as

$$minimize\ f(x), subject\ to\ x\epsilon\mathbb{R}^n \qquad (1.1)$$

Where $f: \mathbb{R}^n \to \mathbb{R}$ is the objective function. On the other hand, constrained optimization problems can be written as

$$minimize \quad f(x) \qquad (1.2a)$$

$$subject\ to\ \ x\epsilon\mathbb{R}^n, \qquad (1.2b)$$

$$g_i(x) \leq 0 \ \ i \in I \qquad (1.2c)$$

$$g_i(x) = 0 \quad i \in \mathcal{E} \qquad (1.2d)$$

conditions $(1.2b - 1.2d)$ indicate the constraints. The disjoint index sets $I\ and\ \mathcal{E}$ correspond to the inequality and equality constraints, respectively, defined by the functions $g_i: \mathbb{R}^n \to \mathbb{R}$, $i \in I \cup \mathcal{E}$. the set X is contained in $\mathbb{R}^n$ and is also contained in the domain of f and $g_i, i \in I \cup \mathcal{E}$.

A point $x\epsilon X$ is said to be feasible if it satisfies all the constraints. And the set of all feasible points is called the feasible set, and denoted by $\mathcal{F}$.

The formulations (1.1) and (1.2) are called standard formulations due to the observation that

$$\max f(x) = -\min(-f(x))$$

1.2.2. Solution concepts

The solution of an optimization problem can be characterized by certain properties. In a minimization problem, if we are looking for a point x^* *in the domain* $\mathfrak{D}$ of f such that

$f(x^*) \leq f(x)$for all $x \in \mathfrak{D}$,

Then x^* is called a global minimizer, where as $f(x^*)$ is the global minimum of f. Similarly in, a constrained problem, the solution must lie in the feasible set $\mathcal{F}$, and thus, a global constrained minimizer x^* satisfies

$$f(x^*) \leq f(x) \text{ for all } x \in \mathcal{F}.$$

However, in both cases, finding a global minimizer of a function f can prove to be very difficult in practice. It might be interesting, thus, to look for a solution x^* in a neighborhood $\aleph$ of x^*, $\aleph \subseteq \mathfrak{D}$ such that

$$f(x^*) \leq f(x), \text{ for all (feasible) } x \text{ in } \aleph \qquad (1.3)$$

The point x^* is then called a local minimizer where as $f(x^*)$ is a local minimum of f in $\aleph$. If x^* is such that

$$f(x^*) < f(x), \text{ for all (feasible) } x \in \aleph \qquad (1.4)$$

Then x^* is said to be a strict local minimizer, and $f(x^*)$ is strict local minimum of f in $\aleph$.

1.2.3 Basic concepts of the methods

Derivatives and Taylor's Theorem

All methods considered here are based on the fact that the function to be minimized has derivatives. If f is differentiable and all derivatives of f are continuous with respect to x, then we say that f is continuously differentiable and is denoted by$f \in C^1$.

If all the second partial derivatives of f exist, then f is said to be twice differentiable. If furthermore all second partial derivatives of f are continuous we say that f is twice continuously differentiable and is denoted by$f \in C^2$.

The gradient of f is a vector that groups all its partial derivatives and is denoted by

$$\begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$$

The $n \times n\, matrix$ defined as $\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{x_n^2} \end{pmatrix}$ is called the Hessian matrix of f.

The curvature of f at $x \epsilon \mathbb{R}^n$ along a direction $d \epsilon \mathbb{R}^n$ is given by

$$\frac{\langle d, \nabla^2 \mathrm{f(x)d} \rangle}{\|d\|}$$

If $\nabla^2 f(x)$ is a positive-semi definite for every x in the domain of f .we say that f is a convex function. If $\nabla^2 f(x)$ is positive definite in its domain we say that f is strictly convex.

Theorem1.1 (Taylor's Theorem)

Let $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ be continuously differentiable and $s \epsilon \mathbb{R}^n$ then there exists some $t \epsilon [0,1]$such that

$f(x+s) = f(x) + \langle \nabla f(x+ts), s \rangle$. Moreover, if f is twice continuously differentiable, then $\nabla f(x+s) = \nabla f(x) + \int_0^1 \nabla^2 f(x+ts) s dt$

Furthermore, we have that for some $t \epsilon (0,1)$

$$f(x+s) = f(x) + \langle \nabla f(x), s \rangle + \frac{1}{2} \langle s, \nabla^2 f(x+ts) s \rangle \qquad (1.5)$$

A function $\mathbb{R}^n \rightarrow \mathbb{R}^m$ is said to be differentiable if all its partial derivatives $\partial f_j(x)/(\partial x_i)$ exist.

1.2.4 Necessary conditions for solutions of differentiable nonlinear optimization problems *(The case of unconstrained problems)*

All the unconstrained minimization methods are iterative in nature and hence they start from an initial trial solution and proceed toward the minimum point in a sequential manner. The iterative process is given by

$$x_{i+1} = x_i + \alpha_i s_i \qquad (1.6)$$

Where x_i is the starting point, S_i is the search direction, α_k is the optimal step length, and x_{i+1} is the final point in iteration i. It is important to note that all the unconstrained minimization methods (1) require an initial point x_1 to start the iterative procedure, and (2) differ from one another only in the method of generating the new point x_{i+1}(from x_i) and in testing the point $x_{i+1}$for optimality.

Rate of Convergence

Different iterative optimization methods have different rates of convergence. In general, an optimization method is said to have convergence of order p if

$$\frac{\|x_{i+1} - x^*\|}{\|x_i - x^*\|^p} \leq k, k \geq 0, p \geq 1 \qquad (1.7)$$

where $x_i$and x_{i+1} denote the points obtained at the end of iterations i and $i + 1$, respectively, x^*represents the optimum point, and $||x||$ denotes the length or norm of the vector x:

$$||x|| = \sqrt{(x_1^2 + x_2^2 + \cdots + x_n^2)}$$

If $p = 1$ and $0 \leq k \leq 1$, the method is said to be *linearly convergent* (corresponds to slow convergence). If $p = 2$, the method is said to be *quadratically convergent* (corresponds to fast convergence). An optimization method is said to have *super linear convergence* (corresponds to fast convergence) if

$$\lim_{i \to \infty} \frac{\|x_{i+1} - x^*\|}{\|x_i - x^*\|} \to 0 \qquad (1.8)$$

The definitions of rates of convergence given in Eqs. (1.7) and (1.8) are applicable to single variable as well as multivariable optimization problems. In the case of single-variable problems, the vector, x_i for example, degenerates to a scalar, x_i.

Theorem 1.2 (First- Order Necessary Conditions) If $x^* is$ a l ocal minimizer of $f: \mathbb{R}^n \longrightarrow \mathbb{R}$, where f is a continuously differentiable in an open neighborhood ℵ of x^*, then

$$\nabla f(x^*) = 0 \qquad (1.9)$$

$if\ \nabla^2(x)$ Exists a nd i s c ontinuous i n a ne ighborhood of x^*, w e ca n st ate an other necessary condition satisfied by a local minimizer.

Theorem 1.3 (Second order necessary conditions) If x^* is a local minimizer of f, and f is twice continuously differentiable in an open neighborhood ℵ of x^*, then

$$\nabla f(x^*)=0 \text{ and } \nabla^2 f(x^*) \text{ is positive-semi definite.} \qquad (1.10)$$

Any point x^* that satisfies (1.9) is called a stationary point of f. Thus Theorem1.2 states that any local minimizer must be a stationary point; it is important to note, however, that the converse is not necessarily true. Fortunately, if the next conditions called sufficient conditions are satisfied by a stationary point x^* they guarantee that it is a local minimizer.

Theorem 1.4 (Second -Order Sufficient Conditions): Let f be t wice c ontinuously differentiable on an open neighborhood ℵ of x^*. If x^* satisfies

$\nabla f(x^*) = 0$, and $\nabla^2 f(x^*)$ is positive definite then x^* is a strict local minimizer of f.

Note that the second order sufficient conditions are not necessary: a point can be a strict local minimizer and fail to satisfy the sufficient conditions.

1.3 Methods of solving nonlinear differentiable optimization problems

1. 3. 1 Gradient method (first order derivative method)

Mathematical pr ogramming i s c oncerned w ith methods t hat c an be us ed t o s olve optimization problems. In practice we will be concerned with algorithms, defined so that their computational implementation finds either an approximate an exact solution to the original mathematical programming problem [7].

These algorithms are mostly iterative methods, which form a starting point x_0, use some rule to compute a sequence of points $\{x_1, x_2, \dots, x_k, \dots\}$ such that

$$\lim_{k\to\infty} x_k = x^*,$$

where x^* is the solution to the problem.

Here we are mostly interested in unconstrained problems. The simplest method developed for the solution of a minimization problem is the steepest descent method. This method is based on the fact that, from any starting point x, the direction in which any function f decrease most rapidly is the direction $-\nabla f(x)$.

***Definition*1.1**(*descent direction*) let $f: \mathbb{R}^n \to \mathbb{R}$ be differentiable at $\bar{x} \in \mathbb{R}^n$ $then$ $0 \neq d \in \mathbb{R}^n$ is a descent direction of f at $\bar{x}$ if there exists $\delta > 0$ such that $f(\bar{x} + \lambda d) < f(\bar{x})$ for all $\lambda \in (0, \delta]$.

Consider

$$\min_{x\in\mathbb{R}^n} \text{f(x) where f} : \mathbb{R}^n \to \mathbb{R}$$

If f is continuously differentiable at $x_k \in \mathbb{R}^n$ and if $\nabla f(\bar{x}) \neq 0$ then there are infinitely many descent direction of f at x_k that is there exists $d_k \in \mathbb{R}^n / \{0\}$ such that $\nabla f(x_k)^T d_k < 0$.

To find the direction in which f decreases most rapidly we solve the problem

$\min_{d_k\in\mathbb{R}^n} \nabla f(x_k)^T d_k$ $with$ $\|d_k\| = 1$ to get

$$d_k = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$$

Now $d_k = -\nabla f(x_k)$ or $d_k = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$, will be , used for searching directions. This method is called the steepest descent method. The following algorithm was given by Cauchy [8].

Algorithm (*Method of steepest descent*)

Given $f: \mathbb{R}^n \to \mathbb{R}$ continuously differentiable,$x^* \in \mathbb{R}^n$ at each iteration k, find the lowest point of f in the direction of $-\nabla f(x_k)$ form x_k that is find α_k that solves

$$\min_{\alpha>0} f(x_k - \alpha_k \nabla f(x_k)), then\ x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

And then we continue until $\nabla f(x_k) = 0$.

If $\nabla f(x_k) = 0$, then $x_{k+1} = x_k$, the algorithm stops. However x_k may be either local minimizer or saddle point i.e., $\nabla^2 f(x_k)$ is either positive semi-definite or indefinite.

We know that from second order Taylor expansion for sufficiently small α at x_k

$$f(x_k + \alpha x_k) = f(x_k) + \alpha \nabla f(x_k)^T d_k + \frac{1}{2}\alpha^2 {d_k}^T \nabla f(x_k) d_k$$

If the searching direction is the eigen vector corresponding to the largest negative eigen value say x for the corresponding value λ of $\nabla^2 f(x_k) at\ x_k\ where\ \nabla f(x_k) = 0$, we have

$$f(x_k + \alpha x) = f(x_k) + \alpha \nabla f(x_k)^T x + \frac{1}{2}\alpha^2 x^T \nabla^2 f(x_k) x \leq f(x_k)$$

This implies that $f(x_k + \alpha x) < f(x_k)$.

The steepest descent method is a line search method that moves along $d_k = -\nabla f(x_k)$ at each step it can choose the step length α_k in various ways. One advantage of this method is it requires only calculations of $\nabla f(x_k)$ but it doe snot require second derivative. However it can be slow on difficult problems, that is this method usually works quite well during the early steps of the optimization process, depending on the point of initialization. The method of steepest descent is the simplest of the gradient methods. The choice of direction is where f decreases most quickly which is the direction opposite to $\nabla f(x_k)$. the search starts at arbitrary point x_0 and then slides down the gradient until we close enough to the solution. The iterative procedure is

$x_{k+1} = x_k - \alpha_k \nabla \mathrm{f}(x_k) = x_k - \alpha_k g(x_k)$, where $g(x_k)$ is the gradient at one given point.

Now the question is how big should the step taken in that direction be, that is what is the value of α_k? obviously we want to move the point where the function f takes on a

minimum v alue, w hich i s w here t he d irectional d erivative i s zer o. The d irectional derivative is given by

$\frac{d}{d\alpha_k} f(x_{k+1}) = \nabla f(x_{k+1})^T \cdot \frac{d}{d\alpha_k} x_{k+1} = -\nabla f(x_{k+1})^T \cdot g(x_k)$ Setting t his e xpression t o zero, we see that α_k should be choosen so that $\nabla f(x_{k+1}) and\ g(x_k)$ are orthogonal. The next step is taken in the direction of the negative gradient at this new point and we may get a zigzag pattern.

However the steepest descent method can be extremely slow for some problems. There are f ortunately, s everal ot her m ethods t hat w ork ve ry w ell i n p ractice, a nd he re w e present some of them.

1.3.2 Newton's method (second order derivative method)

Consider any nonlinear system of equations of the form

$$F(x) = 0 \qquad (1.14)$$

$Where$ F: $\mathbb{R}^n \rightarrow \mathbb{R}^n$. If the Jacobi of F exists, then we can write the Taylor first- order approximation to this function as

$$F(x+s) \approx F(x) + J(x)s \qquad (1.15)$$

Where $J(x)$ denotes the Jacobean of F evaluated at x. form these equations, we can derive an iterative method. Given an initial point x_0, at each iteration k, we will compute a new iterate $x_{k+1} = x_k + s_k$ such that $F(x_k + s_k) = 0$ which means that s_k must satisfy the linear system

$$J(x_k)s_k = -F(x_k)$$

This is called the **Newton's method** for solving nonlinear systems of equations.

Now r eturning t o our o ptimization problem, w e not e that when t he f irst a nd s econd derivatives of f are av ailable, we c an us e N ewton's m ethod t o s olve t he (possible nonlinear) system of equations defined by

$$\nabla f(x) = 0. \qquad (1.16)$$

Since we know form Theorem 1.2 above that any minimizer of f must satisfy this condition. This is the basis for the Newton's method for optimization problems, and with some variables, it is the basis for many other methods in unconstrained optimization. More formally if we want to apply this method to equation (1.16). we also know that the second order Taylor approximation to f at x_k is

$$f(x_k + s_k) \approx f(x_k) + \langle \nabla f(x_k), s_k \rangle + \frac{1}{2}\langle s_k, \nabla^2 f(x_k) s_k \rangle \qquad (1.17)$$

In order to find a minimum of this function, we try to find a solution to $\nabla f(x_k + s_k) = 0$ which is equivalent to

$$\nabla f(x_k) + \nabla^2 f(x_k) s_k = 0.$$

Thus we have that s_k must satisfy the so called Newton equations

$$\nabla^2 f(x_k) s_k = -\nabla f(x_k) \qquad (1.18)$$

If $\nabla^2 f(x_k)$ positive definite (PD), then we can find its inverse, and the solution to (1.17) is

$$s_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \qquad (1.19)$$

This direction s_k is called the Newton direction.

If $\nabla^2 f(x_k)$ is not positive definite the Newton's method (direction) is not suitable for search direction. One of the strategy to overcome such problem is modification of the Hessian matrix $\nabla^2 f(x_k)$ during or before the method of solution $\nabla^2 f(x_k) p_k = -\nabla f(x_k)$ so that it becomes positive definite.

One of the problems in Newton's method is that the method is based on a necessary first order optimality condition (namely that the gradient of the objective function be equal to zero). In order to guarantee that we have found a minimizer of f, it is also necessary to guarantee that the Hessian $\nabla^2 f(x^*)$ be positive definite moreover the approximations

(1.15) and 1.17) are only valid in a neighborhood of the solution of (1.15) and (1.16) respectively.

Thus Newton's method is only appropriate when the starting point x_0 is sufficiently close to the solution x^*. However, when it works, it is very fast and most optimization methods try to mimic its behavior around the solution.

There are so called globalization techniques that can be used to guarantee the convergence of Newton's method from any starting point. These techniques give rise to different methods which can be divided in to two classes: line search methods and trust region methods. The main difference between these two classes is that in line search methods, the direction in which we choose to take our next iteration is selected first, while the size of the step to be taken in this direction is computed with the direction fixed. On the other hand, in trust region methods, the step size and the direction are more or less chosen simultaneously [7]. The two strategies are discussed below in more details.

1.3.3 Line search methods

Although Newton's direction is effective for searching directions, the method may not converge to the solution. The local model may not be a good representative for the given (objective) function. Hence we have to backtrack. The strategy we consider for proceeding from a solution estimate outside the convergence of Newton's method is the method of line searches.

As the name suggests, the idea behind line search methods is to find a step size along a certain line which gives us a good reduction on the function value, while being reasonably in expensive to compute. More formally, they are iterative methods that, at every step, choose a certain descent direction and move along this direction. Each iteration of a line search method computes a search direction p_k and then decides how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k$$

Where, the positive scalar $\alpha_{k,}$ is called a step length, and p_k can be chosen, as the Newton direction s_k given by (1.19).

Moreover, the search direction often has the form $p_k = -\beta_k^{-1}\nabla f(x_k)$ (1.20)

Where β_k is a symmetric and nonsingular matrix. In the steepest descent method β_k is simply the identity matrix I, while in Newton's method β_k is the exact Hessian $\nabla^2 f(x_k)$. in Quasi Newton method, β_k is an approximation to the Hessian that is updated at every iteration, by means of a low rank formula. Where p_k is defined by (1.20) and β_k is positive definite we have

$p_k^T \nabla f(x_k) = -\nabla f(x_k)^T \beta_k^{-1} \nabla f(x_k) < 0.$

and therefore p_k is a descent direction.

Now in the following we study how to choose the matrix β_k or more generally how to compute the search direction, and give careful consideration to the choice of the step length parameter α_k.

Step length: the ideal choice of the step length α_k would be the global minimizer of the univariate function $\phi(.)$ defined by

$\phi(\alpha) = f(x_k + \alpha p_k), \alpha > 0$ (1.21)

But in general, it is too expensive to identify this value. To find even a local minimizer of ϕ to moderate precision generally requires too many evaluations of the objective function f and possibly the gradient ∇f.

In order to ensure that even an approximate solution is enough to guarantee the convergence of the line search method to a minimizer of the objective function, some conditions are imposed on the step length at each iteration. One very important condition that must be satisfied is that the decrease in the objective function is not too small. One way of measuring this is by using the following inequality.

$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \langle \nabla f(x_k), p_k \rangle$, for some $c_1 \in (0,1)$ (1.22)

This condition is sometimes called the Armijo condition and it states that the reduction in f should be proportional to the step length α_k and the derivative of f. On the other hand;

we must also guarantee that the step is not too short. Indeed, condition (1.22 is satisfied for a ll s ufficiently s mall v alues o f α. O ne w ay of e nforcing t his i s by i mposing a curvature condition, which requires that α_k satisfy

$\langle \nabla f(x_k + \alpha_k p_k), p_k \rangle \geq c_2 \langle \nabla f(x_k), p_k \rangle$ for some $c_2 \in (c_1, 1)$ (1.23)

Conditions (1.22) and (1.23) are known as the Wolfe conditions.

Remark: for every function f that is smooth and bounded below, there exist step lengths that satisfy the Wolfe conditions.

These conditions on the step length are very important in practice and are widely used in the line search methods.

Algorithm (*line search*)

Given a d escent d irection p_k, set $x_{k+1} = x_k + \alpha_k p_k$ for so me $\alpha_k > 0$ that m akes x_{k+1} acceptable i terate. H owever, r ather t han set ting $p_k = -\nabla f(x_k)$ we will u se N ewton's direction o r its m odification f or in stance,$-H_k^{-1}\nabla f(x_k)$ where $H_k = \nabla^2 f(x_k) + N_k$ is positive de finite, $N_k = 0$ if $\nabla^2 f(x_k)$ is s afely p ositive d efinite to re tain i ts f ast lo cal convergence t he term l ine se arch refers to t he ch oice of α_k in this a lgorithm. T he common procedure is to use $\alpha_k = 1$ (the full Quassi-Newton's step) i f it fails backtrack in t he symmetric w ay a long t he s ame di rection, t he c ommon s ense i s t o r equire t hat $f(x_{k+1}) < f(x_k)$ but this simple condition doesnot guarantee that $\{x_k\}$ will converge to the m inimizer o f f. I f ve ry s mall r eduction i n f are ta ken re lative to th e le ngth o f th e steps, the limiting value $f(x_k)$ may not be local minimize [8].

For example, $f(x) = x^2, x_0 = 2$, if we choose $\{p_k\} = \{(-1)^{k+1}\}$

$\{\alpha_k\} = \{2 + 3(2^{-(k+1)})\}$, then

$\{x_k\} = \left\{2, -\frac{3}{2}, \frac{5}{4}, -\frac{9}{8}, \dots, \right\} = \{(-1^{\mathrm{k}})(1 + 2^{-\mathrm{k}})\}$, each $\mathrm{p_k}$ is a descent direction at $\mathrm{x_k}$.

And $f(x_{k+1}) < f(x_k), and\ f(x_k)$ is monotonically decreasing with

$$\lim_{k\to\infty} f(x_k) = 1$$

Which is not the minimizer of f and $\{x_k\}$ has limit ± 1 so it does not converge.We fix this by requiring that the average rate of decrease from $f(x_k)$ to $f(x_{k+1})$ be at least some prescribed fraction of the initial rate, that is we pick $\lambda \in (0,1)$ and choose α_k among $\alpha > 0$ that satisfies

$$f(x_k + \alpha p_k) \leq \lambda\alpha\nabla f(x_k)^T p_k \qquad (1.24)$$

It is also possible that if steps are too small relative to the initial rate of decrease of f, then any limiting value $\{x_k\}$ may not be local minimizer of f. For example in the above example taking the same function with the same initial estimate and let us take $\{p_k\} = \{-1\}, \{\alpha_k\} = \{2^{-(k+1)}\}$, then the sequence $\{x_k\} = \left\{2, \frac{3}{2}, \frac{5}{4}, \frac{9}{8}, \dots, \right\} = \{1 + 2^{-k}\}$. each p_k is a descent direction at x_k, $f(x_k)$ is monotonically decreasing with

$$\lim_{k\to\infty} x_k = 1$$

Which is not the minimizer of f. In the above example, we see that monotonically decreasing sequence of iterates that does not converge to the minimizer, to ensure sufficiently large steps we require that the rate of decrease of f in the direction of p at x_{k+1} be larger than some prescribed fraction of the rate of decrease of f in the direction of p at x_k

$$\nabla f(x_{k+1})^T p \geq \beta\nabla f(x_{k+1})^T p \text{ for some } \beta \in (\lambda, 1) \qquad (1.25)$$

But this condition is not necessarily required, because the use of the backtracking strategy avoids excessively small steps.

Step selection by backtracking

Here the strategy is to start with $\alpha_k = 1$, and then if $x_{k+1} = x_k + \alpha_k p_k$ is not acceptable, backtracking (reducing α_k) until an acceptable $x_{k+1} = x_k + \alpha_k p_k$ is found.

Example1.1: consider the problem

$$minimize\ f(x_1, x_2) = 5x_1^2 + x_2^2 - 3x_1x_2,$$

Let $x_k = (2,3)^T$ and $p_k = (-5,-7)^T$, s o t hat $f(x_k) = 65$.if $\alpha_k = 1$, then $f(x_k +$ *akpk=121>f(xk)*

So this is not an acceptable step length. If $\alpha_k = \frac{1}{2}$ then $f(x_k + \alpha_k p_k) = f\left(-\frac{1}{2}, -\frac{1}{2}\right) = \frac{9}{4}$

and so this step length produce a decrease in the function value, as desired. The frame work of this algorithm is given below.

Algorithm (backtracking line search frame work)

Given$\lambda \in \left(0, \frac{1}{2}\right), 0 < l < u < 1, \alpha_k = 1$

while $f(x_k + \alpha_k p_k) < f(x_k) + \lambda\alpha_k \nabla f(x_k)^T p_k$ then $\alpha = \rho\alpha_k$ for s ome $\rho \in [l, u]$. ρ is choosen at each iteration by line search, $x_{k+1} = x_k + \alpha_k p_k$. *in practice* λ is set to be very small so that no more functional value is required [8].

Strategy for choosing $\boldsymbol{\alpha_k}(\boldsymbol{\rho})$

Define $\theta(\alpha) = f(x_k + \alpha p_k)$ which, is one dimensional restriction of f to the line through x_k in the direction p_k, if we need to backtrack we will use our most current information about f to model it and then take the value of α that minimizes the model as a next value of α_k in the above algorithm.

Initially we have $\theta(0) = f(x_k)$ and $\dot{\theta}(0) = \nabla f(x_k)^T p_k$ (1.26)

$$\theta(1) = f(x_k + p_k) \quad (1.27)$$

So if $f(x_k + p_k)$ does not satisfy (1.24), that is, $\theta(1) > \theta(0) + \lambda\dot{\theta}(0)$, then we model $\theta(\alpha)$ by one dimensional quadratic model satisfying (1.26) and (1.27), that is

$\hat{m}(\alpha) = \left[\theta(1) - \theta(0) - \dot{\theta}(0)\right]\alpha^2 + \dot{\theta}(0)\alpha + \dot{\theta}(0)$. A nd c alculate t he poi nt, $\bar{\alpha} = \frac{\dot{\theta}(0)}{2[\theta(1) - \theta(0) - \dot{\theta}(0)]} > 0$ for which $\dot{\hat{m}}(\alpha) = 0$.

Now $\ddot{\hat{m}}(\bar{\alpha}) > 0$.Since $\theta(1) > \theta(0) + \lambda\dot{\theta}(0) > \theta(0) + \dot{\theta}(0)$. Thus $\bar{\alpha}$ minimizes t he model function. Furthermore $\bar{\alpha} > 0$ because $\dot{\theta}(0) < 0$, therefore we take $\bar{\alpha}$ as our new value of α_k in our backtracking a lgorithm. We not e t hat s ince $\theta(1) > \theta(0) + \lambda\dot{\theta}(0)$, we have

$\bar{\alpha} < \frac{1}{2(1-\lambda)}$. In fact $\theta(1) \geq \theta(0)$, then $\bar{\alpha} \leq \frac{1}{2}$ this gives an upper bound of $u = \frac{1}{2}$ on the first value of ρ in the algorithm. On the other hand i f $\theta(1)$ is much larger than $\theta(0), \bar{\alpha}$ can be very small but we impose a lower bound of $l = \frac{1}{10}$ in the algorithm, that is at the first b acktrack if $\bar{\alpha} \leq 0.1$ then w e take $\alpha_k = \frac{1}{10}$. we can u se b acktracking as t he f irst iteration us ing qua dratic m odel i f $\theta(\alpha_k) = f(x_k + \alpha_k p_k)$doesnot s atisfy (1.24) in th is case we need to backtrack again.

Although w e w ould us e qua dratic model a s i n t he f irst b acktrack w e now ha ve f our information on θ which are $\theta(0), \dot{\theta}(0)$, and the two values of $\theta(\alpha)$. so at this and any subsequent backtrack d uring t he c urrent iteration, w e us e t he c ubic model of θ, the calculation of α proceed as follows;

Let α_p, α_{2p} be t he l ast t wo pr evious va lues of α_k then c ubic m odel t hat s atisfies $\theta(0), \dot{\theta}(0), \theta(\alpha_p), \theta(\alpha_{2p})$ $is\ \widehat{M}_{cub}(\alpha) = a\alpha^3 + b\alpha^2 + \dot{\theta}(0)\alpha + \theta(0)$

$$\binom{a}{b} = \frac{1}{\alpha_p - \alpha_{2p}} \begin{bmatrix} \frac{1}{{\alpha_{pr}}^2} - \frac{1}{{\alpha_{2p}}^2} \\ \frac{-\alpha_{2p}}{{\alpha_p}^2} - \frac{\alpha_p}{{\alpha_p}^2} \end{bmatrix} \begin{pmatrix} \theta(\alpha_p) - \theta(0) - \theta(0)\alpha_p \\ \theta(\alpha_{2p}) - \theta(0) - \dot{\theta}(0)\alpha_{2p} \end{pmatrix} =$$

$$\frac{1}{\alpha_p - \alpha_{2p}} \begin{bmatrix} \frac{1}{{\alpha_p}^2} - \frac{1}{{\alpha_{2p}}^2} \\ -\frac{\alpha_{2p}}{{\alpha_p}^2} - \frac{\alpha_p}{{\alpha_{2p}}^2} \end{bmatrix} \begin{pmatrix} \theta(\alpha_p) - \theta(0) - \theta(0)\alpha_p \\ \theta(\alpha_{2p}) - \theta(0) - \dot{\theta}(0)\alpha_{2p} \end{pmatrix}$$

Its local minimizing point is $\bar{\alpha} = \frac{-b + \sqrt{b^2 - 3a\dot{\theta}(0)}}{3a}$. It can be shown that $\bar{\alpha}$ is always real if $\alpha < \frac{1}{4}$. F inally w e us e upper bound f or $u = \frac{1}{2}$ and a l ower bound $l = \frac{1}{10}$ for $\bar{\alpha}$ this

means that if $\bar{\alpha} > \frac{1}{2}\alpha_p$, we set a new $\alpha_k = \frac{1}{2}\alpha_p$ and if $\bar{\alpha} < \frac{1}{10}\alpha_p$ then we use new $\alpha_k = \frac{1}{10}\alpha_p$.

1.3.4 Trust region methods

The concept of the trust region first appears in a paper of Liebenberg (1944) and Marquart (1963) for solving nonlinear least square problems. Trust region methods are iterative numerical procedures like the line search methods in which an approximation of the objective function $f(x)$ by a model $m_k(p)$ is computed in a neighborhood of the current iterate x_k, which we refer to as the trust region. The model $m_k(p)$ should be constructed so that it is easier to handle than $f(x)$ itself. Let us assume for this that our function f is of class C^2[7].

We solve the following subproblem to obtain the next iteration at each step k of a trust region method. $(QP)_{tr}^{k}$ $\begin{cases} \min_p m_k(p) \coloneqq f(x_k) + \nabla f(x_k)^{\mathrm{T}}\mathrm{p} + \frac{1}{2}\mathrm{p}^{\mathrm{T}}\nabla^2\mathrm{f}(x_k)\mathrm{p} \\ subject\ to\ \|p\| \leq \Delta_k \end{cases}$

Where, $\Delta_k > 0$ is the trust region radius, and $\|.\|$ is defined to be the Euclidean norm. These sub problems are constrained optimization problems in which the objective function and the constraints are both quadratic. The constraint is a quadratic inequality constraint and can be written as $-p^T p + \Delta_k^2 \geq 0$. In fact usually, the model $m_k(p)$ is a quadratic function which is truncated from a Taylor series for f around the point x_k:

$$m_k(p) = f(x_k) + \nabla f(x_k)^{\mathrm{T}}\mathrm{p} + \frac{1}{2}\mathrm{p}^{\mathrm{T}}\nabla^2\mathrm{f}(x_k)\mathrm{p} \quad (\mathrm{k} \in \mathrm{N}_0)$$

Where $\mathrm{N}_0 = \{0,1,2,\ldots\}$

We note that one can choose any other norm in the formulations. In this project we consider the Euclidean norm $\|.\| = \|.\|_2$ since it makes some computations easier. Hence our trust region for the model $m_k(p)$ is a bounded neighborhood of the current iterate x_k

$$\beta(x_k) \coloneqq \{x_k + p | \|p\|_2 \leq \Delta_k\}.$$

After constructing the model $m_k(p)$ and its trust region then one seeks a trial step p to the next iteration $x_k = x_k + p$ which will result in a reduction for the model while the size of the step is bounded by $\beta(x_k)$ that is $\|p\|_2 \leq \Delta_k$.Then, the objective function is evaluated at $x_k + p$ to compare its value to the one predicted by the model at this point. If the sufficient reduction predicted by the model is accomplished by the objective function, $x_k + p$ is accepted as the next iterate and the trust region is possibly expanded to include this new point (Δ_k increase). If the reduction in the model is a poor predictor of the actual reduction of the objective function, then the trial point is rejected. We conclude that the region is too large and the size of the trust region is reduced(Δ_k decreases), with the hope that the model provides a better prediction in the smaller region.

1.3.4.1 Outline and properties of the trust region algorithm

In a trust region algorithm, a strategy for determining the trust region radius, Δ_k, at each, iteration is needed to be developed. The trust region radius can be determined by looking at the agreement between the model function m_k and the objective function f at previous iterations. Given a step p_k we define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(x_k) - m_k(x_k + p_k)} = \frac{actual\ reduction}{(predicted\ reduction)} \qquad (1.28)$$

There are various definitions for ρ_k in the mathematical literature but we shall prefer the above in particular here. We note that the denominator of ρ_k namely the predicted reduction is always non negative since the step p_k is computed from the subproblem $(QP)^k_{t.r}$ over a region that includes the step $p = 0$. In fact ρ_k can be called a measure of how good the model $m_k(p)$ predicts the reduction in the f value. If ρ_k is closer to 0 or negative, then the actual prediction is smaller than the predicted one. This indicates that the model cannot be trusted in this region with radius Δ_k. Thus p_k will be rejected and Δ_k will be reduced. On the other hand if ρ_k is close to one, an adequate prediction is obtained we safely expand the trust region since the model can be trusted over a wider region that should be increased. If ρ_k is positive but not close to 1, then the trust region radius is not changed.

Algorithm(Trust-Region Algorithm)

The steps in derivative based trust region methods can be summarized by the following [2].

1. Specify some initial guess of the solution x_0. Select the initial trust-region bound $\Delta_0 > 0$. Specify the constants $0 < \mu < \eta < 1$ (perhaps $\mu = ¼$ and $\eta = ¾$).

2. For $k = 0, 1, \ldots$

(i) if x_k is optimal, stop.

(ii)solve

$$\min_p q_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p$$

$$subject\ to\ \|p\| \leq \Delta_k$$

for the trial step p_k

(iii) Compute

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{f(x_k) - q_k(p_k)} = \frac{actual\ \ reduction}{predicted\ \ reduction}.$$

(iv) $If\ \rho_k \leq \mu, then\ x_{k+1} = x_k$ (unsuccessful step), else $x_{k+1} = x_k + p_k$ (successful step).

(v) Update Δ_k:

$$\rho_k \leq \mu \implies \Delta_{k+1} = \frac{1}{2}\Delta_k$$

$$\mu < \rho_k < \eta \implies \Delta_{k+1} = \Delta_k$$

$$\rho_k \geq \eta \implies \Delta_{k+1} = 2\Delta_k.$$

The value of ρ_k indicates how well the model predicts the reduction in the function value. If ρ_k is small (that is, $\rho_k \leq \mu$), then the actual reduction in the function value is much smaller than that predicted by $q_k(\rho_k)$, indicating that the model cannot be trusted for a bound as large as Δ_k; in this case the step ρ_k will be rejected and Δ_k will be reduced. If ρ_k is large (that is, $\rho_k \geq \eta$), then the model is adequately predicting the reduction in the function value, suggesting that the model can be trusted over an even wider region; in this case the bound Δ_k will be increased.

.

Chapter 2
Derivative Free Optimization Methods

Introduction (What is derivative free optimization)

Derivative Free Optimization (DFO) methods are typically designed to solve optimization problems whose objective function is computed by a "black box"; hence, the gradient computation is unavailable. Each call to the "black box" is often expensive, so estimating derivatives by finite differences may be prohibitively costly. Finally, the objective function value may be computed with some noise, and the finite differences estimates may not be accurate [6].

The derivative free optimization method which we use approximates the objective function explicitly without approximating its derivatives. The theoretical analysis presented in this project assumes that no noise is present. Expensive experiments and intuition support the claim that robustness of DFO doesn't suffer from presence of modern level of noise [1].

Derivative free optimization has been developed for solving optimization problems of the following form

$$\min f(x)$$

$$(P) \qquad \text{Such that } a_i \leq g_i(x) \leq b_i \quad (i = 1, 2 \dots, m)$$

$$x \in F \subseteq \mathbb{R}^n,$$

Where the objective function $f(x)$ and the constraints $g_i(x)$ are expensive to compute at a given vector x and the derivatives of f at x are not available.

The idea is to approximate the objective function by a model which is assumed to describe the objective function well in a trust region without explicitly modeling its derivatives. This model is computationally less expensive to evaluate and easier to optimize than the objective function itself. The model is obtained by interpolating the objective function using a quadratic interpolation polynomial.

Derivative free optimization methods for unconstrained optimization build a linear or quadratic model of the objective function and apply one of the fundamental approaches of continuous optimization i.e. a trust region or a line search, to optimize this model. While derivative based methods typically use a Taylor-based model which is an approximation of the objective function, derivative free methods use interpolation, regression or other sample-based models. If the problem has constraints, the strategy of derivative free methods is usually to apply sequential quadratic programming methods for the linearization of the constraints [6]. The main concern of this project is on unconstrained nonlinear programming problems without using derivatives.

We now consider two basic methods in DFO: line search and trust region methods without using derivatives.

2.2 line search methods

Line search, also called one-dimensional search, refers to an optimization procedure for solving nonlinear Univariant problems. One dimensional line search is the back bone of many algorithms for solving non linear programming problems. Many non linear programming algorithms proceed as follows. Given a point x_k , find a direction vector p_k and then a suitable step size α_k , yielding a new point, $x_{k+1} = x_k + \alpha_k p_k$; the process is then repeated.

Finding the new step size α_k involves solving the sub problem to minimize $f(x_k + \alpha k p k$ which is a one dimensional search problem in the variable α. The minimization is over all real α , a non negative α, or α such that $x_k + \alpha d_k$ is feasible.

As stated before, in multivariable optimization algorithms, for given x_k, the iterative scheme$is x_{k+1} = x_k + \alpha_k p_k$. (2.1)

The key is to find the direction vector p_k and a suitable step size α_k. Let

$$\phi(\alpha) = f(x_k + \alpha d_k). \quad (2.2)$$

So, the problem that departs from x_k and finds a step size in the direction p_k such that

$$\phi(\alpha_k) < \phi(0)$$

is just line search about α.

If we find α_k such that the objective function in the direction p_k is minimized,

i.e.,

$$f(x_k + \alpha_k p_k) = \min_{\alpha>0} f(x_k + \alpha p_k)$$

or

$$\phi(\alpha_k) = \min_{\alpha>0} \phi(\alpha),$$

such a line search is called exact line search or optimal line search, and α_k is called optimal step size. If we choose α_k such that the objective function has acceptable descent amount, i.e., such that the descent $f(x_k) - f(x_k + \alpha_k p_k) > 0$is acceptable by users, such a line search is called inexact line search, or approximate line search, or acceptable line search.

Since, in practical computation, theoretically exact optimal step size generally cannot be found, and it is also expensive to find almost exact step size, therefore the inexact line search with less computation load is highly popular.

The framework of line search is as follows. First, determine or give an initial search interval which contains the minimizer; then employ some section techniques or interpolations to reduce the interval iteratively until the length of the interval is less than some given tolerance [10].

Next, we give a notation about the search interval and a simple method to determine the initial search interval.

Interval of uncertainty

Definition 2.1. *Let* $\phi : \mathbb{R} \to \mathbb{R}, \alpha^* \in [0, +\infty), and$

$$\phi(\alpha^*) = \min_{\alpha\geq 0} \phi(\alpha)$$

I*f* there exists a closed interval $[a, b] \subset [0, +\infty)$ *such that* $\alpha^* \in [a, b]$, *then* $[a, b]$ *is called a search interval for one-dimensional minimization problem* $\min_{\alpha\geq 0} \phi(\alpha)$. *Since the exact location of the minimum of* ϕ *over* $[a, b]$ *is not known, this interval is also called the interval of uncertainty.*

A simple method to determine an initial interval is called the forward-backward method. The basic idea of this method is as follows. Given an initial point and an initial step length, we attempt to determine three points at which their function values show "high–

low–high" ge ometry. I f i t i s not s uccessful t o go f orward, w e w ill go ba ckward. Concretely, given an initial point, α_0, and a step length, $h_0 > 0$. If

$$\phi(\alpha_0 + h_0) < \phi(\alpha_0),$$

then, next step, depart from $\alpha_0 + h_0$ and continue going forward with a larger step until the objective function increases. If

$$\phi(\alpha_0 + h_0) > \phi(\alpha_0),$$

then, next step, depart fromα_0 and go backward until the objective function increases. So, we will obtain an initial interval which contains the minimum α^*.

Algorithm 2.1.2 *(Forward-Backward Method)*

Step 1. Given$\alpha_0 \in [0, \infty), h_0 > 0$*, the multiple coefficient t* > 1

(Usually t = 2*). Evaluate*$\phi(\alpha_0), k := 0$.

Step 2. Compare the objective function values. Set $\alpha_{k+1} = \alpha_k + h_k$ *and evaluate*$\phi_{k+1} = \phi(\alpha_{k+1})$. *If*$\phi_{k+1} < \phi_k$*, go to Step 3; otherwise, go to Step 4.*

Step 3. F orward s tep. Se t $h_{k+1} := th_k, \alpha := \alpha_k, \alpha k := \alpha_{k+1}, \phi_k := \phi_{k+1}, k := k + 1$*, go to Step 2.*

Step 4. Backward step. If k = 0*, invert the search direction. Set*

$h_k := -h_k, \alpha k := \alpha k + 1$*, go to Step 2; otherwise, set*

$$a = min\{\alpha, \alpha_{k+1}\}, b = max\{\alpha, \alpha_{k+1}\},$$

Output [a, b] and stop.

The methods of line search presented in this chapter use the unimodality of the function and interval. T he f ollowing de finitions and t heorem i ntroduce their c oncepts a nd properties.

Definition 2.2 *Let*$\phi : \mathbb{R} \to \mathbb{R}, [a, b] \subset \mathbb{R}$*. If there is* $\alpha^* \in [a, b]$ *such that* $\phi(\alpha)$ *is strictly decreasing on* $[a, \alpha^*]$ *and strictly increasing on* $[\alpha^*, b]$*, then* $\phi(\alpha)$ *is called a* unimodal *function on* $[a, b]$*. Such an interval* $[a, b]$ *is called a unimodal interval related to* $\phi(\alpha)$.

The unimodal function can also be defined as follows.

Definition 2.3 *If there exists a unique* $\alpha^* \in [a, b]$*, such that for any* $\alpha_1, \alpha_2 \in [a, b], \alpha_1 < \alpha_2$*, th e fo llowing s tatements h old: if* $\alpha_2 < \alpha^*$*, t hen* $\phi(\alpha_1) > \phi(\alpha_2)$*; if* $\alpha_1 > \alpha^*$*, then* $\phi(\alpha_1) < \phi(\alpha_2)$*; then* $\phi(\alpha)$ *is the unimodal function on* $[a, b]$.

Note that, first, the unimodal function does not require the continuity and differentiability of the function; second, using the property of the unimodal function, we can exclude portions of the interval of uncertainty that do not contain the minimum, such that the interval of uncertainty is reduced. The following theorem shows that if the function ϕ is unimodal on $[a, b]$, then the interval of uncertainty could be reduced by comparing the function values of ϕ at two points within the interval.

Theorem 2.1 *Let* $\phi : \mathbb{R} \to \mathbb{R}$ *be unimodal on* $[a, b]$. *Let* $\alpha_1, \alpha_2 \in [a, b]$, *and* $\alpha_1 < \alpha_2$. *Then*

1. if $\phi(\alpha_1) \leq \phi(\alpha_2)$, *then* $[a, \alpha_2]$ *is a unimodal interval related to* ϕ;

2. if $\phi(\alpha_1) \geq \phi(\alpha_2)$, *then* $[\alpha_1, b]$ *is a unimodal interval related to* ϕ.

Proof. From the Definition 2.2, there exists $\alpha^* \in [a, b]$ such that $\phi(\alpha)$ is strictly decreasing over $[a, \alpha^*]$ and strictly increasing over $[\alpha^*, b]$. since $\phi(\alpha_1) \leq \phi(\alpha_2)$, then $\alpha^* \in [a, \alpha_2]$.Since $\phi(\alpha)$ is unimodal on $[a, b]$, it is also unimodal on $[a, \alpha_2]$. Therefore $[a, \alpha_2]$ is a unimodal interval related to $\phi(\alpha)$ and the proof of the first part is complete.
The second part of the theorem can be proved similarly.
This theorem indicates that, for reducing the interval of uncertainty, we must at least select two observations, evaluate and compare their function values.

Theorem 2.2 Let $\phi: \mathbb{R} \to \mathbb{R}$ be a strictly quassi-convex over the interval $[a, b]$. Let $\alpha, \mu \in [a, b]$ be such that $\alpha < \mu$.then;

If $\phi(\alpha) > \phi(\mu)$, then $\phi(z) \geq \phi(\mu)$, for all $z \in [a, \alpha)$. If $\phi(\alpha) \leq \phi(\mu)$, then $\phi(z) \geq \phi(\alpha)$, for all $z \in [\mu, b)$.

Proof: suppose that $\phi(\alpha) > \phi(\mu)$, and let $z \in [a, \alpha)$.
By contradiction, suppose that $\phi(z) < \phi(\mu)$, since α can be written as a convex combination of z and μ , and by the strict quasi-convexity of ϕ, we have $\phi(\alpha) < max\{\phi(z), \phi(\mu)\} = \phi(\mu)$ contradicting $\phi(\alpha) > \phi(\mu)$. Hence, $\phi(z) \geq \phi(\mu)$. The second part of the Theorem can be proved similarly.

FromTheorem.2.2, under strict quassi-convexity, if $\phi(\alpha) > \phi(\mu)$, the new interval of uncertainty is $[\alpha, b]$, on the other hand, if $\phi(\alpha) \leq \phi(\mu)$, the new interval of uncertainty is $[a, \mu]$.

The Golden Section Method and the Fibonacci Method

The golden section method and the Fibonacci method are section methods.

Their basic idea for minimizing a unimodal function over $[a, b]$ is iteratively reducing the interval of uncertainty by comparing the function values of the observations. When the length of the interval of uncertainty is reduced to some desired degree, the points on the interval can be regarded as approximations of the minimizer. Such a class of methods only needs to evaluate the functions and has wide applications; especially it is suitable to nonsmooth problems and those problems with complicated derivative expressions.

2.2.1.1 The golden section method

We now describe the more efficient golden section method for minimizing a strictly quassi-convex function. At a general iteration k of the golden section method, let the interval of uncertainty be $[a_k, b_k]$ then by the above theorem the new interval of uncertainty $[a_{k+1}, b_{k+1}]$ is given by $[\alpha_k, b_k]$ if $\phi(\alpha_k) > \phi(\mu_k)$ and by $[a_k, \mu_k]$ if $\phi(\alpha_k) \leq \phi(\mu_k)$. The points α_k and μ_k are selected such that;

1) The length of the new interval of uncertainty $b_{k+1} - a_{k+1}$ does not depend up on the out come of k^{th} iteration, that is, on whether $\phi(\alpha_k) > \phi(\mu_k)$ or $\phi(\alpha_k) \leq \phi(\mu_k)$. Therefore we must have $b_k - \alpha_k = \mu_k - a_k$. Thus, if α_k is of the form

$$\alpha_k = a_k + (1 - \lambda)(b_k - a_k) \qquad (*)$$

$where\ \lambda \in (0,1)$.Then μ_k must be of the form

$$\mu_k = a_k + \lambda(b_k - a_k) \qquad (**)$$

So that $b_{k+1} - a_{k+1} = \lambda(b_k - a_k)$

2) As μ_{k+1} and α_{k+1} are selected for the purpose of a new iteration, either α_{k+1} coincides with μ_k or μ_{k+1} coincides with α_k. If this can be realized, then during iteration $k + 1$, only one extra observation is needed. Consider the following two cases

Case 1 : $\phi(\alpha_k) > \phi(\mu_k)$ in this case $a_{k+1} = \alpha_k \; and \; b_{k+1} = b_k$. To satisfy $\alpha_{k+1} = \mu_k$, and applying $(*)$ with k replaced by $k+1$, we get

$$\mu_k = \alpha_{k+1} = a_{k+1} + (1-\lambda)(b_{k+1} - a_{k+1}) = \alpha_k + (1-\lambda)(b_k - \alpha_k)$$

Substituting the expressions of α_k and μ_k from $(*) and \; (**)$ into the above equation we get $\lambda^2 + \lambda - 1 = 0$.

Case 2: $\phi(\alpha_k) \leq \phi(\mu_k)$

In this case $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$ to satisfy $\mu_{k+1} = \alpha_k$ and apply $(**)$ with k replaced by $k+1$. we get

$$\alpha_k = \mu_{k+1} = a_{k+1} + \lambda(b_{k+1} - a_{k+1})$$

$$= a_k + \lambda(\mu_k - a_k).$$

Noting (*) and (**), the above equation gives $\lambda^2 + \lambda - 1 = 0$.

The roots of the equation $\lambda^2 + \lambda - 1 = 0$. are $\lambda \cong 0.618 \; and \; \lambda \cong -1.618$. since λ must be in the interval $(0,1)$ then $\lambda \cong 0.618$.

To summarize, if at iteration k, μ_k and α_k are chosen according to (*) and (**), where$\lambda = 0.618$, then the interval of uncertainty is reduced by a factor of 0.618. At the first iteration, two observations are needed at α_1 and μ_1, but at each subsequent iteration, only one evaluations is needed, since either $\alpha_{k+1} = \mu_k$ or $\mu_{k+1} = \alpha_k$.

Algorithm *(the golden section method)*

The following is a summary of the golden section method for minimizing the strictly quassi-convex function over the interval $[a, b]$ [8].

Initialization step: choose an allowable final length of uncertainty $\ell > 0$.let $[a_1, b_1]$ be the interval of uncertainty, and let $\alpha_1 = a_1 + (1-\lambda)(b_1 - a_1)$ and

$\mu_1 = a_1 + \lambda(b_1 - a_1)$ for $\alpha = 0.618$. Evaluate $\phi(\alpha_1) and \phi(\mu_1)$. Let $k = 1$, and go to the main step.

step1: if $b_k - \alpha_k < l, stop$: the optimal solution lies in the interval $[a_k, b_k]$. Otherwise, if $\phi(\alpha_k) > \phi(\mu_k$ go to step 2; and if $\phi(\alpha_k) \leq \phi(\mu_k)$, go to step 3.

Step 2: let $a_{k+1} = \alpha_k$ and $b_{k+1} = b_k$. Furthermore, let$\alpha_{k+1} = \mu_k$, and let $\mu_{k+1} = a_{k+1} + \lambda(b_{k+1} - a_{k+1})$. Evaluate $\phi(\mu_{k+1})$ and go to step 4.

Step3: let $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$. Furthermore, let $\mu_{k+1} = \alpha_k \; and$

$let\alpha_{k+1} = \; a_{k+1} + \alpha_k + (1 - \lambda)(b_{k+1} - a_{k+1})$. Evaluate $\phi(\alpha_{k+1})$ and go to step 4.

Step4: replace k by $k + 1$ and go to step 1.

Example2.1: consider the following problem

$$minimize x^2 + 2x \;\; subject\ to - 3 \leq x \leq 5$$

Note that the true minimum is -1.0

Table2.1; Summary of computations for the above problem using golden section method

Iteration k	a_k	b_k	α_k	μ_k	$\phi(\alpha_k)$	$\phi(\mu_k)$
1	-3.000	5.000	0.056	1.944	0.115*	7.667*
2	-3.000	1.944	-1.112	0.056	-1.987*	0.115
3	-3.000	0.056	-1.832	-1.112	-0.308 *	-0.987
4	-1.832	0.056	-1.112	-0.664	-0.987	-0.887 *
5	-1.832	-0.664	-1.384	-1.112	-0.853 *	-0.987
6	-1.384	-0.664	-1.112	-0.936	-0.987	-0.996*
7	-1.112	-0.664	-0.936	-0.840	-0.996	-0.974*
8	-1.112	-0.840	-1.016	-0.936	-1.000*	-0.996

Clearly the function ϕ to be minimized is strictly quassi-convex, and the initial interval of uncertainty is of length 8. We reduce this interval of uncertainty to one whose length is at most 0.2. The first two observations are located at

$$\alpha_1 = -3 + 0.382(8) = 0.056$$

$$\mu_1 = -3 + 0.618(8) = 1.944$$

Note that $\phi(\alpha_1) \leq \phi(\mu_1)$. Hence the new interval of uncertainty is $[-3, 1.944]$. The process is repeated, and the computations are summarized in the above table. The values of ϕ that are computed at each iteration are indicated by an asterisk.

After eight iterations involving nine observations, the interval of uncertainty is $[-1.112, -0.936]$, so that the minimum can be estimated to be the midpoint -1.024.

The numerical result for example 2.1 is -0.9998 (see appendix for the matlab code)

2.2.1.2 The Fibonacci search

The Fibonacci search is a line search procedure for minimizing strictly quassi-convex function ϕ over a closed bounded interval. Similar to the golden section method the Fibonacci search procedure makes two functional evaluations at the first iteration and then only one evaluation at each of the subsequent iterations. However, the procedure differs from golden section method in the reduction of interval of uncertainty varies from one iteration to another [8].

The procedure is based on the Fibonacci sequence $\{F_v\}$ defined as follows;

$$F_{v+1} = F_v + F_{v-1} \qquad v = 1,2 \ldots \qquad (2.3)$$

$$F_0 = F_1 = 1$$

The sequence is therefore $1,1,2,5,8,13,21,34,55,89,144,233, \ldots$. at the iteration k suppose that the interval of uncertainty is $[a_k, b_k]$. Consider the two points α_k and μ_k given below, where n is the total number of functional evaluations planned.

$$\alpha_k = a_k + \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k), \qquad k = 1,2,\dots,n-1 \qquad (2.4)$$

$$\mu_k = a_k + \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k), \quad k = 1,2,\dots,n-1 \qquad (2.5)$$

By theorem(2.2), the new interval of uncertainty $[a_{k+1}, b_{k+1}]$ is given by

$[\alpha_k, b_k] if \phi(\alpha_k) > \phi(\mu_k)$ and is given by$[a_k, \mu_k] if \phi(\alpha_k) \leq \phi(\mu_k)$.

In the former case, nothing (2.4) and letting $v = n - k$ in(2.3), we get

$$b_{k+1} - a_{k+1} = b_k - \alpha_k$$

$$= b_k - a_k - \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k) \qquad (2.6)$$

$$= \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k)$$

In the later case, nothing (2.5) we get $b_{k+1} - a_{k+1} = \mu_k - a_k = \frac{F_{n-k}}{F_{n-k+1}}(b_k -$

ak *(2.7)*

Thus, in either case the interval of uncertainty is reduced by the factor $\frac{F_{n-k}}{F_{n-k+1}}$.We now show that at iteration k+1, either $\alpha_{k+1} = \mu_k \; or \; \mu_{k+1} = \alpha_k$, so that only one functional evaluation is needed. Suppose that $\phi(\lambda_k) > \phi(\mu_k$ then by theorem(2.2),$a_{k+1} = \alpha_k, and \; a_{k+1} = b_k$.

Thus, applying (2.4) with k replaced by k+1, we get

$$\alpha_{k+1} = a_{k+1} + \frac{F_{n-k-2}}{F_{n-k}}(b_{k+1} - a_{k+1})$$

$$= \alpha_k + \frac{F_{n-k-2}}{F_{n-k}}(b_k - \lambda_k)$$

Substituting for α_k from (2.4) we get

$$\alpha_{k+1} = a_k + \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k) + \frac{F_{n-k-2}}{F_{n-k}}\left(1 - \frac{F_{n-k-1}}{F_{n-k+1}}\right)((b_k - a_k))$$

Letting $v = n - k$ in (2.3), it follows that $1 - \frac{F_{n-k-1}}{F_{n-k+1}} = \frac{F_{n-k}}{F_{n-k+1}}$.

$$\alpha_{k+1} = a_k + (\frac{F_{n-k-1} + F_{n-k-2}}{F_{n-k+1}})(b_k - a_k)$$

Now l et $v = n - k - 1$ in (2.3), and not hing (2.5), it f ollows th at $\alpha_{k+1} = a_k + \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k) = \mu_k$.Similarly, if $\phi(\alpha_k) \leq \phi(\mu_k)$, and then we can easily verify that

$\mu_{k+1} = \alpha_k$. Thus, in either case, only one observation is needed at iteration $k + 1$.

To summarize, at the first iteration, two observations are made, and at each, subsequent iteration, only one observation is necessary. Thus, at the end of iteration $n - 2$, we have computed $n - 1$ functional evaluations. Further, $for\ k = n - 1$ it follows from (2.4) and (2.5) that

$\alpha_{n-1} = \mu_{n-1} = \frac{1}{2}(a_{n-1} + b_{n-1})$. Since either $\alpha_{n-1} = \mu_{n-2}$ or $\mu_{n-1} = \alpha_{n-2}$, theoretically no new observations are to be made at this stage. However in order to further reduce the interval of uncertainty; the last observation is placed slightly to the right or to the left of the middle point $\alpha_{n-1} = \mu_{n-1}$ so that $\frac{1}{2}(b_{n-1} - a_{n-1})$ is the length of the final uncertainty$[a_n, b_n]$.

Algorithm (Fibonacci search method)

The f ollowing i s a s ummary o f t he F ibonacci search f or minimizing a qua ssi-convex function over the interval$[a_1, b_1]$.

Initialization s tep: ch ose an al lowable f inal l ength o f u n c ertainty $\ell > 0$ and a distinguishability c onstant $\varepsilon > 0$. let $[a_1, b_1]$ be t he in itial interval o f u ncertainty, a nd choose the number of observations n to to be taken such that $F_n > \frac{(b_1 - a_1)}{\ell}$. Let $\alpha_1 = a_1 + \frac{F_{n-2}}{F_n}(b_1 - a_1)$ and

$\mu_1 = a_1 + \frac{F_{n-1}}{F_n}(b_1 - a_1)$. Evaluate $\phi(\lambda_1)$ and $\phi(\mu_1)$, let k=1, and go to the main step

Main steps

1) If $\phi(\alpha_k) > \phi(\mu_k)$, go to step2: and if $\phi(\lambda_k) \leq \phi(\mu_k)$, go to step 3
2) Let $a_{k+1} = \alpha_k$ and $b_{k+1} = b_k$. Furthermore, let $\alpha_{k+1} = \mu_k$, and let
   $\mu_{k+1} = a_{k+1} + \left(\frac{F_{n-k-1}}{F_{n-k}}\right)(b_{k+1} - a_{k+1})$.
   If $k = n-2$, go to step5; otherwise evaluate $\phi(\mu_{k+1})$ and go to step 4.
3) Let $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$. Furthermore, let $\mu_{k+1} = \alpha_k$, and let
   $\alpha_{k+1} = a_{k+1} + \left(\frac{F_{n-k-2}}{F_{n-k}}\right)(b_{k+1} - a_{k+1})$. If $k = n-2$: go to step 5: otherwise evaluate $\phi(\alpha_{k+1})$ and go to step 4
4) Replace k by $k+1$ and go to step 1
5) Let$\alpha_n = \alpha_{n-1}$, and $\mu_n = \alpha_{n-1} + \varepsilon$. If $\phi(\alpha_n) > \phi(\mu_n)$, let $a_n = \alpha_n$ and $b_n = b_{n-1}$. Otherwise, if $\phi(\alpha_n) \leq \phi(\mu_n)$, let $a_n = a_{n-1}$ and $b_n = \alpha_n$. Stop; the optimal solution lies in the interval $[a_n, b_n]$

Example2.2: consider the following problem

$$minimize x^2 + 2x \ \ subject\ to - 3 \leq x \leq 5$$

Note that the function is strictly quassi-convex on the interval and that the true minimum occurs at $x = -1$. We reduce the interval of uncertainty to one whose length is at most 0.2. Hence we must have $F_n > \frac{8}{0.2} = 40$. So that $n = 9$.we adopt the distinguishabilty constant $\varepsilon = 0.01$

The first two observations are located at

$$\alpha_1 = -3 + \frac{F_7}{F_9}(8) = 0.054545,\ \mu_1 = -3 + \frac{F_8}{F_9}(8) = 1.9454554\ .$$

Note that $\phi(\alpha_1) < \phi(\mu_1)$. Hence, the interval of uncertainty is $[-3.000000, 1.945454]$

Table2.2; Summary of computations for the Fibonacci search method

Iteration k	a_k	b_k	α_k	μ_k	$\phi(\alpha_k)$	$\phi(\mu_1)$
1	-3.000000	5.000000	0.054545	1.945454	0.1120*	7.67569 *
2	-3.000000	1.945454	-1.109091	0.05454	0.9886*	0.112065
3	-3.000000	0.054545	-1.836363	-1.109091	0.30049*	-0.988099
4	-1.836363	0.054545	-1.109091	-0.67727	0.98809	-0.89289*
5	-1.836363	-0.672727	-1.399999	-1.109091	0.8400*	-0.988099
6	-1.399999	-0.672727	-1.109091	-0.963636	0.9889	-0.99867*
7	-1.109091	-0.672727	-0.963636	-0.818182	0.9986	-0.96694*
8	-1.109091	-0.818182	-0.963636	-0.963636	0.9986	-0.99867
9	-1.109091	-0.963636	-0.963636	-0.953636	0.9986*	-0.99785*

The process is repeated, and the computations are summarized in the above table. The values of ϕ that are computed at each iteration are indicated by an asterisk. Note that at $k = 8$, $\alpha_k = \mu_k = \alpha_{k-1}$, so that no functional evaluations are needed at this stage. For $k = 9$, $\alpha_k = \alpha_{k-1} = -0.963636$ and $\mu_k = \alpha_k + \varepsilon = -0.953636$. Since, $\phi(\mu_k) > \phi(\alpha_k)$, the final interval of uncertainty $[a_9, b_9]$ is $[-1.109091, -0.963636]$, whose length ℓ is 0.145455. We approximate the minimum to be the midpoint -1.036364. Note from example 2.1 that with the same number of observations $n = 9$, the golden section method gave a final interval of uncertainty whose length is 0.176.

The Fibonacci method has the following limitations [9]

1. The initial interval of uncertainty, in which the optimum lies, has to be known.

2. The function being optimized has to be unimodal in the initial interval of uncertainty.

3. The exact optimum cannot be located in this method. Only an interval known as the *final* interval *of uncertainty* will be known. The final interval of uncertainty can be made as small as desired by using more computations.

4. The number of function evaluations to be used in the search or the resolution required has to be specified beforehand.

2.2.3Interpolation Methods

The interpolation methods were originally developed as one-dimensional searches within multivariable optimization techniques, and are generally more efficient than Fibonacci-type approaches. The aim of all the one-dimensional minimization methods is to find λ^*, the smallest nonnegative value of λ, for which the function

$$f(\lambda) = f(X + \lambda S) \qquad (2.8)$$

attains a local minimum. Hence if the original function f (X) is expressible as an explicit function of x_i $(i = 1, 2, \ldots, n)$, we can readily write the expression for $f(\lambda) = f(\mathrm{X} + \lambda \mathrm{S})$ for any specified vector S, set

$$\frac{df}{d\lambda}(\lambda) = 0 \qquad (2.9)$$

and solve Eq. (2.9) to find λ^* in terms of X and S. However, in many practical problems, the function $f(\lambda)$ cannot be expressed explicitly in terms of λ. In such cases the interpolation methods can be used to find the value of λ^*.

Example 2.3 Derive the one-dimensional minimization problem for the following case:

Minimize $f(X) = \left(x_1^2 - x_2\right)^2 + \left(1 - x_1\right)^2 \qquad (E1)$

from the starting point $X_1 = \begin{Bmatrix} -2 \\ -2 \end{Bmatrix}$ along the search direction $S = \begin{Bmatrix} 1.00 \\ 0.25 \end{Bmatrix}$.

SOLUTION the new design point X can be expressed as

$$X = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = X_1 + \lambda \mathrm{S} = \begin{Bmatrix} -2 + \lambda \\ -2 + 0.25\lambda \end{Bmatrix}$$

By substituting $x_1 = -2 + \lambda$ and $x_2 = -2 + 0.25\lambda$ in Eq.$(E1)$, we obtain f as a function of λ as

$$f(\lambda) = f\begin{pmatrix} -2 + \lambda \\ -2 + 0.25\lambda \end{pmatrix} = [(-2 + \lambda)^2 - (-2 + 0.25\lambda)]^2 + [1 - (-2 + \lambda)]^2$$
$$= \lambda^4 - 8.5\lambda^3 + 31.0625\lambda^2 - 57.0\lambda + 45.0$$

The value of λ at which $f(\lambda)$ attains a minimum gives λ^*.

In the following sections, we discuss different interpolation methods with reference to one-dimensional minimization problems that arise during multivariable optimization problems.

2.2.3.1Quadratic interpolation method

The quadratic interpolation method uses the function values only; hence it is useful to find the minimizing step (λ^*) of functions f (X) for which the partial derivatives with respect to the variables x_i are not available or difficult to compute.

This method finds the minimizing step length λ^* in three stages. In the first stage the S-vector is normalized so that a step length of $\lambda = 1$ is acceptable. In the second stage the function $f(\lambda)$ is approximated by a quadratic function $h(\lambda)$ and the minimum, $\tilde{\lambda}^*$, of $h(\lambda)$ is found. If $\tilde{\lambda}^*$is not sufficiently close to the true minimum λ^*a third stage is used. In this stage a new quadratic function (refit) $h'(\lambda) = a' + b'\lambda + c'\lambda^2$ is used to approximate $f(\lambda)$, and a new value of $\tilde{\lambda}^*$is found. This procedure is continued until a $\tilde{\lambda}^*$that is sufficiently close to λ^* is found.

*Stage 1***.** In this stage, the ***S*** vector is normalized as follows: Find

$$\Delta = \max_i |s_i|,$$

where s_i is the i^{th} component of $\boldsymbol{S}$ and divide each component of S by Δ. Another method of normalization is to find $\Delta = (s_1^2 + s_2^2 + \cdots + s_n^2)^{\frac{1}{2}}$ and divide each component of S by Δ.

Stage 2**.** Let $h(\lambda) = a + b\lambda + c\lambda^2$ (2.10)

be the quadratic function used for approximating the function $f(\lambda)$. It is worth noting at this point that a quadratic is the lowest-order polynomial for which a finite minimum can exist. The necessary condition for the minimum of $h(\lambda)$ is that

$$\frac{dh}{d\lambda} = b + 2c\lambda = 0.$$

that is,

$$\tilde{\lambda}^* = -\frac{b}{2c} \qquad (2.11)$$

The sufficiency condition for the minimum of $h(\lambda)$ is that

$$\left.\frac{d^2 f}{d\lambda^2}\right|_{\tilde{\lambda}^*} > 0$$

that is,

$$c > 0 \qquad (2.12)$$

To evaluate the constants $a, b, and\ c$ in Eq. (2.10), we need to evaluate the function

$f(\lambda)$ at three points. Let $\lambda = A, \lambda = B, and\ \lambda = C$ be the points at which the function $f(\lambda)$ is evaluated and let $f_A, f_B, and\ f_C$ be the corresponding function values, that is,

$$f_A = a + bA + cA^2$$
$$f_B = a + bB + cB^2$$
$$f_C = a + bC + cC^2 \qquad (2.13)$$

The solution of Eqs. (2.13) gives

$$a = \frac{\left(f_A BC(C - B) + f_B CA(A - C) + f_C AB(B - A)\right)}{(A - B)(B - C)(C - A)} \qquad (2.14)$$

$$b = \frac{\left(fA(B2 - C2) + fB(C2 - A2) + fC(A2 - B2)\right)}{(A - B)(B - C)(C - A)} \qquad (2.15)$$

$$c = -\frac{f_A(B - C) + f_B(C - A) + f_C(A - B)}{(A - B)(B - C)(C - A)} \qquad (2.16)$$

From Eqs. (2.11), (2.15), and (2.16), the minimum of $h(\lambda)$ can be obtained as

$$\tilde{\lambda}^* = -\frac{b}{2c} = \frac{f_A(B^2 - C^2) + f_B(C^2 - A^2) + (A^2 - B^2)}{2[f_A(B - C) + f_B(C - A) + f_C(A - B)]} \qquad (2.17)$$

Provided that c, as given by Eq. (2.16), is positive.

To start with, for simplicity, the points A, B, and C can be chosen as $0, t$, and $2t$, respectively, where t is a preselected trial step length. By this procedure, we can save one function evaluation since $f_A = f(\lambda = 0)$ is generally known from the previous iteration (of a multivariable search). For this case, Eqs. (2.14) to (2.17) reduce to

$$a = f_A \qquad (2.18)$$

$$b = \frac{4f_B - 3f_A - f_C}{2t} \qquad (2.19)$$

$$c = \frac{f_C + f_A - 2f_B}{2t^2} \qquad (2.20)$$

$$\tilde{\lambda}^* = \frac{(4f_B - 3f_A - f_C)}{4f_B - 2f_C - 2f_A}\, t \qquad (2.21)$$

Provided that

$$c = \frac{f_C + f_A - 2f_B}{2t^2} > 0 \qquad (2.22)$$

The inequality (2.22) can be satisfied if

$$\frac{f_A + f_C}{2} > f_B \qquad (2.23)$$

(i.e., the function value f_B should be smaller than the average value of f_A and f_C).

This can be satisfied if f_B lies below the line joining f_A and f_C.

The following procedure can be used not only to satisfy the inequality (2.23) but also to ensure that the minimum $\tilde{\lambda}^*$lies in the interval$0 < \tilde{\lambda}^* < 2t$.

1. Assuming that $f_A = f\ (\lambda = 0)$ and the initial step size t_0 are known, evaluate the function f at $\lambda = t_0$ and obtain $f_1 = f\ (\lambda = t_0)$.

2. If $f_1 > f_A$,set $f_C = f_1$ and evaluate the function f at

$\lambda = \frac{t_0}{2}\ and\ \tilde{\lambda}^*$ using Eq. (2.21) with $t = {}^{t_0}/_{2}$.

3. If $f_1 \leq f_A$, set $f_B = f_1$, and evaluate the function f

at $\lambda = 2t_0$ to find $f_2 = f\ (\lambda = 2t_0)$.

4. If f_2 turns out to be greater than f_1, set $f_C = f_2$and compute

$\tilde{\lambda}^*$ according to Eq. (2.21) with $t = t_0$.

5. If f_2 turns out to be smaller than f_1, set new $f_1 = f_2$ and $t_0 = 2t_0$, and repeat steps 2 to 4 until we are able to find $\tilde{\lambda}^*$.

Stage 3. The $\tilde{\lambda}^*$found in stage 2 is the minimum of the approximating quadratic $h(\lambda)$ and we have to make sure that this $\tilde{\lambda}^*$ is sufficiently close to the true minimum λ^* of $f\ (\lambda)$ before taking $\tilde{\lambda}^* \simeq \tilde{\lambda}^*$. several tests are possible to ascertain this. One possible test is to compare $f\ (\tilde{\lambda}^*)$ with $h(\tilde{\lambda}^*)$ and consider $\tilde{\lambda}^*$ a sufficiently good approximation if they differ not more than by a small amount. This criterion can be stated as

$$\left|\frac{h(\tilde{\lambda}^*) - f\ (\tilde{\lambda}^*)}{f\ (\tilde{\lambda}^*)}\right| \leq \varepsilon_1 \qquad (2.24)$$

Another possible test is to examine whether $df/d\lambda$ is close to zero at $\tilde{\lambda}^*$. Since the derivatives of f are not used in this method, we can use a finite-difference formula for $df/d\lambda$ and use the criterion

$$\left|\frac{f\left(\tilde{\lambda}^* + \Delta\tilde{\lambda}^*\right) - f\left(\tilde{\lambda}^* - \Delta\tilde{\lambda}^*\right)}{2\Delta\tilde{\lambda}^*}\right| \leq \varepsilon_2 \qquad (2.25)$$

to stop the procedure. In Eqs. (2.24) and (2.25), ε_1 and ε_2 are small numbers to be specified depending on the accuracy desired. If the convergence criteria stated in Eqs. (2.24) and (2.25) are not satisfied, a new quadratic function

$$h'(\lambda) = a' + b'\lambda + c'\lambda^2$$

is used to approximate the function $f(\lambda)$. To evaluate the constants $a', b', and\ c'$, the three best function values of the current $f_A = f(\lambda = 0), f_B = f(\lambda = t_0), f_C = f(\lambda = 2t_0)$, and $f\tilde{} == f(\lambda = \tilde{\lambda}^*)$ are to be used. This process of trying to fit another polynomial to obtain a better approximation to $\tilde{\lambda}^*$ is known as *refitting* the polynomial.

For refitting the quadratic, we consider all possible situations and select the best three points of the present A, B, C, and $\tilde{\lambda}^*$. There are four possibilities. A new value of $\tilde{\lambda}^*$ is computed by using the general formula, Eq. (2.17). If this $\tilde{\lambda}^*$ also does not satisfy the convergence criteria stated in Eqs. (2.24) and (2.25), a new quadratic has to be refitted.

Example 2.4 Find the minimum of $f = \lambda^5 - 5\lambda^3 - 20\lambda + 5$.

SOLUTION Since this is not a multivariable optimization problem, we can proceed directly to $stage\ 2$. Let the initial step size be taken as $t_0 = 0.5\ and\ A = 0$.

Iteration 1

$$f_A = f(\lambda = 0) = 5$$

$$f_1 = f(\lambda = t_0) = 0.03125 - 5(0.125) - 20(0.5) + 5 = -5.59375$$

Since $f_1 < f_A, we\ set\ f_B = f_1 = -5.59375$, and find that

$$f_2 = f(\lambda = 2t_0 = 1.0) = -19.0$$

As $f_2 < f_1$, we set new $t_0 = 1$ and $f_1 = -19.0$. Again we find that $f_1 < f_A$ and hence set $f_B = f_1 = -19.0$, and find that $f_2 = f(\lambda = 2t_0 = 2) = -43$. Since $f_2 < f_1$, we again set $t_0 = 2$ and $f_1 = -43$. As this $f_1 < f_A$, set $f_B = f_1 = -43$ and evaluate $f_2 = f(\lambda = 2t_0 = 4) = 629$. This time $f_2 > f_1$ and hence we set $f_C = f_2 = 629$ and compute $\tilde{\lambda}^*$ from Eq. (2.21) as

$$\tilde{\lambda}^* = \frac{4(-43) - 3(5) - 629}{4(-43) - 2(629) - 2(5)}(2) = \frac{1632}{1440} = 1.135$$

Convergence test: Since $A = 0, f_A = 5, B = 2, f_B = -43, C = 4$, and $f_C = 629$, the values of $a, b, and\ c$ can be found to be

$$a = 5, b = -204, c = 90$$

and

$$h(\tilde{\lambda}^*) = h(1.135) = 5 - 204(1.135) + 90(1.135)^2 = -110.9$$

Since

$$f\tilde{} = = f(\tilde{\lambda}^*) = (1.135)^5 - 5(1.135)^3 - 20(1.135) + 5.0 = -23.127$$

we have

$$\left|\frac{h(\tilde{\lambda}^*) - f(\tilde{\lambda}^*)}{f(\tilde{\lambda}^*)}\right| = \left|\frac{-116.5 + 23.127}{-23.127}\right| = 3.8$$

As this quantity is very large, convergence is not achieved and hence we have to use *refitting*.

Iteration 2

Since $\tilde{\lambda}^* < B\ and\ f\tilde{} > f_B$, we take the new values of $A, B, and\ C$ as

$$A = 1.135, \qquad f_A = -23.127$$
$$B = 2.0, \qquad f_B = -43.0$$
$$C = 4.0, \qquad f_C = 629.0$$

and compute new $\tilde{\lambda}^*$, using Eq. (2.17), as

$$\tilde{\lambda}^* = \frac{(-23.127)(4.0 - 16.0) + (-43.0)(16.0 - 1.29) + (629.0)(1.29 - 4.0)}{2[(-23.127)(2.0 - 4.0) + (-43.0)(4.0 - 1.135) + (629.0)(1.135 - 2.0)]}$$
$$= 1.661$$

Convergence test*:* To test the convergence, we compute the coefficients of the Quadratic as

$$a = 288.0, b = -417.0, c = 125.3$$

As

$$h(\tilde{\lambda}^*) = h(1.661) = 288.0 - 417.0(1.661) + 125.3(1.661)2 = -59.7$$
$$f\tilde{} = f(\tilde{\lambda}^*) = 12.8 - 5(4.59) - 20(1.661) + 5.0 = -38.37$$

we obtain

$$\left|\frac{h(\tilde{\lambda}^*) - f(\tilde{\lambda}^*)}{f(\tilde{\lambda}^*)}\right| = \left|\frac{-59.70 + 38.37}{-38.37}\right| = 0.556$$

Since this quantity is not sufficiently small, we need to proceed to the next refit.

2.3. Multidimensional search

2.3.1 Unvariate Method

In this method we change only one variable at a time and seek to produce a sequence of improved approximations to the minimum point. By starting at a base point X_i in the i^{th} iteration, we fix the values of $n - 1$ variables and vary the remaining variable. Since only one variable is changed, the problem becomes a one-dimensional minimization problem and any of the methods discussed above can be used to produce a new base point X_{i+1}. The search is now continued in a new direction. This new direction is obtained by changing any one of the $n - 1$ variables that were fixed in the previous iteration. In fact, the search procedure is continued by taking each coordinate direction in turn. After all the n directions are searched sequentially, the first cycle is complete and hence we repeat the entire process of sequential minimization. The procedure is continued until no further improvement is possible in the objective function in any of the n directions of a cycle. The univariate method can be summarized as follows [9]:

1. Choose an arbitrary staring point X_1 and set $i = 1$.

2. Find the search direction $\boldsymbol{s_i}$ as

$$S_i^T = \begin{cases} (1,0,0,\ldots,0) & \text{for i} = 1, \text{n} + 1, 2\text{n} + 1, \ldots \\ (1,0,0,\ldots,0) & \text{for i} = 2, \text{n} + 2, 2\text{n} + 2, \ldots \\ (0,0,1,\ldots,0) & \text{for i} = 3, \text{n} + 3, 2\text{n} + 3, \ldots \\ \quad\vdots & \\ (0,0,0,\ldots,1) & \text{for i} = \text{n}, 2\text{n}, 3\text{n}, \ldots \end{cases} \qquad (2.26)$$

3. Determine whether λ_i should be positive or negative. For the current direction $\mathrm{S}_{\boldsymbol{i}}$, this means find whether the function value decreases in the positive or negative direction. For this we take a small probe length (ε) and evaluate $f_i = f(X_i), f^+ = f(X_i + \varepsilon S_i)$, and $f^- = f(X_i - \varepsilon S_i)$. If $f^+ < f_i$, S_i will be the correct direction for decreasing the

value of f and if $f^- < fi$, $-S_i$ will be the correct one. If both $f^+$and f^- are greater than f_i, we take X_i as the minimum along the direction S_i .

4. Find the optimal step length λ_i^* such that

$$f(X_i \pm \lambda_i^* S_i) = \min_{\lambda_i}(X_i \pm \lambda_i S_i) \qquad (2.27)$$

where $+$ or $-$ sign has to be used depending upon whether S_i or $-S_i$ is the direction for decreasing the function value.

5. Set $X_{i+1} = X_i \pm \lambda_i^* S_i$ depending on the direction for decreasing the function value, and $f_{i+1} = f(X_{i+1})$.

6. if $\|X_{i+1} - X_i\| < \varepsilon$, stop. Otherwise go to 7

7. Set the new value of $i = i + 1$ and go to step2. Continue this procedure until no significant change is achieved in the value of the objective function.

The univariate method is very simple and can be implemented easily. However, it will not converge rapidly to the optimum solution, as it has a tendency to oscillate with steadily decreasing progress toward the optimum. Hence it will be better to stop the computations at some point near to the optimum point rather than trying to find the precise optimum point. In theory, the univariate method can be applied to find the minimum of any function that possesses continuous derivatives. However, if the function has a steep valley, the method may not even converge. If the univariate search starts at point P, the function value cannot be decreased either in the direction $\pm S_1$ or in the direction $\pm S_2$. Thus the search comes to a halt and one may be misled to take the point P, which is certainly not the optimum point, as the optimum point. This situation arises whenever the value of the probe length ε needed for detecting the proper direction $(\pm S_1 \; or \; \pm S_2)$ happens to be less than the number of significant figures used in the computations.

Example 2.5 Minimize $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$ with the starting point $(0, 0)$.

SOLUTION we will take the probe length (ε) as 0.01 to find the correct direction for decreasing the function value in step 3. Further, we will use the differential calculus method to find the optimum step length λ_i^* along the direction $\pm S_i$ in step 4.

Iteration $i = 1$

Step 2: Choose the search direction S_1 as $S_1 = \begin{Bmatrix}1\\0\end{Bmatrix}$.

Step 3: To find whether the value of f decreases along S_1 or $-S_1$, we use the probe length ε. Since

$$f_1 = f(X_1) = f(0,0) = 0,$$

$$f^+ = f(X_1 + \varepsilon S_1) = f(\varepsilon, 0) = 0.01 - 0 + 2(0.0001) + 0 + 0 = 0.0102 > f_1$$

$$f^- = f(X_1 - \varepsilon S_1) = f(-\varepsilon, 0) = -0.01 - 0 + 2(0.0001) + 0 + 0 = -0.0098 < f_1,$$

$-S_1$ is the correct direction for minimizing f from X_1.

Step 4: To find the optimum step length λ_1^*, we minimize

$$f(X_1 - \lambda_1 S_1) = f(-\lambda_1, 0) = (-\lambda_1) - 0 + 2(-\lambda_1)^2 + 0 + 0 = 2\lambda_1^2 - \lambda 1$$

As λ_1 is a step length we take $0 \leq \lambda_1 \leq 1$ and solving above equation for λ_1 using golden section method we get $\lambda_1^* = \frac{1}{4}$

Step 5: Set

$$X_2 = X_1 - \lambda_1^* S_1 = \begin{Bmatrix}0\\0\end{Bmatrix} - \frac{1}{4}\begin{Bmatrix}1\\0\end{Bmatrix} = \begin{Bmatrix}-\frac{1}{4}\\0\end{Bmatrix}$$

$$f_2 = f(f_2) = f(-\frac{1}{4}, 0) = -\frac{1}{8}$$

Iteration $i = \mathbf{2}$

Step 2: Choose the search direction S_2 *as* $S_2 = \begin{Bmatrix}0\\1\end{Bmatrix}$

Step 3: Since $f_2 = f(X_2) = -0.125$,

$$f^+ = f(X_2 + \varepsilon S_2) = f(-0.25, 0.01) = -0.1399 < f_2$$

$$f^- = f(X_2 + \varepsilon S_2) = f(-0.25, -0.01) = -0.1099 > f_2$$

S_2 is the correct direction for decreasing the value of f from X_2.

Step 4: We minimize $f(X_2 + \lambda_2 S_2)$ to find λ_2^2.

Here

$$f(X_2 + \lambda_2 S_2) = f(-0.25, \lambda_2)$$

$$= -0.25 - \lambda_2 + 2(0.25)^2 - 2(0.25)(\lambda 2) + \lambda_2^2$$

$$= \lambda_2^2 - 1.5\lambda_2 - 0.125$$

$$\text{using golden section method we get } \lambda_2^* = 0.75$$

Step 5: Set

$$X_3 = X_2 + \lambda_2^* S_2 = \begin{Bmatrix} -025 \\ 0 \end{Bmatrix} + 0.75 \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} + = \begin{Bmatrix} -0.25 \\ 0.75 \end{Bmatrix}$$

$$f_3 = f(X_3) = -0.6875$$

Next we set the iteration number as $i = 3$, and continue the procedure until the optimum solution $X^* = \begin{Bmatrix} -1.0 \\ 1.5 \end{Bmatrix}$ $with\ f(X^*) = -1.25$ is found.

Note: If the method is to be computerized, a suitable convergence criterion has to be used to test the point $X_{i+1} (i = 1, 2, \dots)$ for optimality.

2.3.2 Pattern Directions

In the univariate method, we search for the minimum along directions parallel to the coordinate axes. We noticed that this method may not converge in some cases and that even if it converges, its convergence will be very slow as we approach the optimum point. These problems can be avoided by changing the directions of search in a favorable manner instead of retaining them always parallel to the coordinate axes. Let the points $1, 2, 3, \dots$ indicate the successive points found by the univariate method. It can be noticed that the lines joining the alternate points of the search $(e.g., 1, 3;\ 2, 4;\ 3, 5;\ 4,6;\ \dots)$ lie in the general direction of the minimum and are known as *pattern directions*.

It can be proved that if the objective function is a quadratic in two variables, all such lines pass through the minimum. Unfortunately, this property will not be valid for multivariable functions even when they are quadratics. However, this idea can still be used to achieve rapid convergence while finding the minimum of an n-variable function. Methods that use pattern directions as search directions are known as *pattern search methods*.

One of the best-known pattern search methods, the Powell's method, is discussed here. In general, a pattern search method takes n univariate steps, where n denotes the number of design variables and then searches for the minimum along the pattern direction S_i, defined by

$$S_i = X_i - X_{i-n} \qquad (2.28)$$

Where X_i the point is obtained at the end of n univariate steps and X_{i-n} is the starting point before taking the n univariate steps. In general, the directions used prior to taking a move along a pattern direction need not be univariate directions.

2.3.4 POWELL'S METHOD

Powell's method is an extension of the basic pattern search method. It is the most widely used direct search method and can be proved to be a method of conjugate directions [9]. A conjugate directions method will minimize a quadratic function in a finite number of steps. Since a general nonlinear function can be approximated reasonably well by a quadratic function near its minimum, a conjugate directions method is expected to speed up the convergence of even general nonlinear objective functions.

The definition, a method of generation of conjugate directions, and the property of quadratic convergence are presented in this section.

2.3.4.1 Conjugate Directions

Definition2.4 (Conjugate Directions). Let $A = [A]$ be an $n \times n$ symmetric matrix. A set of n vectors (or directions) $\{S_i\}$ is said to be conjugate (more accurately A-conjugate) if

$$S_i^T AS_j = 0 \; for \; all \; i \neq j, \qquad i = 1, 2, \ldots, n, j = 1, 2, \ldots, n \qquad (2.29)$$

It can be seen that *orthogonal directions* are a special case of conjugate directions($obtained \; with \; [A] = [I] \; in \; Eq.(2.29)$.

Definition2.5 (Quadratically Convergent Method): If a minimization method, using exact arithmetic, can find the minimum point in n steps while minimizing a quadratic function in n variables, the method is called a *quadratically convergent method*.

Theorem 2.3 Given a quadratic function of n variables and two parallel hyper planes 1 and 2 of dimension $k < n$. Let the constrained stationary points of the quadratic function in the hyper planes be X_1 and X_2, respectively. Then the line joining X_1 and X_2 is conjugate to any line parallel to the hyperplanes.

Proof: Let the quadratic function be expressed as

$$Q(X) = \frac{1}{2} X^T AX + B^T X + C \qquad (2.30)$$

The gradient of Q is given by

$$\nabla Q(X) = AX + B$$

and hence

$$\nabla Q(X_1) - \nabla Q(X_2) = A(X_1 - X_2) \quad (2.31)$$

If S is any vector parallel to the hyper planes, it must be orthogonal to the gradients $\nabla Q(X_1)$ *and* $\nabla Q(X_2)$. Thus

$$S^T \nabla Q(X_1) = S^T A X_1 + S^T B = 0 \quad (2.32)$$

$$S^T \nabla Q(X_2) = S^T A X_2 + S^T B = 0 \quad (2.33)$$

By subtracting $Eq.(6.29)$ from Eq. (6.28), we obtain

$$S^T A(X_1 - X_2) = 0 \quad (2.34)$$

Hence S and $(X_1 - X_2)$ are A-conjugate.

Theorem 2.4 If a quadratic function

$$Q(X) = \frac{1}{2} X^T A X + B^T X + C \quad (2.35)$$

is minimized sequentially, once along each direction of a set of n mutually conjugate directions, the minimum of the function Q will be found at or before the nth step irrespective of the starting point.

Proof: Let X^* minimize the quadratic function $Q(X)$. Then

$$\nabla Q(X^*) = B + AX^* = 0 \quad (2.36)$$

Given a point X_1 and a set of linearly independent directions $S_1, S_2, \ldots, S_n$, constants β_i Can always be found such that

$$X^* = X_1 + \sum_{i=1}^{n} \beta_i S_i \quad (2.37)$$

where the vectors $S_1, S_2, \ldots, S_n$ have been used as basis vectors. If the directions S_i are A-conjugate and none of them is zero, the **S**i can easily be shown to be linearly independent and the β_i can be determined as follows.

Equations (2.36) and (2.37) lead to

$$B + AX_1 + A\left(\sum_{i=1}^{n} \beta_i S_i\right) = 0 \quad (2.38)$$

Multiplying this equation throughout by S_j^T , we obtain

$$S_j^T(B + AX_1) + S_j^T A\left(\sum_{i=1}^{n} \beta_i S_i\right) = 0 \qquad (2.39)$$

Equation (2.39) can be rewritten as

$$(B + AX_1)^T S_j + \beta_j S_j^T AS_j = 0 \qquad (2.40)$$

that is,

$$\beta_j = -\frac{(B + AX_1)^T S_j}{S_j^T AS_j} \qquad (2.41)$$

Now consider an iterative minimization procedure starting at point X_1, and successively minimizing the quadratic $Q(X)$ in the directions $S_1, S_2, \ldots, S_n$, where these directions satisfy Eq. (2.29). The successive points are determined by the relation

$$X_{i+1} = X_i + \lambda_i^* S_i, i = 1 \text{ to } n \qquad (2.42)$$

where λ_i^* is found by minimizing $Q(X_i + \lambda_i S_i)$ so that

$$S_i^T \nabla Q(X_{i+1}) = 0 \qquad (2.43)$$

Since the gradient of Q at the point X_{i+1} is given by

$$\nabla Q(X_{i+1}) = B + AX_{i+1} \qquad (2.44)$$

Eq. (2.43) can be written as

$$S_i^T \{B + A(X_i + \lambda_i^* S_i)\} = 0 \qquad (2.45)$$

This equation gives

$$\lambda_i^* = -\frac{(B + AX_i)^T S_i}{S_i^T AS_i} \qquad (2.46)$$

From Eq. (2.42), we can express $\boldsymbol{X_i}$ as

$$X_i = X_1 + \sum_{j=1}^{i-1} \lambda_j^* S_j \qquad (2.47)$$

so that

$$X_i^T AS_i = X_1^T AS_i + \sum_{j=1}^{i-1} \lambda_j^* S_j^* AS_j$$

$$= X_1^T AS_i \qquad (2.48)$$

using the relation (2.29). Thus Eq. (2.46) becomes

$$\lambda_i^* = -(\mathrm{B} + \mathrm{AX}_1)^T \frac{\mathrm{S_i}}{\mathrm{S_i^T AS_i}} \qquad (2.49)$$

which can be seen to be identical to Eq. (2.41). Hence the minimizing step lengths are given by β_i *or* λ_i^* . Since the optimal point X^* is originally expressed as a sum of n quantities$\beta_1, \beta_2, \ldots, \beta_n$, which have been shown to be equivalent to the minimizing step lengths, the minimization process leads to the minimum point in n steps or less.

Since we have not made any assumption regarding X_1 and the order of $\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_n$, the process converges in n steps or less, independent of the starting point as well as the order in which the minimization directions are used.

Example 2.6 Consider the minimization of the function

$$f(x_1, x_2) = 6x_1^2 + 2x_2^2 - 6x_1x_2 - x_1 - 2x_2$$

If $\boldsymbol{S_1} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$ denotes a search direction, find a direction S_2 that is conjugate to the direction S_1.

SOLUTION The objective function can be expressed in matrix form as

$$f(X) = B^T X + \frac{1}{2} X^T [A] X$$

$$= \{-1 \ -2\} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \frac{1}{2} \{x_1 \ x_2\} \begin{bmatrix} 12 & -6 \\ -6 & 4 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}$$

and the Hessian matrix $[A]$ can be identified as

$$[A] = \begin{bmatrix} 12 & -6 \\ -6 & 4 \end{bmatrix}$$

The direction $S_2 = \begin{Bmatrix} s_1 \\ s_2 \end{Bmatrix}$ will be conjugate to $S_1 = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$ if

$$S_1^T [A] S_2 = (1 \ 2) \begin{bmatrix} 12 & -6 \\ -6 & 4 \end{bmatrix} \begin{Bmatrix} s_1 \\ s_2 \end{Bmatrix} = 0$$

which upon expansion gives $2s_2 = 0$ or $s_1 =$ arbitrary and $s_2 = 0$. Since s_1 can have any value, we select $s_1 = 1$ and the desired conjugate direction can be expressed as $S_2 = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}$.

Powell's Algorithm

In Powell's method for a two-variable function the function is first minimized once along each of the coordinate directions starting with the second coordinate direction and then in the corresponding pattern direction. For the next cycle of minimization, we discard one of

the co ordinate d irections (the x_1 direction in t he p resent ca se) i n f avor o f t he p attern direction.

Then w e ge nerate a ne w pa ttern d irection S_2. F or t he ne xt c ycle of m inimization, we discard one of the previously used coordinate directions (the x_2 direction in this case) in favor of t he ne wly ge nerated pa ttern di rection. T hen, w e minimize a long di rections S_1 and S_2. F or t he ne xt c ycle of minimization, s ince t here i s n o c oordinate di rection to discard, w e r estart t he w hole p rocedure b y minimizing a long t he x_2 direction. T his procedure is continued until the desired minimum point is found.

Note that the search will be made sequentially in the directions S_n; $S_1, S_2, S_3, \ldots, S_{n-1}, S_n$; $S_p^{(1)}$; $S_2, S_p^{(1)}$, $S_p^{(1)}$; $S_p^{(2)}$; $S_{\boldsymbol{p}}^{(\mathbf{1})}, S_p^{(1)}$, $S_{\boldsymbol{p}}^{(2)}$; $S_p^{(3)}$, ... until the m inimum point i s found. Here S_i indicates the coordinate direction u_i and $S_p^{(j)}$ the j^{th} pattern direction.

Quadratic Convergence the p attern d irections $S_p^{(1)}$, $S_p^{(2)}$, $S_p^{(3)}$, ... are n othing but the lines joining the minima found along the directions S_n, $S_p^{(1)}$, $S_p^{(2)}$, ..., respectively. Hence by T heorem 2.3, t he pa irs of directions $(S_n, S_p^{(1)})$, $(S_p^{(1)}, S_p^{(2)})$, and s o on, are A-conjugate. T hus all t he directions $S_n, S_p^{(1)}, S_p^{(2)}$, ... are A-conjugate. S ince, by T heorem 2.4, a ny s earch m ethod i nvolving minimization a long a s et of c onjugate di rections i s quadratically convergent,

Powell's method is quadratically convergent. From the method used for constructing the conjugate directions $S_p^{(1)}$, $S_p^{(2)}$, . . . , we find that n minimization cycles are required to complete the construction of n conjugate directions. In the ith cycle, the minimization is done along t he already c onstructed i conjugate directions a nd t he $n - i$ nonconjugate (coordinate) d irections. T hus a fter n cycles, al l t he n search d irections ar e m utually conjugate a nd a qua dratic w ill t heoretically be m inimized in n^2 one-dimensional minimizations. This proves the *quadratic convergence* of Powell's method.

It is to be noted that as with most of the numerical techniques, the convergence in many practical problems may not be as good as the theory seems to indicate. Powell's method may require a lot more iterations to minimize a function than the theoretically estimated number. There are several reasons for this:

1. Since the number of cycles n is valid only for quadratic functions, it will take generally greater than n cycles for nonquadratic functions.

2. The proof of *quadratic convergence* has been established with the assumption that the exact minimum is found in each of the one-dimensional minimizations.

However, the actual minimizing step lengths λ_i^* will be only approximate, and hence the subsequent directions will not be conjugate. Thus the method requires more number of iterations for achieving the overall convergence.

3. Powell's method, described above, can break down before the minimum point is found. This is because the search directions S_i might become dependent or almost dependent during numerical computation.

Powell's method is a very popular means of successive minimizations along conjugate directions. It is a zero-order method, requiring the evaluation of $F(x)$ only. If the Problem involves *n* design variables, the basic algorithm is given by the following [3]

• Choose a point x_0 in the design space.

• Choose the starting vectors $v_i, i = 1, 2, \ldots, n$ (the usual choice is $v_i = e_i$, where e_i is the unit vector in the x_i-coordinate direction).

• Cycle

– do with $i = 1, 2, \ldots, n$

* Minimize $F(x)$ along the line through x_{i-1} in the direction of v_i. Let the minimum point be x_i.

– end do

– $v_{n+1} \leftarrow x_0 - x_n$ (this vector can be shown to be conjugate to v_{n+1} produced in the previous cycle)

– Minimize $F(x)$ along the line through x_0 in the direction of v_{n+1}. Let the minimum Point be x_{n+1}.

– if $|x_{n+1} - x_0| < \varepsilon$ exit loop

– do with $i = 1, 2, \ldots, n$

* $\boldsymbol{v_i \leftarrow v_{i+1}}$ (v_1 is discarded, the other vectors are reused)

– end do

• end cycle

Powell demonstrated that the vectors v_{n+1} produced in successive cycles are mutually conjugate, so that the minimum point of a quadratic surface is reached in precisely cycles. In practice, the merit function is seldom quadratic, but as long as it can be approximated locally, Powell's method will work. Of course, it usually takes more than n cycles to arrive at the minimum of a nonquadratic function. Note that it takes n line minimizations to construct each conjugate direction.

We start with point x_0 and vectors v_1 and v_2. Then we find the distance s_1 that minimizes $F(x_0 + sv_1)$, finishing up at point $x_1 = x_0 + s_1 v_1$. Next, we determine s_2 that minimizes $F(x_1 + sv_2)$ which takes us to $x_2 = x_1 + s_2 v_2$. The last search direction is $v_3 = x_2 - x_0$. After finding s_3 by minimizing $F(x_0 + sv_3)$ we get to $x_3 = x_0 + s_3 v_3$, completing the cycle.

As explained before, the first cycle starts at point P_0 and ends up at P_3. The second cycle takes us to P_6, which is the optimal point. The directions P_0P_3 and P_3P_6 are mutually conjugate.

Powell's method does have a major flaw that has to be remedied if $F(x)$ is not a quadratic; the algorithm tends to produce search directions that gradually become linearly dependent, thereby ruining the progress toward the minimum. The source of the problem is the automatic discarding of v_1 at the end of each cycle. It has been suggested that it is better to throw out the direction that resulted in the *largest decrease* of $F(x)$, a policy that we adopt. It seems counter-intuitive to discard the best direction, but it is likely to be close to the direction added in the next cycle, thereby contributing to linear dependence. As a result of the change, the search directions cease to be mutually conjugate, so that a quadratic form is not minimized in n cycles any more. This is not a significant loss, since in practice $F(x)$ is seldom a quadratic anyway.

***Example* 2.7** Minimize $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$ from the starting point $\boldsymbol{X_1} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$ using Powell's method.

SOLUTION

Cycle 1: Univariate Search

We minimize f along $S_2 = S_n = \begin{Bmatrix}0\\1\end{Bmatrix}$ from X_1. To find the correct direction $(+S_2\ or\ -S_2)$ for decreasing the value of f , we take the probe length as $\varepsilon = 0.01$. As

$f_1 = f(X_1) = 0.0$, and

$$f^+ = f(X_1 + \varepsilon S_2) = f(0.0, 0.01) = -0.0099 < f_1$$

f decreases along the direction +**S**2. To find the minimizing step length λ^* along S_2, we minimize

$$f(X_1 + \lambda S_2) = f(0.0, \lambda) = \lambda^2 - \lambda$$

Using golden section method we get$\lambda^* = 12$, *we have* $X_2 = X_1 + \lambda^* S_2 = \begin{Bmatrix}0\\0.5\end{Bmatrix}$.

Next we minimize f along $S_1 = \begin{Bmatrix}0\\1\end{Bmatrix}$from $X_2 = \begin{Bmatrix}0.5\\0.0\end{Bmatrix}$ since

$$f2 = f(X2) = f(0.0, 0.5) = -0.25$$

$$f+ = f(X2 + \varepsilon S1) = f(0.01, 0.50) = -0.2298 > f2$$

$$f- = f(X2 - \varepsilon S1) = f(-0.01, 0.50) = -0.2698$$

f decreases al ong$-S_1$. As $f(X_2 - \lambda S_1) = f(-\lambda, 0.50) = 2\lambda^2 - 2\lambda - 0.25$,using golden section method we get$\lambda^* = \frac{1}{2}$. Hence $X_3 = X_2 - \lambda^* S_1 = \begin{Bmatrix}-0.5\\0.5\end{Bmatrix}$

Now we minimize f along $S_2 = \begin{Bmatrix}0\\1\end{Bmatrix}$ from $X_3 = \begin{Bmatrix}-0.5\\0.5\end{Bmatrix}$,as$f_3 = f(\boldsymbol{X_3}) = -0.75$,

$f^+ = f(X_3 + \varepsilon S_2) = f(-0.5, 0.51) = -0.7599 < f_3$, f decreases al ong $+\boldsymbol{S}2$ direction.

Since

$f(X_3 + \lambda S_2) = f(-0.5, 0.5 + \lambda) = \lambda^2 - \lambda - 0.75$, using gol den s ection m ethod we get $\lambda^* = \frac{1}{2}$

This gives

$$X_4 = X_3 + \lambda^* S_2 = \begin{Bmatrix}-0.5\\1.0\end{Bmatrix}$$

Cycle 2: Pattern Search

Now we generate the first pattern direction as

$$S_p^{(1)} = X_4 - X_2 = \begin{Bmatrix}-\frac{1}{2}\\1\end{Bmatrix} - \begin{Bmatrix}0\\\frac{1}{2}\end{Bmatrix} = \begin{Bmatrix}-0.5\\0.5\end{Bmatrix}$$

and minimize f along $\boldsymbol{S_p^{(1)}}$ from X_4. Since

$$f4 = f(X_4) = -1.0$$

$$f^+ = f(X_4 + \varepsilon S_p^{(1)} = f(-0.5 - 0.005, 1 + 0.005)$$

$$= f(-0.505, 1.005) = -1.004975$$

f decreases in the positive direction of $S_p^{(1)}$. As

$f(X_4 + \lambda S_p^{(1)} = f(-0.5 - 0.5\lambda, 1.0 + 0.5\lambda) = 0.25\lambda^2 - 0.50\lambda - 1.00,$

using golden section method we get $\lambda^* = 1.0$ and hence

$$X_5 = X_4 + \lambda^* S_p^{(1)} = \begin{Bmatrix} -\frac{1}{2} \\ 1 \end{Bmatrix} + 1.0 \begin{Bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{Bmatrix} = \begin{Bmatrix} -1 \\ 1.5 \end{Bmatrix}$$

The point X_5 can be identified to be the optimum point.

If we do not recognize X_5 as the optimum point at this stage, we proceed to minimize f along the direction $S_2 = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$ from X_5. Then we would obtain

$f_5 = f(X_5) = -1.25$, $f^+ = f(X_5 + \varepsilon S_2) > f_5$, and $f^- = f(X_5 - \varepsilon S_2) > f_5$

This shows that f cannot be minimized along S_2, and hence X_5 will be the optimum point.

In this example the convergence has been achieved in the second cycle itself.

This is to be expected in this case, as f is a quadratic function, and the method is a Quadratically convergent method.

The n umerical re sult f or th is example i s s ummarized be low (see a ppendix f or t he Powell's matlab code).

xmin	fmin
(-1.503, 2.356)	-0.872520631787970
(-1.000, 1.500)	-1.249999992736614
(-1.000, 1.500)	-1.249999999974476

The minimum point (-1.000, 1.500) is reached at the 3th cycle.

Chapter 3
Trust region methods

3.1Trust region frame work

In the last section of the third chapter we explained the trust region methods as one of the solution procedures for treating a nonlinear programming problem. Here a brief review will be helpful to follow the integration of trust region method into DFO algorithm. The trust region frame work is usually in the context when at least the gradient and sometimes Hessian of the objective function can be evaluated or estimated accurately.

Main steps of a typical trust region method are [2]

1. Given a current iterate, build a good local approximation model.
2. Choose a neighborhood around the current iterate when the model 'is trusted' to be accurate. Minimize the model in this neighborhood.
3. Determine if the step is successful by evaluating the true objective function at the new point comparing the true reduction in value of the objective with the reduction predicted by the model.
4. If the step is successful, accept the new point as the next iterate. Increase the size of the trust region, if the success is really significant. Otherwise reject the new point and reduce the size of the trust region.
5. Repeat until convergence.

For a model based on the Taylor series expansion we know that if the trust region is made small enough, then the approximation is sufficiently accurate and the algorithm will make a successful step (unless the optimum has been reached).

To use trust region fame work in derivative free case we use an alternative approximation technique, which does not use derivative estimates. Quadratic interpolation is one such technique which can be applied successfully with in a trust region method. However, we need to guarantee that the approximation model is locally good: that is that a successful step will be made after sufficient reduction of the trust region.

3.2 Quadratic interpolation

Consider the problem of interpolating a given or suitably chosen function $f: \mathbb{R}^n \to \mathbb{R}$ by a quadratic p olynomial $Q(x)$ at a ch osen set o f p oints $Y = \{y^1, y^2, \dots, y^p\} \subseteq \mathbb{R}^n$. the quadratic polynomial $Q(x)$ is an interpolation of the function $f(x)$ with respect to the set Y if

$$Q(y^j) = f(y^j) \quad (j = 1,2, \dots, p) \qquad (3.1)$$

Such, t hat f i s know n a t a ll f initely m any e lements o f Y. H ere w e n ote that Q is our model w hich w e de fined a s m_k in th e f irst c hapter. W ith th e c ontext o f trust region methods; that

$$m_k(x) = Q(x)$$

Suppose that the space of quadratic polynomials is spanned by a set of basis functions

$\phi_i(.)(i = 1, 2, \dots q)$.

Then any quadratic polynomial can be written in terms of these basis functions, that is

$$Q(x) = \sum_{i=1}^{q} \alpha_i \, \phi_i(x),$$

Where, the coefficient vector $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_q)^T$ is to be determined. We need

$q = 1 + n + n + n\left(\frac{n-1}{2}\right) = \frac{1}{2}(n+1)(n+2)$ Points to f ind a ll o f th e in terpolation parameters. If we have $\frac{1}{2}(n+1)(n+2)$ points, we can ensure that the quadratic model is entirely de termined by t he f ollowing s ystem of e quations. When t his i s t he c ase, t he system of linear equations

$$\sum_{i=1}^{q} \alpha_i \, \phi_i(y^j) = f(y^j) \quad (j = 1,2, \dots, p) \qquad (3.2)$$

Can be solved to, derive the interpolation parameters. The parameter or coefficient matrix of this system is of the type $p \times q$ and looks as follows;

$$\phi(Y) = \begin{pmatrix} \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \ddots & \vdots \\ \phi_1(y^p) & \cdots & \phi_q(y^p) \end{pmatrix}. \qquad (3.3)$$

For a given set of points and a set of function values, an interpolation polynomial exists and is unique if and only if $\phi(Y)$ is square, that is $p = q$, and nonsingular. Theoretically, this means that the system (3.3) can be solved but in practice the solvability of this system depends on whether the matrix $\phi(Y)$ is ill-conditioned or not.

From the above argument, we conclude that if we manage to determine the quadratic polynomial uniquely, then we have $p = q = \frac{1}{2}(n+1)(n+2)$. however, we need to be aware of the fact that not any $\frac{1}{2}(n+1)(n+2)$ point in $\mathbb{R}^n$ can be interpolated by a quadratic polynomial. Obviously although, 3 distinct points can be interpolated by a quadratic function in univariate interpolation, this is not the case in multivariate interpolation. in fact, 3 points will not be enough to obtain a quadratic interpolation polynomial whenever the dimension of the interpolation space is greater than one. By inspection, one can see that 6 points are necessary to obtain a unique quadratic interpolation of a function in two dimensions. However, an interpolation set Y of six points lying on one line cannot be interpolated by a quadratic function. Therefore, the points of Y must satisfy a geometric condition to ensure the existence and uniqueness of the quadratic model. This geometric condition is known as the poisedness of the point set.
Definition3.1: A set of points Y is called poised, with respect to a given subspace of polynomials, if the considered function $f(x)$ can be interpolated at the points of Y by the polynomials from this subspace, that is, if there always exists a suitable interpolating polynomial in that sub space.

Remark: In DFO, poisedness is a necessary geometric condition on the interpolation set Y that ensures the existence and uniqueness of the quadratic model $Q(x)$wanted and used in DFO algorithm.

We illustrate the implied geometric character of poisedness by the following examples.

Example3.1: Suppose n = 2 and Y is a set of six points on a unit circle. Then, $Y \subseteq \mathbb{R}^2$ cannot be interpolated by a polynomial of the form

$a_0 + a_1x_1 + a_2x_2 + a_{1,1}x_1^2 + a_{1,2}x_1x_2 + a_{2,2}x_2^2$. Hence, Y is not poised with respect to the

Space of quadratic polynomials. On the other hand, Y can be interpolated by a polynomial of the form

$a_0 + a_1x_1 + a_2x_2 + a_{1,1}x_1^2 + a_{1,2}x_1x_2 + a_{2,2}x_2^2 + a_{1,1,1}x_1^3$. Therefore Y is poised in an appropriate subspace of the space of cubic polynomials.

Example3.2: Consider the two quadrics $q_1(x, y) = 2x + x^2 - y^2$ and

$q_2(x, y) = x^2 + y^2$, Whose intersection curve I projects in the $(x, y) - plane$ to the conics

$C(x, y) = 0 with\ C(x, y) = x - y^2$. Namely,

$$\begin{aligned} &2x + x^2 - y^2 = x^2 + y^2, \\ &\Leftrightarrow 2x = 2y^2 \\ &\Leftrightarrow x = y^2 \end{aligned}$$

Definition3.2: A set of points Y is called $well - poised$, if it remains poised under small perturbations. For example, if $n = 2$, six points almost on a line may define a poised set . However, since some small perturbation of the points might make them aligned, it is not a well-poised set.

As we mentioned before, a set of points is poised if $\phi(Y)$ is nonsingular with respect to the space of quadratic polynomials. If we look for an understanding of this by the fact that an interpolation polynomial exists and is unique if and only if $\phi(Y)$ is square and nonsingular, then we conclude:

Y is poised if the determinant of $\phi(Y)$ is nonvanishing, that is,

$$\delta(y) = det\begin{pmatrix} \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \ddots & \vdots \\ \phi_1(y^p) & \cdots & \phi_q(y^p) \end{pmatrix} = det\phi(Y) \neq 0 \qquad (3.4)$$

The measure of poisedness in a DFO algorithm can be explained by a methodology based on Newton fundamental polynomials. In DFO, the approach of handling the poisedness in combination with the Newton fundamental polynomials is a distinctive issue. This is so, because it allows us not only to choose a good interpolation set from a given set of sample points but also to find a new sample point which improves the poisedness of the interpolation set. If we had no such a useful tool, then, removing a point from the set would have caused the conditioning of the coefficient matrix to get worse in the updating step for the interpolation set of DFO. There is also a detailed work on DFO in which the quadratic approximation model is determined by Lagrange interpolation polynomials instead of the Newton fundamental polynomials

Let us focus on the Newton fundamental points: The points y in our interpolation

Set $Y = \{y^1, y^2, \dots, y^p\}$ *which* is a subset of $\mathbb{R}^n$ are organized into $d+1$ blocks, where $Y^{[l]}$ $(l = 1, 2, \dots, d)$ is the l^{th} block, containing $\left|Y^{[l]}\right| = \binom{l+n-1}{l}$ points.

Definition3.3: A single Newton fundamental polynomial of degree l corresponds to each point $(y^i)^{[l]} \in Y^{[l]}$ satisfying the following conditions:

$N_i^l(y^j)^{[m]} = \delta_{ij}\,\delta_{lm}$ for all $(y^j)^{[m]}$ with $m \in \{0,1,2, \dots, l\}$.

Here we refer to Kronecker's symbol for $i, j = 0,1,2, \dots, l$:

$$\delta_{ij} = \begin{cases} 1, & if\ i = j \\ 0, & else. \end{cases}$$

Consider the set of interpolation points being partitioned into three disjoint blocks $Y^{[0]}, Y^{[1]}, Y^{[2]}$ which correspond to the constant term, the linear terms and the quadratic terms of a quadratic polynomial, respectively. Hence $Y^{[0]}$ has a single element, $Y^{[1]}$ has n elements and $Y^{[2]}$ has $\frac{n(n+1)}{2}$ elements. The basis $\{N_i(.)\}$ of NFP is also partitioned into three blocks $\{N_i^0(.)\}, \{N_i^1(.)\}, \{N_i^2(.)\}$ with the approximate number of elements in each block. The unique element of $\{N_i^0(.)\}$ is a polynomial of degree zero. Each of the n elements $\{N_i^1(.)\}$ is a polynomial of degree one and, finally, each of the $\frac{n(n+1)}{2}$ elements of $\{N_i^2(.)\}$ is a polynomial of degree two.

The basis elements and the interpolation points are set in one to one correspondence, so that the points from block $Y^{[l]}$ correspond to the polynomials from block $\{N_i^1(.)\}$. a

Newton Fundamental Polynomial (NFP) $N_i(.)$ and a point y^i are in the correspondence with each other if and only if the value of that polynomial at that point is one and its value at any other point in the same block or in any previous block is zero. In other words, if y^i corresponds to N_i, then $N_i(y^i) = 1$ and $N_i(y^j) = 0$ for all other indices j

Example3.3: consider the quadratic interpolation on a plane. We require six interpolation points using three blocks.

$Y^{[0]} = \{(0,0)\}, Y^{[1]} = \{(1,0), (0,1),\}\ and\ \ Y^{[2]} = \{(2,0), (1,1), (0,2)\}$

Corresponding to the initial basis functions; $1, x_1, x_2, x_1x_2, and\ x_2^2$ respectively. Applying some procedures we find the NFP:

$N_1^0 = 1, N_1^1 = x_1, N_2^1 = x_2, N_1^2 = \frac{x_1^2 - x_1}{2}$, $N_2^2 = x_1x_2\ and\ N_3^2 = \frac{x_2^2 - x_2}{2}$

Algorithm *(derivative free trust region method)*

The steps of derivative free trust region methods are given as follows [6]

Step 0: initializations

Let a starting point x_s and the value of $f(x_s)$ be given.

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

Choose at least one additional point not further than $\Delta_0 > 0$ away from x_s to create an initial well-poised interpolation set Y and initial basis of Newton fundamental polynomials.

Determine $x_0 \in$ Y which has the best objective function value; i.e. x_0 solves the problem

$$\min_{s.t\ x \in Y} f(x)$$

Set $k = 0$, choose parameters $\eta_0, \eta_1, where\ 0 < \eta_0 < \eta_1 < 1\ and\ 0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$

Step 1: build the model using the interpolation set Y and basis of NFP, build a quadratic interpolation polynomial $Q_k(x)$.

Step 2: minimize the model with in the trust region.

Set $\beta_k = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$. compute the point $\hat{x}_k$ such that

$$Q_k(\hat{x}_k) = \min_{x \in \beta_k} Q_k\,(x).$$

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

$$\rho_k = \frac{f(x_k) - f(\hat{x}_k)}{Q_k(x_k) - Q_k(\hat{x}_k)}.$$

Step3: 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 β_k, generate new interpolation point in β_k, while preserving/ impronving well-poissedness.
- 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 the cardinality of $Y \cap \beta_k$ was less than $n+1$ when $\hat{x}_k$ was computed, reduce the trust region radius

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

$$otherwise\ set\ \Delta_{k+1} = \Delta_k$$

Step 5: update the current iterate.

Determine $\bar{x}_k$ with best objective function value by solving the discrete problem

$$\min_{\substack{y^i \in Y \\ y^i \neq x_k}} f(y^i)$$

If improvement is sufficient (in the sense of prediction) that is

$\frac{f(x_k)-f(\bar{x}_k)}{Q_k(x_k)-Q_k(\hat{x}_k)} \geq \eta_0$ Then we put $\bar{\rho}_k = \frac{f(x_k)-f(\bar{x}_k)}{Q_k(x_k)-Q_k(\hat{x}_k)}$. Set $x_{k+1} = \bar{x}_k$. otherwise $x_{k+1} = x_k$. increase k by one, and go to step one.

Appendix (MATLAB codes)

1. Code for Golden search

```
function [xmin,fmin] = goldSearch(f,a,b)

% Golden section search for the minimum of f(x).

% The minimum point must be bracketed in a <= x <= b.

% usage: [fmin,xmin] = goldsearch(func,xstart,h)

% input:

% f = handle of function that returns f(x).

% a, b = limits of the interval containing the minimum.

% output:

% fmin = minimum value of f(x).

% xmin = value of x at the minimum point.

N=20;% N is the number of function evaluations done.

c = (-1+sqrt(5))/2;

x1 = c*a + (1-c)*b;

f1 = feval(f,x1);

x2 = (1-c)*a + c*b;

f2 = feval(f,x2);

%fprintf('-----------------------------------------------------------------\n');

fprintf('     x1           x2          f(x1)       f(x2)         b - a \n');

fprintf('-----------------------------------------------------------------\n');
```

```
fprintf('%.4e    %.4e    %.4e    %.4e    %.4e\n', x1, x2, f1, f2, b-a);
% Main loop
for i = 1:N-2
  if f1 < f2
     b = x2;
     x2 = x1;
     f2 = f1;
     x1 = c*a + (1-c)*b;
     f1 = feval(f,x1);
  else
     a = x1;
     x1 = x2;
     f1 = f2;
     x2 = (1-c)*a + c*b;
     f2 = feval(f,x2);
  end;
  fprintf('%.4e    %.4e    %.4e    %.4e    %.4e\n', x1, x2, f1, f2, b-a);

end
   if (abs(b-a) < eps)
     fprintf('succeeded after %d steps\n', i);
```

```
    return;
  end;
if f1 < f2; fmin = f1; xmin = x1;
else
   fmin = f2; xmin = x2;
end
```

2. Code for Powell method

The algorithm for Powell's method is listed below. It utilizes two arrays: df contains the decreases of the merit function in the first *n* moves of a cycle, and the matrix u stores the corresponding direction vectors $\boldsymbol{v_i}$(one vector per column).

To implement this algorithm we use the gold bracket and the gold search algorithms together with it.

i. Gold bracket

```
function [a,b] = goldBracket(fun,x1,h)
% Brackets the minimum point of f(x).
% USAGE: [a,b] = goldBracket(func,xStart,h)
% INPUT:
% func   = handle of function that returns f(x).
% x1     = starting value of x.
% h      = initial step size used in search.
% OUTPUT:
% a, b = limits on x at the minimum point.
```

```
c = 1.618033989;
f1 = feval(fun,x1);
x2 = x1 + h; f2 = feval(fun,x2);
% Determine downhill direction & change sign of h if needed.
if f2 > f1
   h = -h;
   x2 = x1 + h; f2 = feval(fun,x2);
   % Check if minimum is between x1 - h and x1 + h
   if f2 >f1
      a = x2; b = x1 - h; return
   end
end
% Search loop
for i = 1:50
   h = c*h;
   x3 = x2 + h; f3 = feval(fun,x3);
   if f3 >f2
      a = x1;
      b = x3;
      return
   end
```

```
    x1 = x2; x2 = x3; f2 = f3;
end
error('goldbracket did not find minimum')
```

ii. Golden search

```
function [xmin,fmin] = goldensearch(f,a,b)
% Golden section search for the minimum of f(x).
% The minimum point must be bracketed in a <= x <= b.
% usage: [fmin,xmin] = goldsearch(func,xstart,h)
% input:
% f = handle of function that returns f(x).
% a, b = limits of the interval containing the minimum.
% output:
% fmin = minimum value of f(x).
% xmin = value of x at the minimum point.
%eps = 1.0e-3;
N = 20; % N is the number of function evaluations done.
c = (-1+sqrt(5))/2;
x1 = c*a + (1-c)*b;
f1 = feval(f,x1);
x2 = (1-c)*a + c*b;
f2 = feval(f,x2);
```

```
% Main loop
for i = 1:N-2
   if f1 < f2
      b = x2;
      x2 = x1;
      f2 = f1;
      x1 = c*a + (1-c)*b;
      f1 = feval(f,x1);
   else
      a = x1;
      x1 = x2;
      f1 = f2;
      x2 = (1-c)*a + c*b;
      f2 = feval(f,x2);
   end;
end
if f1 < f2
   fmin = f1; xmin = x1;
else
   fmin = f2; xmin = x2;
end
```

iii. function powell

```
clc
global x V
x=[0;0];
tol = 1.0e-4;h = 0.5;
if size(x,2) > 1; x = x'; end  % x must be column vector
n = length(x);    % Number of design variables
df = zeros(n,1);  % Decreases of f stored here
u = eye(n);       % Columns of u store search directions V
fprintf('       xmin                    fmin   \n');
fprintf('   -------------               --------- \n');
for j = 1:30    % Allow up to 30 cycles
   xOld = x;
   fOld = feval(@myfun2,xOld);
   % First n line searches record the decrease of f
   for i = 1:n
      V = u(1:n,i);
      [a,b] = goldBracket(@fLine,0.0,h);
      [s,fmin] = goldensearch(@fLine,a,b);
      df(i) = fOld - fmin;
      fOld = fmin;
```

```
    x = x + s*V;
end
% Last line search in the cycle
V = x - xOld;
[a,b] = goldBracket(@fLine,0.0,h);
[s,fmin] = goldensearch(@fLine,a,b);
x = x + s*V;
fprintf('   (%3.3f,%3.3f)         %2.15f       \n',x, fmin);
if sqrt(dot(x-xOld,x-xOld)/n) < tol
    y = x; break
end
% Identify biggest decrease of f & update search
% directions
iMax = 1; dfMax = df(1);
for i = 2:n
    if df(i) > dfMax
        iMax = i; dfMax = df(i);
    end
end
for i = iMax:n-1
    u(1:n,i) = u(1:n,i+1);
```

```
    end
    u(1:n,n) = V;
end
fprintf('The minimum point (%2.3f,%2.3f)is reached at the %3dth cycle.\n',y,j)
    function z = fLine(s) % f in the search direction V
        global x V
        z = feval(@myfun2,x+s*V);
```

for the **example** on page **55** we have the following.

```
function y = myfun2(x)
%
%
y = x(1)-x(2)+2*(x(1)).^2+2*x(1)*x(2)+(x(2)).^2;
```

Reference

[1] Basak Aktek, Derivative Free Optimization Methods: Application in Stirrer Configuration and data clustering, M.Sc Thesis, the Middle East Technical Universitye, July, 2005.

[2] Igor Griva, Stephen G.Nash, Ariela Sofer, Linear and Nonlinear Optimization second edition George Mason University Fairfax, Virginia 2009.

[3] Jaan Kiusalaas, Numerical Methods in engineering with matlab, Second Edition, Cambride university printing press, 2010.

[4] Jorge J. More and Stefan M.Wild: Benchmarking Derivative Free Optimization Algorithms. Preprint ANL/MCS-P1471-1207 December 2007.

[5] Jorge Necedah and Stephen J. W. Wright: Numerical methods in optimization. Springer-Verlag New York, Inc. 1999. 2^{nd} Edition.

[6] Katya Sheinberg, Derivative Free Optimization method, CS 4/6-TE3, SEW ENG 4/6-TE3, Tamas Terlaky,IBM Watson Research Center

[7] Melissa Weber Men donça, M ultilevel O ptimization: C onvergence T heory, Algorithms a nd A pplication to D erivative-Free O ptimization, Ph.D T hesis, Facultés Universitaires Notre-Dame de la Paix Faculté des Sciences rue de Bruxelles, 61, B-5000 Namur, Belgium, 2009

[8] Mokhtar S. Bazaraa, Hanit D. Sherali C.M. Shetty; Nonlinear programming: Theory and algorithms, 2^{nd} edition.

[9] Singiresu S. Rao Engineering Optimization Theory and Practice, by John Wiley & Sons, Fourth Edition, Copyright, 2009.

[10] Wenyu Sun, Optimization Theory and Methods Nonlinear Programming, Nanjing Normal U niversity, N anjing, C hina YA-XIANG Y UAN Chinese A cademy o f S cience, Beijing, China Springer Science+Business Media, LLC, 2006.