
# *TRUST REGION NEWTON METHOD WITH TWO DIMENSIONAL SUBSPACE MINIMIZATION*

Ageze Abye Admasu

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE
DEPARTMENT OF MATHEMATICS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT OF THE DEGREE OF MASTER OF SCIENCE IN MATHEMATICS (OPTIMIZATION)

Advisor: Berhanu Guta(PhD.)

May, 2018

**Addis Ababa, Ethiopia**

ADDIS ABABA UNIVERSITY
DEPARTMENT OF MATHEMATICS

The undersigned hereby certify that they have read and recommend to the department of mathematics for acceptance of this thesis entitled **"Trust Region Newton Method with Subspace Minimization"** by **Ageze Abye** in partial fulfillment of the requirements for the degree of Master of Science in Mathematics (optimization).

Advisor: Dr. Berhanu Guta

Signature: _____

Date _____

Examiner 1: Dr. Tadesse Bekeshie

Signature: _____

Date: June 13/2018

Examiner 2: Dr. Hunduma Legesse

Signature: _____

Date June 18, 2018

ADDIS ABABA UNIVERSITY
DEPARTMENT OF MATHEMATICS

**Author**: Ageze Abye

**Title**: **Trust Region Newton Method with Subspace Minimization**

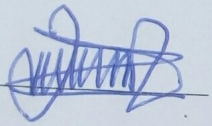
**Department**: Mathematics

**Degree**: M.Sc.

**Convocation**: May

**Year**: 2018



Ageze Abye:
Signature: 

Date June 18/2018

# Contents

# List of Figures

# List of Tables

# Acknowledgement

First of all, I would like to express my deep gratitude to my advisor Dr Berhanu Guta for his consistent guidance and help through out my study time next to God. Secondly, I would like to thank Addis Ababa University especially mathematics department staffs who have a crucial role for the completion of my study. Thirdly, my compliment goes to Woldia University, sponsoring me for Msc study. The last but not the least compliment is to my families for their persistent help.

# Abstract

The trust region method, minimization of a real valued function $f$ by using its quadratic model subject to Euclidean norm trust region constraint, occurs in many trust region algorithms. In most cases, it is inexpensive and even unnecessary to find an exact solution of a trust region problem unless the number of variables is relatively small. In order to alleviate this difficulty, in this thesis, we emphasized on a trust region method that takes Hessian of the model to be the true Hessian of a twice continuously differentiable real valued objective function $f$ on $\mathbb{R}^n$ , at a given current iterate point $x_k$ using the so called **two dimensional subspace minimization** strategy. As the name indicates, the two dimensional subspace minimization method defines an approximate minimizer $s$ to lie in a subspace $S_k$ of $\mathbb{R}^n$ spanned by two reasonably chosen directions. We constructed such subspace using Lanczos iterative method. This subspace method first reduces the $n$-dimensional trust region problem to 2-dimensional constrained problem, and then to finding roots of a fourth degree polynomial with the help of Lagrangian approach for constrained optimization problems. We compared the performance of the two dimensional subspace minimization method with that of an alternative trust region method, namely, the dogleg method, and the other common methods such as the steepest descent and Newton's method using **MATLAB** and we obtain that the two dimensional subspace minimization method is more efficient.

# Introduction

We consider the following unconstrained optimization problem.

$$\min\ f(x) \quad s.t \quad x \in \mathbb{R}^n$$

where $f : \mathbb{R}^n \to \mathbb{R}$ be at least twice continuously differentiable function. It may not be possible to solve this problem directly due to many reasons such as difficulty in solving zeros of the gradient of $f$ and hence a numerical method is used instead. In numerical optimization there are mainly two iterative procedures, namely, line search method and trust region method. Since both of these approaches are iterative, we need an initial guess $x_k$ as current approximate minimizer.In line search method, to approximate minimizer of $f$,at each iteration $k$, we seek a nonzero vector $d_k$, called a descent direction of $f$ at $x_k$ , along which $f$ is decreasing. Then, we use this vector to determine a nonnegative scalar $t_k$, called step length by solving(approximately) the one dimensional problem

$$\min \quad f(x_k + t_k d_k) \quad s.t \quad t \geq 0$$

to locate next iterate point $x_{k+1}$ using the scheme $x_{k+1} = x_k + t_k d_k$. There are many line search methods. Some of them are steepest descent method, Newton's method and quasi-Newton method.

Newton's method is **fast** and globally convergent only if the initial guess is close to the exact minimizer unless the objective function $f$ is convex. This is because the Hessian matrices of $f$ may not be positive definite and so the Newton point may not be available or it may not be a descent direction. Thus, this method has only **local convergence** property. However, the steepest descent method has global convergence property with **slow** convergence rate **[3],[5],[7]**. In order to **mitigate** this difficulties, we focus on the second approach, trust region method. A trust region method approximates $f$ by some simpler function usually quadratic function, an approximation to the second order Taylor representation of $f$ at a given initial guess $x_k$,called model of $f$ at $x_k$ denoted by $m_k$ , and (approximately) solve the model in some neighbourhood of this guess.Such constrained quadratic problems are referred to as **trust region subproblems(TRS)**. If the matrix in the quadratic term of $m_k$ is taken to be the true Hessian of $f$, we call it the Newton trust region method **[1],[5],[3]**.As we have said, a trust region method is not only to replace line search to get global convergence but also circumvents the difficulty caused by non-positive definite Hessian matrices in line search. The $k^{th}$ iteration of a trust region Newton method for the above unconstrained minimization involves finding an approximate solution of the trust region subproblem:

$$\min_{s \in \mathbb{R}^n} m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T H_k s \quad s.t \quad \|s\| \leq \Delta_k, \tag{1}$$

where $m_k(s)$ is the quadratic model of the objective function $f$ to be minimized , $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, gradient vector of $f$ at $x_k$ , $H_k = \nabla^2 f(x_k)$, the Hessian matrix of $f$ at $x_k$ , $\Delta_k$ is a given positive trust region radius, and $\|.\|$ is the Euclidean norm.

Trust region methods find approximate minimizer $s_k$ of (1) sequentially and use this approximate minimizer as trial step to move to next iterate point. Thus, in trust

region algorithm, the main source of computation effort other than the function evaluations required, is on the problem (1) to determine the step from the current iterate **[1],[3],[5],[8]**.

There are different trust region methods such as dogleg method, two dimensional subspace minimization method, Steihaug's method etc. Trust region algorithms are differ in their strategies for approximately solving (1). The dogleg method rely on an improvement of approximate solution of (1) along the steepest descent direction, called the Cauchy point **[5],[8]**. This method is designed that an approximate solution to (1) to be the step s, with $\|s\| \leq \Delta_k$ , on a piecewise linear curve passing through the origin, the unconstrained Cauchy point, $s_k^c = \frac{-\|g_k\|^2}{g_k^T H_k g_k} g_k$ and the Newton point , $s_k^N = -H_k^{-1} g_k$ **[5],[6],[7],[8]**.The dogleg method is appropriate when the objective function $f$ is convex(that is, the Hessian $H_k$ is always positive semidefinite) **[1],[3],[5]**. This motivates us to consider other techniques for finding approximate solution of (1) which include the directions of negative curvature(that is, the directions $d_k$ for which $d_k^T H_k d_k < 0$) in the space of candidate trust region steps to handel the indefinite case, and still produce an improvement on the Cauchy point for convergence. The so called **two dimensional(2D) subspace minimization** strategy is designed with these properties **[2],[5]**.

The **purpose** of this thesis is to solve the above unconstrained optimization problem by Newton trust region approach with two dimensional subspace minimization method which mitigates the aforementioned **drawbacks** of Newton's and the steepest descent methods. So, we use the two dimensional subspace minimization method to approximately solving each subproblem. The subspace of this method is spanned by two reasonably chosen directions, namely, the steepest descent direction(or the unconstrained minimizer of the quadratic model) and the Newton direction(in case $H_k$ is positive definite) or the modified Newton direction $-(H_k + \alpha_k I)^{-1} g$ or the direction of negative curvature (to handle indefinite matrices $H_k$). Therefore, the two dimensional subspace minimization strategy is an extension of the dogleg method [5]. Such an $\alpha$ is determined using **Lanczos** method to construct a subspace of the method. This directions are chosen for the seek of a global and fast convergence property **[2],[4],[5],[7]**.

The use of two dimensional subspace approach converts the $n$-dimensional subproblem (1) to a 2-dimensional constrained problem so that the cost of computation is very small relative to the original problem for large n. Then, Lagrangian approach aids to reduce this 2-dimensional subproblem to finding roots of a $fourth$ degree polynomial **[5],[6],[8]**. We compare the performance of the 2D subspace minimization method with that of an alternative method, namely the dogleg method, and the other basic methods such as the steepest descent and Newton's method.

The remaining part of this project is organized in the following way. Section 1 deals about basic concepts(preliminaries), mainly, general trust region methods and some related concepts, section 2 (the main body of this paper) focuses on the subspace minimization technique for Newton trust region step, and the last part(section 3) shows numerical implementations of some selected problems using **MATLAB** showing that the method achieves its goal efficiently provided that the parameters used are chosen appropriately.

# Chapter 1

# Preliminaries

In this section, we will see basic definitions, terminologies and ideas that are important for the consistency of this project.

## 1.1 Definitions and Terminologies

Even though there are different norms on $\mathbb{R}^n$, we use the Euclidean norm defined below throughout this paper.

**Definition 1.1.** *Let $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$. Then,*

*i.* $\|x\| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$.

*ii.* $\|A\| = \sup\{\|Ax\| : x \in \mathbb{R}^n \quad and \quad \|x\| = 1\}$.

Thus, we have

$$\|Ax\| \leq \|A\|\|x\| \quad and \quad \|AB\| \leq \|A\|\|B\|, \forall x \in \mathbb{R}^n, A, B \in \mathbb{R}^{n\times n}.$$

**Definition 1.2.** *Let $A = \begin{pmatrix} v_1 & v_2 & . & . & . & v_n \end{pmatrix}$, where $v_i \in \mathbb{R}^n, \forall i = 1, 2, ..., n$. Then, the set $S = span\{v_1, v_2, ..., v_n\}$ is called the column space or range of $A$.*

**Definition 1.3.** *Let $x, y \in \mathbb{R}^n$ and $S = \{v_1, v_2, ..., v_n\}$ be a set in $\mathbb{R}^n$. Then,*

*a. $x$ and $y$ are orthogonal if $x^T y = 0$.*

*b. $x$ is orthogonal to $S$ if $x^T v_i = 0$ for each $i = 1, 2, .., n$.*

*c. $S$ is orthogonal if $v_i^T v_j = 0$ for all $i \neq j$.*

*d. $S$ is orthogonal if it is orthogonal and $\|v_i\| = 1$ for each $i = 1, 2, .., n$.*

The following remark has a vital role in characterizing eigenvalues and eigenvectors, and factorization of symmetric matrices.

**Remark 1.1.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric. Then,*

*a. All eigenvalues of $A$ are real.*

b. *Eigenvectors corresponding to distinct eigenvalues are orthogonal.*

c. *A is orthogonally diagonalizable.That is,$A = Q^T DQ$, where $D$ is a diagonal matrix of eigenvalues of $A$ and $Q$ is an orthogonal matrix(that is,$Q^T = Q^{-1}$) of order $n$ whose columns are the corresponding eigenvectors. Such factorization of $A$ is said to be its eigendecomposion or spectral decomposition.*

**Theorem 1.1.** *(Cauchy–Schwartz inequality)*
*For any vectors $x$ an $y$ in $\mathbb{R}^n$*

$$x^T y \leq \|x\|\|y\|.$$

The common vectors and matrices in optimization are defined below.

**Definition 1.4.** *(Gradient vector and Hessian matrix)*
*Let $f : \mathbb{R}^n \to \mathbb{R}$ be at least twice continuously differentiable function. Then, for any given vector $x_k \in \mathbb{R}^n$, the gradient vector and Hessian matrix of $f$ at $x_k$ denoted by $\nabla f(x_k)$ and $\nabla^2 f(x_k)$ respectively are defined as:*

$$\nabla f(x_k) = \begin{pmatrix} \frac{\partial f(x_k)}{\partial x_1} \\ \frac{\partial f(x_k)}{\partial x_2} \\ . \\ . \\ . \\ \frac{\partial f(x_k)}{\partial x_n} \end{pmatrix}$$

*and*

$$\nabla^2 f(x_k) = \begin{pmatrix} f_{11}(x_k) & f_{12}(x_k) & . & . & . & f_{1n}(x_k) \\ f_{21}(x_k) & f_{22}(x_k) & . & . & . & f_{2n}(x_k) \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ f_{n1}(x_k) & f_{n2}(x_k) & . & . & . & f_{nn}(x_k) \end{pmatrix}$$

*where $f_{ij}(x_k) = \frac{\partial^2 f(x_k)}{\partial x_i \partial x_j}$, for $i = 1, 2, .., n$ and $j = 1, 2, .., n$.*

**Remark 1.2.** *Since $H_k = \nabla^2 f(x_k)$ is symmetric, it is orthogonally diagonalizable. That is, for each $k$, there exist an orthogonal matrix $Q_k$ and a diagonal matrix $D_k$ such that $H_k = Q_k^T D_k Q_k$, where the diagonal elements of $D_k$ are eigenvalues of $H_k$ and columns of $Q_k$ are the corresponding eigenvectors. Moreover, we can orthonormalize the set of columns $Q_k$ to form an orthonormal basis for some subspace of $\mathbb{R}^n$.*

**Definition 1.5.** *Let $A$ be an $n \times n$ matrix on $\mathbb{R}^n$. Then, $A$ is said to be*

i. *Positive definite(PD) if $x^T Ax > 0$ for any nonzero $x \in \mathbb{R}^n$.*

ii. *Positive semidefinite(PSD) if $x^T Ax \geq 0$ for any nonzero $x \in \mathbb{R}^n$.*

iii. *Negative definite(ND) if $x^T Ax < 0$ for any nonzero $x \in \mathbb{R}^n$.*

*iv. Negative semidefinite(NSD) if $x^T Ax \leq 0$ for any nonzero $x \in \mathbb{R}^n$.*

*v. Indefinite if it is neither PD, PSD, ND nor NSD.*

**Definition 1.6.** *All eigenvalues of PD, PSD, ND and NSD matrices are positive, non-negative, negative and none-positive respectively.*

## 1.2 Optimization Problems And Optimality Conditions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be at least twice continuously differentiable function. Since maximization problem is equivalent to minimization, we can consider any of one, minimization, in this project.

- The general form of Unconstrained problem is:

$$min \quad f(x) \quad s.t \quad x \in \mathbb{R}^n.$$

- The general form of constrained problem is:

$$min \quad f(x) \quad s.t \quad x \in S \subseteq \mathbb{R}^n,$$

where the feasible set $S = \{x \in \mathbb{R}^n : g_i(x) \leq 0 \quad and \quad h_j(x) = 0\}$ for $i \in I = \{1, 2, .., m\}, j \in J = \{1, 2, ..., p\}$ and $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$ are all differentiable functions.

**Remark 1.3.** *A point $x^* \in S$ is called a **global** minimizer of f on S if $f(x^*) \leq f(x), \forall x \in S$, and it is a **local** minimizer of f on S if there exists $\epsilon > 0$ such that $f(x^*) \leq f(x), \forall x \in N_\epsilon(x^*) \bigcap S$, where $N_\epsilon(x^*) = \{x \in \mathbb{R}^n : \|x - x^*\| < \epsilon\}$, called $\epsilon$-neighbourhood of $x^*$.*

**Optimality Conditions**

i. First order necessary condition: If f attains its local/global minima at $x^*$, then $\nabla f(x^*) = 0$.

ii. Second order necessary condition: If $x^*$ is a local/global minimizer of f on $\mathbb{R}^n$, then $\nabla f(x^*) = 0 \quad and \quad \nabla^2 f(x^*)$ is **PSD**.

iii. Second order sufficient condition: If $\nabla f(x^*) = 0 \quad and \quad \nabla^2 f(x^*)$ is **PD**, then $x^*$ is a local minimizer of f.

iv. Karush-Kuhn-Tucker (KKT) Conditions: If $x^*$ is a local/global optimal solution of

$$\min \quad f(x) \quad s.t \quad g_i(x) \leq 0, i = 1, 2, .., m$$

, then the following KKT conditions should hold:

1. $\nabla f(x^*) + \sum_{i=1}^m \lambda_i \nabla g_i(x^*) = 0$, for some $\lambda_i \geq 0, i = 1, ..., m$. This is called dual feasibility.
2. $\lambda_i g_i(x^*) = 0, i = 1, 2, ..., m$. It is called complementary slackness.
3. $g_i(x^*) \leq 0, i = 1, 2, ..., m$. It is primal feasibility.

### 1.2.1 Convexity and Minimization Problems

**Definition 1.7.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be **convex** if for any $x, y \in \mathbb{R}^n$ and any $\lambda \in [0, 1]$,*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

**Remark 1.4.** *i) Let $f$ be twice continuously differentiable on $\mathbb{R}^n$. The Hessian matrix of $f$ is positive semi definite at each $x \in \mathbb{R}^n$ if and only if $f$ is convex on $\mathbb{R}^n$.*

*ii) An optimization problem whose objective function and its constraint set are convex is convex.*

*iii) Every local minimizer of a convex problem is global.*

## 1.3 Lagrangian Approach For Constrained Problems

We can change a constrained problem in to an equivalent unconstrained problem. It is performed by adding a function $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$ such that $f_0(x) = 0$, for each feasible point x on to the objective function to be minimized. Then, the resulted function is called **Lagrangian** for the problem.

### 1.3.1 Lagrangian Method For equality Constraints

Consider the problem

$$\min \quad f(x) \quad s.t \quad x \in S \tag{1.1}$$

, where $S := \{x \in \mathbb{R}^n : h_j(x) = 0, j = 1, 2, ..., p\}$. Then,we have $f_0(x) = \sum_{j=1}^{p} \mu_j h_j(x) = 0$, for any $\mu_j \in \mathbb{R}$. Therefore, the Lagrangian for (1.1) is

$$L(x, \mu) = f(x) + \sum_{j=1}^{p} \mu_j h_j(x),$$

where $\mu = (\mu_1, ..., \mu_p)^T \in \mathbb{R}^p$ called Lagrange multiplier. The corresponding Lagrange problem is

$$\min \quad L(x, \mu) \quad s.t \quad x \in \mathbb{R}^n, \forall \mu \in \mathbb{R}^p \tag{1.2}$$

**Remark 1.5.** *If $x^*$ is an optimal solution of (1.2) and $x^* \in S$ for some $\mu^* \in \mathbb{R}^p$, then $x^*$ is optimal solution of (1.1). Thus, an optimal solution $x^*$ of (1.1) should satisfy $\nabla L(x^*) = 0$ and $x^* \in S$.*

### 1.3.2 Lagrangian Method For Inequality Constraints

We consider the problem

$$\min \quad f(x) \quad s.t \quad x \in S \tag{1.3}$$

, where $S := \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, 2, ..., m\}$. We have the same transformation as above but the Lagrange multipliers $\lambda_1, ..., \lambda_m$ are all nonnegative.

## 1.4 Rate Of Convergence Of Sequences

Rate of convergence measures how fast a sequence does converge.

**Definition 1.8.** *Let $x_k$ be a sequence in $\mathbb{R}^n$ that converges to $x^*$. The rate of convergence is said to be*

a. *Linear if there is a constant $r \in (0,1)$ and $k_0 \geq 0$ such that $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} \leq r, \forall k \geq k_0$.*

b. *Super linear if $\lim_{k\to\infty} \frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} = 0$.*

c. *Quadratic if there exist constants $M > 0$ and $k_0 \geq 0$ such that $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^2} \leq M, \forall k \geq k_0$.*

Let $f : \mathbb{R}^n \to \mathbb{R}$ be at least twice continuously differentiable function. In the study of unconstrained optimization, we seek to solve a problem of the form:

$$min \quad f(x) \quad s.t \quad x \in \mathbb{R}^n \tag{1.4}$$

Solving (1.4) directly may not be always possible due to various reasons. To overcome this difficulty, we need to study numerical optimization which solves it numerically. The iterative procedures that we use have to be convergent.

**Definition 1.9.** *An iterative method is said to be*

i. *locally convergent if the sequence of iterates generated by the method converges to a solution when the initial approximation is already close enough to the solution.*

ii. *globally convergent if the sequence of iterates generated by the method converges for any initial approximation.*

In numerical optimization there are mainly two iterative procedures to find an approximate minimizer of (1.4). These are the line search and trust region methods.

## 1.5 Line Search method

- In line search methods to determine a step length is to is find a minimizer of a univariate function along a line.

**Definition 1.10.** *Consider a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and a point $x_k \in dom(f)$.*
*A nonzero vector $d_k \in \mathbb{R}^n$ is called a **descent direction** of $f$ at $x_k$ if there exists a $\delta > 0$ such that*

$$f(x_k + td_k) < f(x_k), \forall t \in (0, \delta)$$

**Theorem 1.2.** *Let $\nabla f(x_k)$ be the gradient of $f$ at $x_k$.*
*If $(\nabla f(x_k))^T d_k < 0$, then $d_k$ is a descent direction of $f$ at $x_k$.*

**Remark 1.6.** *i. If $\nabla f(x_k) \neq 0$, then $d_k = -\nabla f(x_k)$ is called the **steepest descent direction** of $f$ at $x_k$.*

*ii. Most numerical methods for minimization of $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ constructs sequence of iterate points using the scheme $x_{k+1} = x_k + t_k d_k$, where $t_k$(called the step length) is obtained by a line search along $d_k$ that minimizes $\phi(t) = f(x_k + td_k)$ on $0 \leq t \leq b$, for some $b > 0$ at each iterate $x_k$. The step length may an optimal solution to our univariate function and is called the exact step length or else it satisfies some what smooth conditions only and is called inexact step length. The conditions that our inexact step length required to satisfy are called **Wolfe** conditions. These conditions are*

$W_1$. $f(x_k + t_k d_k) \leq f(x_k) + c_1 t_k \nabla f(x_k)^T d_k$, *for some* $0 < c_1 < 1$.

$W_2$. $\nabla f(x_k + t_k d_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k$, *for some* $c_1 < c_2 < 1$.

**The Generic Idea of Line Search Method**

Given an iterate $x_k$ as current approximate solution to (1.1) . In line search method, we find a descent direction $d_k$ of $f$ at $x_k$ and a step length $t_k$, a minimizer(approximate) of the one dimensional problem:

$$min \quad f(x_k + td_k) \quad s.t \quad t \geq 0$$

to locate the next iterate $x_{k+1}$ by the scheme $x_{k+1} = x_k + t_k d_k$.

There are a number of line search methods. A certain numerical method need to have fast global convergence whenever possible. Thus, we focus on the steepest descent and Newton's methods having global and fast local convergence respectively **[3]**,**[5]**'**[9]**.

### 1.5.1 The Steepest Descent Method

The steepest descent method is a line search method in which the search direction at $x_k$ is given by $d_k = -\nabla f(x_k)$. This method has global convergence property, but its convergence rate is slow(linear) due to its zigzaging nature, that is, $\nabla f(x_{k+1})^T d_k = 0$ for each k **[5]**. We have the following algorithm to solve (1.4).

**Algorithm 1.1.** ***(Steepest descent):***
***Initial step:*** *Choose small error tolerance, $\epsilon > 0$ and initial guess $x_1$. Let $k = 1$.*
***Main step:*** *While $\|\nabla f(x_k)\| > \epsilon$*
*Let $d_k = -\nabla f(x_k)$. Find(approximate) $t_k$ that minimizes $\phi(t) = f(x_k + td_k), t \geq 0$.*
*$x_{k+1} = x_k + t_k d_k$.*
*end*

### 1.5.2 Newton's Method

Newton's method is a basic tool in numerical optimization. In its original form, this method is destined for solving equations. However, it can be easily tailored for unconstrained optimizations also.

## Newton's Method For Minimization

We can use Newton's method to approximate minimizer of one as well as multidimensional problems. Newton's method uses the Taylor's approximation to approximately solve the targeted optimization problem.

**Theorem 1.3.** *(Taylor's Theorem)*
*Suppose that $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuously differentiable and $s \in \mathbb{R}^n$. Then we have*

$$f(x+s) = f(x) + g(x+ts)^T s,$$

*for some $t \in (0,1)$, where $g$ is the gradient of $f$. Moreover, if $f$ is twice continuously differentiable, we have that*

$$g(x+s) = g(x) + \int_0^1 H(x+ts)p dt,$$

*and that*

$$f(x+s) = f(x) + g(x)^T s + \frac{1}{2} s^T H(x+ts)s,$$

*where $t \in (0,1)$ and $H$ is the Hessian of $f$.*

## One Dimensional Newton's Method

Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be at least twice continuously differentiable function. Newton's method starts from a suitable initial guess $t_0$ and it takes the next iterate point $t_1$ to be the minimizer of the **quadratic approximation** of $f$ at $t_0$,
$q(t) = f(t_0) + f^{'}(t_0)(t-t_0) + \frac{1}{2} f^{''}(t_0)(t-t_0)^2$. That is,

$$t_1 = t_0 - \frac{f^{'}(t_0)}{f^{''}(t_0)}.$$

Subsequent iterations are determined in similar manner and we have the scheme:

$$t_{k+1} = t_k - \frac{f^{'}(t_k)}{f^{''}(t_k)},$$

until $|f(t_k)| < \epsilon$ or $|t_{k+1} - t_k| < \epsilon$ for some sufficiently small error tolerance $\epsilon$.

**Remark 1.7.** *Newton's method is applicable whenever*

a. $f^{''}(t_k) \neq 0, \forall t_k$.

b. $f^{''}(t_k) > 0, \forall t_k$*(i.e $f$ is strictly convex).*

The first limitation is alleviated by using the approximation $f^{''}(t_k) \approx \frac{f^{'}(t_k) - f^{'}(t_{k-1})}{t_k - t_{k-1}}$, called the secant method and the second by using $|f^{''}(t_k)|$ instead of $f^{''}(t_k)$, called one dimensional modified Newton's method.

## Multidimensional Newton's Method

Newton's Method for multidimensional minimization is a direct extension of its one dimensional version. That is, given a current iterate point $x_k \in \mathbb{R}^n$, the next iterate point $x_{k+1}$ is the point that minimizes the quadratic approximation of f around $x_k$. This approximating function is

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

So,we must have $\nabla q(x_{k+1}) = 0$. Thus, the sequence of iterates is generated by

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1}\nabla f(x_k),$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is at least twice continuously differentiable. Therefore, the Newton's method can be considered as a line search method with step direction $d_k^N = -(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$, called Newton's direction and step length $t_k = 1$, full Newton step length. However, $d_k^N$ is a descent direction only when the Hessian matrix is positive definite.

To find a minimum point of a twice differentiable function $f$ defined on $\mathbb{R}^n$, we use the following algorithm.

**Algorithm 1.2.** ***(Newton's method):***
***Input:*** $f, g = \nabla f, H = \nabla^2 f, \epsilon$ *(error tolerance)*

1. *Choose a suitable initial guess* $x_1 \in \mathbb{R}^n$. $k = 1$
2. *If* $\|g(x_k)\| < \epsilon$, *stop*($x^* = x_k$ *is an approximate minimizer). Else go to step (3)*
3. *Solve the system of linear equations* $H(x_k)x = -g(x_k)$
4. $x_{k+1} = x_k + x$. *Set* $k := k + 1$, *and repeat from step (2).*

**Definition 1.11.** *A function* $f : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ *is said to be Lipschitz continuous on* $D$ *if there exists the smallest constant* $L > 0$*(called Lipschitz constant) such that*

$$|f(x) - f(y)| \leq L\|x - y\|, \forall x, y \in D.$$

The convergence property of Newton's method is given below.

**Theorem 1.4.** *Suppose that* $f$ *is twice differentiable and that the Hessian* $H(x) = \nabla^2 f(x)$ *is Lipschitz continuous in a neighborhood of a solution* $x^*$ *at which the second order sufficient conditions are satisfied. Consider the iteration* $x_{k+1} = x_k - H_k^{-1}g_k$. *Then,*

*i. if the starting point* $x_0$ *is sufficiently close to* $x^*$, *then the sequence of iterates converges to* $x^*$*;*

*ii. the rate of convergence of* $\{x_k\}$ *is quadratic,and*

*iii. the sequence of gradient norms* $\{g_k\}$ *converges quadratically to zero.*

*Proof.* See[**5,pp 44-45**]. □

**Remark 1.8.** *Newton's method has a quadratic global convergence whenever $f$ is convex, otherwise the method is successful only when the initial guess $x_1$ is close to $x^*$.*

We can overcome the above limitation of Newton's method by doing the following

1. Use matrix factorization such as eigenvalue decomposition to approximate the Hessian matrix $H_k$ by a positive symmetric definite matrix $B_k$ so that $d_k = -B_k^{-1}g_k$ is a descent direction.

2. Make line search along $d_k$ to have $f(x_{k+1}) < f(x_k)$.

## 1.6 Trust Region Method

Trust region method is one of the numerical optimization methods in solving nonlinear programming problems. As we have said, solving (1.4) analytically may not be always possible due to various reasons such as complicated nature of the objective function $f$. The trust region method mitigates this difficulty by approximating $f$ to a simpler function(usually quadratic function) called model of $f$, at a given current iterate point $x_k$. This quadratic model is an approximation to the Taylor representation of $f$ at $x_k$ and is denoted by $m_k$. That is,

$$m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T B_k s$$

, where $B_k$ is symmetric.

In Taylor approximation theory, an approximation to a given function at a given point is **adequate** in some neighbourhood of that point. In order to preserve this property of Taylor approximation, the trust region method fixes a region around the current iterate $x_k$ on which the model is trusted to be an adequate representation(approximation) of $f$. This region, called "trust-region", is fixed by choosing a positive number $\Delta_k$, called trust region radius, and the method minimizes the quadratic model $m_k$ over a ball of radius $\Delta_k$, $\Omega = \{s \in \mathbb{R}^n : \|s\| \leq \Delta_k\}$, where $\|.\|$ is the Euclidean norm.

Once approximating the objective function $f$ by its quadratic model $m_k$, and fixing the trust region $\Omega$, our optimization problem is essentially reduced to solving a sequence of trust region subproblems in which both the objective function and the constraint are **quadratic**. That is,

$$\min_{s \in \mathbb{R}^n} \; m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T B_k s \quad s.t \quad \|s\| \leq \Delta_k \tag{1.5}$$

, where $f_k = f(x_k)$, $B_k$ is the Hessian or its approximation, and the rest quantities are as above. Thus, trust region method aims to find an approximate solution of (1.5) which requires a suitable choice of the trust region radius $\Delta_k$ .

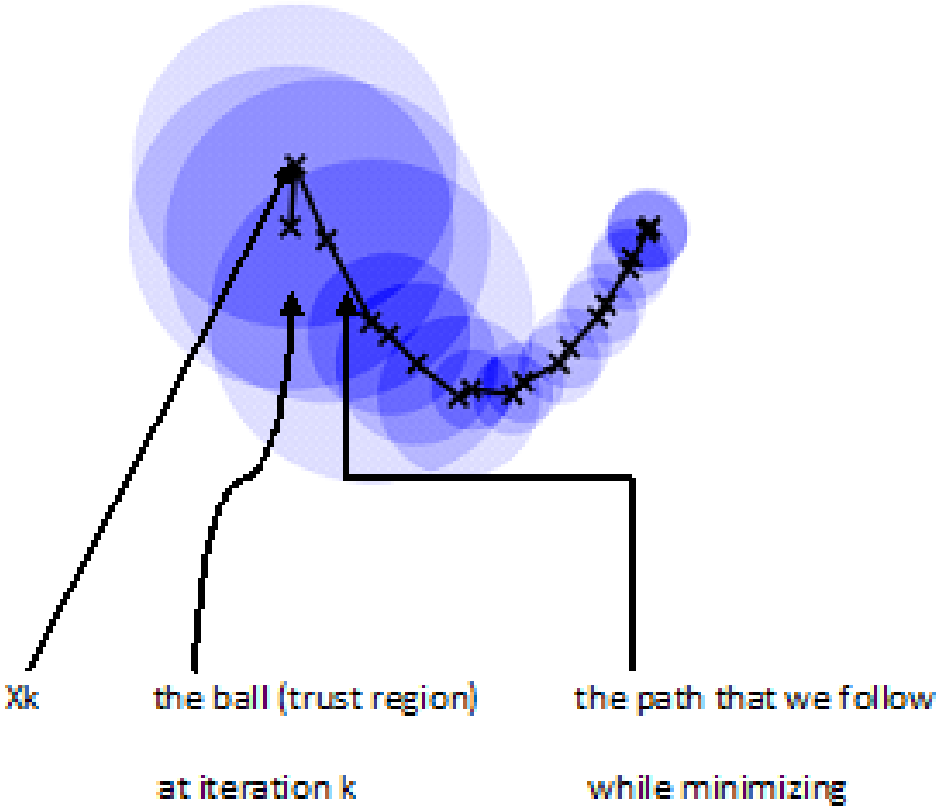



Figure 1.1: Trace of unconstrained optimization with trust region method

### 1.6.1 Choice Of The trust Region Radius,$\Delta_k$

The size of the trust region is critical to the effectiveness of each step.The direction of the step changes whenever the size of the trust region is altered. Since the size of the trust region is a determinant factor for agreement between $m_k(s) \quad and \quad f(x_k+s)$ , a suitable choice of $\Delta_k$ has to be done.

If the trust region is too small, the algorithm misses an opportunity to take a substantial step that will move it much closer to the minimizer of the objective function. If too large, the minimizer of the model may be far from the minimizer of the objective function in the region. Thus, in the former case we may enlarge the region to include those missed steps, and we may reduce the trust region and try again in the later case. In other words, if $\Delta_k$ is too small, there is well agreement between $m_k(s) \quad and \quad f(x_k+s)$ but this may hinder the progress of iterates. This is an indication for enlarging the region at the next iterate. If $\Delta_k$ is too large, then $m_k(s) \quad and \quad f(x_k+s)$ may not agree and the minimizer of $m_k(s)$ over the region can fail to produce acceptable next iterate **[1],[3],[5]**.

Given a step $s_k$, agreement between the model function $m_k$ and the objective function $f$ is determined in terms of the ratio defined by:

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)}; \tag{1.6}$$

the numerator is called the actual reduction(ared), and the denominator is the predicted reduction(pred) (that is, the reduction in $f$ predicted by the model function).

As the step $s_k$ is obtained by minimizing the model $m_k$ over a region that includes $s = 0$, the pred will always be nonnegative. This shows that the sign of $\rho_k$ is determined using the sign of $f(x_k) - f(x_k + s_k)$. Hence, if $\rho_k$ is negative, the new objective value $f(x_k + s_k)$ is greater than the current value $f(x_k)$, so the step must be rejected. On the other hand, if $\rho_k$ is close to 1, there is good agreement between the model $m_k$ and the function $f$ over this step, so it is safe to expand the trust region for the next iteration. If $\rho_k$ is positive but significantly smaller than 1, we do not alter the trust region, but if it is close to zero or negative, we shrink the trust region by reducing $\Delta_k$ at the next iteration. Thus, at each iteration we have to determine the value of the ratio, $\rho_k$ in order to update the trust region radius and decide the acceptance or rejection of the step $s_k$. Trust region methods to approximately solve the subproblems at each iteration differ in the way of determining the step $s_k$. However,they use the following algorithm to update their trust region radius and to determine the acceptance of the step.

### 1.6.2 Trust Region Algorithm

Apart from the question how the step(approximate minimizer) of the trust region subproblem is determined, the **generic** trust region algorithm is given as follow.

**Algorithm 1.3.** *(**Generic trust region algorithm**):*
*Given: initial guess($x_0$),* $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, $\eta \in [0, \frac{1}{4})$,
$tol > 0$*(suff small error tolerance),*$k := 0$.

$$
\begin{aligned}
&\textbf{\textit{while}} \ \ \|\nabla f(x_k)\| > tol \\
&Obtain \ \ s_k \ \ by(approximately) \ \ solving \ \ (1.5) \\
&Evaluate \ \ \rho_k \ \ from \ \ (1.6) \\
&if \ \ \rho_k < \frac{1}{4}, then \\
&\Delta_{k+1} = \frac{1}{4}\Delta_k \\
&else \\
&if \ \ \rho_k > \frac{3}{4} \ \ and \ \ \|s_k\| = \Delta_k, then \\
&\Delta_{k+1} = min(2\Delta_k, \bar{\Delta}) \\
&else \\
&\Delta_{k+1} = \Delta_k \\
&end \quad \%if \\
&end \quad \%if
\end{aligned}
$$

$$
\begin{aligned}
&if \ \ \rho_k > \eta, then \\
&x_{k+1} = x_k + s_k \\
&else \\
&x_{k+1} = x_k \\
&end \quad \%if \\
&end \quad \%while
\end{aligned}
$$

The next important issue is methods for (approximately) solving the trust region subproblem (1.5) with convergence.

### 1.6.3 The Cauchy Point

Although in principle we seek the optimal solution of the subproblem (1.5), it is enough for purposes of global convergence to find an approximate solution $s_k$ that lies within the trust region and gives a sufficient reduction in the model.

**Definition 1.12.** *The minimizer of the model function $m_k$ along the steepest descent direction $s_k^d = -g_k$ subject to the trust region bound is called the **Cauchy point**, denoted by $s_k^c$.*

As we have said above, we need a step in the trust region that results a sufficient reduction in the model $m_k$. We can quantify this *sufficient reduction* in terms of the Cauchy point. That is why we seek to have a focus about it.

**Procedures To Determine $s_k^c$**

The Cauchy step $s_k^c$ is inexpensive to calculate(no matrix factorizations are required) and has crucial importance in deciding if an approximate solution of the trust-region subproblem is acceptable. We follow two main steps to find it.

**Step1:** Determining a vector $s_k^b$ that minimizes the linear version of the model $m_k$ subject to the trust region bound. That is, $s_k^b = -\frac{\Delta_k}{\|g_k\|} g_k$

**Step2:** Finding a scalar $\tau_k$ that minimizes $m_k(\tau s_k^b)$ subject to the trust region bound, and then we set $s_k^c = \tau_k s_k^b$.

Thus,

$$\tau_k = \begin{cases} 1, & \text{if } g_k^T B_k g_k \leq 0 \\ min\{\frac{\|g_k\|^3}{\Delta_k g_k^T B_k g_k}, 1\}, & \text{otherwise.} \end{cases}$$

**Improvement On The Cauchy Point**

Although the Cauchy point $s_k^c$ provides sufficient reduction in the model function $m_k$ with global convergence, and it is easy to calculate, we usually need to improve it. The reason is that always taking the Cauchy point as our step is equivalent to the steepest descent method with a particular choice of step length **[4],[5],[6]**. The steepest descent method has slow rate of convergence even if the optimal step length is used and so we have to improve $s_k^c$ as much as possible.

There are various trust region methods that compute the Cauchy point and try to improve on it to find a better approximate solution of (1.5). However, this paper focuses on the **two dimensional subspace minimization** strategy for Newton trust region(see Chapter 2) which is a generalization of the so called Newton dogleg method. So, let us see the dogleg method for general trust region discussed above.

### 1.6.4 The Dogleg Method

The dogleg method is one of the earliest methods which originally proposed by Powell, to find an approximate solution of the trust region subproblem (1.5), and based on an improvement of the Cauchy point. In this section we drop the iteration subscript k for simplicity and denote $s^*$ to be the solution of (1.5).

This method is more effective whenever the Hessian B of the model function m is positive definite. If B is positive definite, then the unconstrained minimizer of the model m is $s^N = -B^{-1}g$ called the Newton point. Thus, we take it as an approximate solution $s^*$ of (1.5) whenever $s^N$ is feasible.

Consider the case $\|s^N\| > \Delta$. In this case if the Cauchy point $s^c$ reaches the boundary, then no more improvement on it and this method takes $s^*$ to be $s^c$. That is,

$$s^* = s^c$$

If $s^c$ is strictly inside the trust region, then this method made a technical improvement on it. Since $\Delta$ is small relative to $\|s^N\|$, the restriction $\|s\| \leq \Delta$ shows that the linear

approximation is effective. For such $\Delta$, we can get an approximation to $s^*$ by simply omitting the quadratic term from the model m. That is,

$$s^*(\Delta) = -\Delta \frac{g}{\|g\|} \quad when \quad \Delta \quad is \quad small.$$

For intermediate value of $\Delta$ , the solution $s^*$ typically follows a curved trajectory.

The dogleg method finds an approximate solution by replacing the curved trajectory for $s^*$ with a path consisting of two line segments. The first line segment runs from the origin to the minimizer of m along the steepest descent direction, which is

$$s^u = -\frac{g^T g}{g^T B g} g$$

, and the second line segment runs from $s^u$ to $s^N$. This curve is called the dogleg path. Mathematically, we can express it as:

$$s(\tau) = \begin{cases} \tau s^u, & 0 \leq \tau \leq 1 \\ s^u + (\tau - 1)(s^N - s^u), & 1 \leq \tau \leq 2. \end{cases}$$

The dogleg method chooses s to minimize the model m along this path, subject to the trust-region bound. The following theorem shows that the minimum along the dogleg path can be found easily.

**Theorem 1.5.** *Let B be positive definite. Then*

1. $\|s(\tau)\|$ *is an increasing function of* $\tau$ *,and*
2. $m(s(\tau))$ *is a decreasing function of* $\tau$*.*

*Proof.* (See [**5**]) □

Since the model $m$ is decreasing along the path $s(\tau)$, its intersection point with the trust region boundary called the dogleg point(DP), provides an improvement on $s^u$. Therefore,

$$s^* = DP = s^u + \alpha(s^N - s^u), \tag{1.7}$$

where $\alpha$ is a positive number satisfying the scalar quadratic equation:

$$\|s^u + (\alpha - 1)(s^N - s^u)\| = \Delta \tag{1.8}$$

That is,

$$\alpha = \frac{h_1 + \sqrt{h_1^2 + h_2^2 h_3}}{\|h_2\|^2},$$

where $h_1 = (s_k^c)^T(s_k^c - s_k^N), h_2 = \|s_k^c - s_k^N\|^2$ and $h_3 = \Delta_k^2 - \|s_k^c\|^2$.

**Algorithm 1.4.** ***(Newton-dogleg method:)***
*Given: initial guess*$(x_0)$, $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, $\eta \in [0, \frac{1}{4})$,
$tol > 0$*(suff small error tolerance),* $g_0, H_0(Hessian \ \ of \ \ f \ \ at \ \ x_0), k := 0.$

1. *If* $H_k$ *is not PD, we take* $s_k = s_k^c$.
2. *If* $H_k$ *is PD and* $\|s_k^B\| \leq \Delta_k$, *we take* $s_k = s_k^B$. *Otherwise goto step 3.*
3. *If* $\|s_k^B\| = \Delta_k$, *we take* $s_k = s_k^c$. *Otherwise goto step 4.*
4. *We solve* $\alpha$ *from (1.8) and improve* $s_k^c$ *using (1.7). Set* $k := k+1$ *until* $\|g_k\| \leq tol$.

**Remark 1.9.** *The dogleg method is appropriate when the objective function* $f$ *is convex(that is, the Hessian* $H_k$ *is always positive semidefinite).*

The above remark motivate us to use other techniques which are suitable for the general case. In the next chapter, we will see the two dimensional subspace method.

# Chapter 2

# Two Dimensional Subspace Minimization For Trust Region Newton Step

## 2.1 Newton Trust Region Method

In the previous chapter we have seen that trust region methods do not require the Hessian matrix $B_k$ of the model function to be positive definite but it is assumed to be symmetric. Since the objective function $f$ is assumed to be at least twice continuously differentiable, we can use the true Hessian $H_k = \nabla^2 f(x_k)$ of $f$ at current iterate $x_k$ instead of $B_k$ .
**Notation:** In the sequel we use

i. $s$ instead of $s_k$ to denote an approximate minimizer of the $k^{th}$ subproblem.

ii. $x_k$ and $x^*$ to denote approximate minimizer of $f$ at the $k^{th}$ iteration and optimal(exact) solution of $f$ respectively.

iii. $f_k$ to denote $f(x_k)$.

iv. $g_k$ to denote the gradient vector of $f$ at iterate point $x_k$.

v. $H_k$ to denote the Hessian matrix of $f$ at $x_k$.

vi. $m_k$ to denote the quadratic model of $f$.

**Definition 2.1.** *A Newton trust region method is a trust region method where the symmetric matrix $B_k$ in the quadratic model is taken to be exact Hessian $H_k = \nabla^2 f(x_k)$ at a given current iterate $x_k$, of the objective function $f$ to be minimized.*

From this definition we can understand that a Newton trust region method is special case of the general trust region method. Hence all concepts that hold for general trust region method also hold for it. This paper is targeted to use two dimensional subspace minimization technique to (approximately) solve a Newton trust region subproblem

$$\min_{s\in\mathbb{R}^n} \; m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T H_k s \quad s.t \quad \|s\| \leq \Delta_k, \tag{2.1}$$

Similarly, consider the following subproblem

$$\min_{s\in\mathbb{R}^n} \; m_k(s) = g_k^T s + \frac{1}{2} s^T H_k s \quad s.t \quad \|s\| \leq \Delta_k. \tag{2.2}$$

Since $f_k$ doesn't depend on $s$ for each iteration k, minimizing (2.1) is the same as minimizing (2.2). Thus, we can use (2.1) and (2.2) interchangeably while talking about their minimizer. The existence of a global solution for (2.1) is guaranteed by theorem (2.1) below.

**Remark 2.1.** *Since, for each k, quadratic model $m_k(s)$ is continuous and the trust region $\Omega = \{s \in \mathbb{R}^n : \|s\| \leq \Delta_k\}$ is a closed and bounded subset of $\mathbb{R}^n$ (and hence compact), (2.1) has a solution.*

Furthermore, if f is not a convex function, then the Hessian matrix $H_k$ of $f$ at $x_k$ may not be positive definite. Thus, the global solution of (2.1) may not exist.

The main purpose of this project is to obtain an approximate solution of (2.1) using the so called two dimensional subspace minimization strategy. The resulted solution of this method is assumed to has a good resemblance(good approximate) with the exact solution.

## 2.2 Characterization Of Exact Solutions

When the problem is relatively small(that is, n is not too large), it may be worthwhile to exploit the model more fully by looking for a closer approximate to the solution of the subproblem. In order to have a precise characterization of the solution of (2.1), let us see the following lemma which deals with the unconstrained minimizers of quadratics and its particular characterization in the case where the Hessian is positive semidefinite.

**Lemma 2.1.** *Let $m$ be a quadratic function defined by*

$$m(s) = g^T s + \frac{1}{2} s^T B s$$

*where $B$ is any symmetric matrix. Then*

*i. $m$ attains a minimum if and only if $B$ is PSD and $g$ is in the range of $B$.*

*ii. $m$ has a unique minimizer if and only if $B$ is positive definite.*

*iii. If $B$ is positive semidefinite, then every $s$ satisfying $Bs = -g$ is a global minimizer of $m$.*

*Proof.* i ($\Leftarrow$) : Since g is in the range of B, we can find s such that $Bs = -g$. Then for any $w \in \mathbb{R}^n$, we have

$$\begin{aligned} m(s+w) &= g^T(s+w) + \frac{1}{2}(s+w)^T B(s+w) \\ &= m(s) + g^T w + (Bs)^T w + \frac{1}{2} w^T B w \\ &= m(s) + \frac{1}{2} w^T B w \\ &\geq m(s) \quad (as \quad B \quad is \quad PSD) \end{aligned}$$

Thus, $s$ is a minimizer of m. Conversely, if $s$ is a minimizer of $m$, then by the $2^{nd}$ order necessary condition $\nabla m(s) = Bs + g = 0$(so g is in the range of B), and $\nabla^2 m(s) = B$ is PSD.

ii. ($\Leftarrow$) : Suppose B is PD. Then, B is invertible and hence we can find s such that $Bs = -g$. From $(i)$ above ,

$$m(s + w) > m(s), \forall w \neq 0.$$

Therefore, the minimizer s is unique. Conversely, assume that m has a unique minimizer say s. By $(i)$ B must be PSD. Claim that B is PD. If B is not PD, one can find $w \neq 0$ such that $Bw = 0$. Then, $m(s) = m(s + w)$, which contradicts the uniqueness of s.

iii. Follow from $(i)$.

□

**Remark 2.2.** *This lemma also holds for the specific case $B = H$, exact Hessian of $f$.*

Based on this lemma, the following theorem characterizes the solution of (2.1) precisely where the iteration subscript k is omitted for simplicity.

**Theorem 2.1.** *The vector $s^*$ is a global solution of the Newton trust region subproblem*

$$\min_{s \in \mathbb{R}^n} \quad m(s) = f + g^T s + \frac{1}{2} s^T H s \quad s.t \quad \|s\| \leq \Delta, \tag{2.3}$$

*if and only if $s^*$ is feasible and there is $\alpha \geq 0$ such that the following conditions are satisfied.*

*a)* $(H + \alpha I)s^* = -g$

*b)* $\alpha(\Delta - \|s^*\|) = 0$

*c)* $(H + \alpha I)$ *is positive semidefinite*

*Proof.* ($\Leftarrow$:) Suppose that $\|s^*\| \leq \Delta$ and there is $\alpha \geq 0$ such that $(a), (b), (c)$ are satisfied. Lemma 2.1$(iii)$, implies that $s^*$ is a global minimizer of the quadratic function

$$\bar{m}(s) = g^T s + \frac{1}{2} s^T (H + \alpha I) s = m(s) + \frac{\alpha}{2} s^T s.$$

Since $\bar{m}(s^*) \leq \bar{m}(s)$, we have

$$m(s^*) + \frac{\alpha}{2}(s^*)^T s^* \leq m(s) + \frac{\alpha}{2} s^T s.$$

That is

$$m(s) \geq m(s^*) + \frac{\alpha}{2}((s^*)^T s^* - s^T s).$$

Then, by $(b)$ $\alpha(\Delta^2 - (s^*)^T s^*) = 0$. Thus,$m(s) \geq m(s^*) + \frac{\alpha}{2}(\Delta^2 - s^T s)$. Therefore, as $\alpha \geq 0$, we conclude that $m(s^*) \leq m(s)$, for all $s$ such that $\|s\| < \Delta$, and hence $s^*$ is a global

minimizer as desired. For the converse, we assume that $s^*$ is a global solution and we need to find an $\alpha \geq 0$, satisfying $(a), (b), (c)$. If $\|s^*\| < \Delta$, by assumption $\nabla m(s^*) = Hs^* + g = 0$ and $\nabla^2 m(s^*) = H$ is PSD. Thus, $\alpha = 0$ satisfies $(a), (b), (c)$. Assume in the remainder of the proof $\|s^*\| = \Delta$. Then $(b)$ holds and $s^*$ minimizes m subject to $\|s\| = \Delta$. By applying optimality conditions for constrained optimization to this problem(see chapter 1), there exists an $\alpha$ such that the Lagrangian function $L(s, \alpha) = m(s) + \frac{\alpha}{2}(s^T s - \Delta^2)$ has a stationary point at $s^*$. That is,

$$\frac{\partial L}{\partial s}(s^*) = 0 \Rightarrow (H + \alpha I)s^* = -g.$$

Thus,(a) holds. Since $m(s) \geq m(s^*)$, for any $s$ with $s^T s = (s^*)^T s^* = \Delta^2$, we have

$$m(s) \geq m(s^*) + \frac{\alpha}{2}((s^*)^T s^* - s^T s).$$

Substituting $g = -(H + \alpha I)s^*$, we get

$$\frac{1}{2}(s - s^*)^T (H + \alpha I)(s - s^*) \geq 0.$$

Since the set of directions

$$\{w : w = \pm \frac{s - s^*}{\|s - s^*\|}, for \quad some \quad \|s\| = \Delta\},$$

is dense in the unit sphere, (c) holds.

Now it is remained to show that $\alpha \geq 0$. As (a) and (c) are satisfied by $s^*$, by Lemma 2.1 (i), it minimizes $\bar{m}$, and so $m(s) \geq m(s^*) + \frac{\alpha}{2}((s^*)^T s^* - s^T s)$. Suppose that there are only negative values of $\alpha$ satisfying (a) and (c). This implies $m(s) \geq m(s^*)$ whenever $\|s\| > \|s^*\| = \Delta$. Thus, $s^*$ is a global minimizer of m as it is already a minimizer for $\|s\| \leq \Delta$. Then, by Lemma 2.1 (i), $Hs^* = -g$ and H is PSD. Therefore, (a) and (c) are satisfied by $\alpha = 0$, which is a contradiction. Hence, $\alpha \geq 0$. □

**Corollary 2.1.** *1) The point $s = -H^{-1}g$ is the unconstrained minimizer of $m(s)$ whenever H is positive definite. Moreover, if it is feasible,then $s^* = s$.*

*2) If H is positive definite and $\|H^{-1}g\| < \Delta$, then the solution $s^*$ of (2.1) doesn't lie on the boundary of the trust region.*

*Proof.*

1) It is immediate consequence of the above theorem.

2) Suppose $\|s^*\| = \Delta$. By (1), $s = -H^{-1}g$ is a solution of (2.1). Since H is positive definite and $\|H^{-1}g\| < \Delta$, it contradicts Lemma 2.1(ii)(uniqueness of $s^*$). Hence, $\|s^*\| \neq \Delta$.

□

## 2.3 Two Dimensional Subspace Minimization Strategy

As we have seen in the previous chapter, the dogleg method is effective when the Hessian $B_k$ of the quadratic model is positive definite at each iteration k. The same is true for Newton trust region method($B_k = H_k = \nabla^2 f(x_k)$). Moreover, it may be costly to (approximately)solve (2.1) over all of $\mathbb{R}^n$. Thus, we need a trust region method in between these two.

**Definition 2.2.** *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be twice continuously differentiable and $g$ and $H$ be the gradient vector and Hessian matrix of $f$ respectively.*

*i. If $H(x)$ has a negative eigenvalue, the $x$ is said to be an indefinite point.*

*ii. If $x$ is an indefinite point and a vector $d$ satisfies $d^T H(x)d < 0$, then $d$ is said to be a **negative curvature direction** of $f$ at $x$.*

*iii. If $r^T g \leq 0$, $d^T g \leq 0$, $d^T H(x)d < 0$, then the vector pair (r,d) is said to be a descent pair at indefinite point $x$. If $x$ is not an indefinite point and satisfies*

$$r^T g < 0, \quad d^T g \leq 0, \quad d^T H(x)d = 0,$$

*then the vector pair (r,d) is said to be a descent pair at $x$.*

**Remark 2.3.** *If a negative curvature direction $d$ satisfies*

*i. $d^T g = 0$, then both $d$ and $-d$ are descent directions.*

*ii. $d^T g \leq 0$, then $d$ is a descent direction.*

*iii. $d^T g \geq 0$, the $-d$ is a descent direction.*

The two dimensional subspace minimization approach was firstly suggested by Shultz, Schnabel and Byrd [7]. This method defines the step $s$ to lie in a subspace $S_k$ spanned by the unconstrained minimizer, $s_k^u = -\frac{g_k^T g_k}{g_k^T H_k g_k} g_k$ and the Newton point $s_k^N = -H_k^{-1} g_k$ (equivalently, $s_k^d = -g_k$ and $s_k^N$ ) (in case when $H_k$ is positive definite) or the inexact Newton directions, $(H_k + \alpha I)^{-1} g$, otherwise where $\alpha_k$ is an approximate to the smallest eigenvalue of $H_k$.

**Remark 2.4.** *The steepest descent and the Newton directions are chosen due to their **global** and **fast** convergence properties respectively.*

**Lemma 2.2.** *Consider the problem (2.1) and let $H$ be positive definite. For any $\alpha \geq 0$, the curve $s(\alpha) = -(H + \alpha I)^{-1} g$ is a descent direction for $f$. Moreover, either there is a unique $\alpha^*$ such that $\|s(\alpha^*)\| = \Delta$ and $s(\alpha^*)$ is the solution of (2.1) or $\|s(0)\| < \Delta$ and $s(0)$ is the solution of (2.1).*

*Proof.* If $\|s(0)\| < \Delta$ , then clearly $s(0)$ is a global minimizer of (2.1). Otherwise consider the Lagrangian

$$L(s, \alpha) = f + g^T s + \frac{1}{2} s^T H s + \frac{1}{2}\alpha(s^T s - \Delta^2),$$

Then, we have

$$\frac{\partial L(s,\alpha)}{\partial s} = Hs + \alpha s + g = 0 \Rightarrow s = -(H + \alpha I)^{-1} g \quad and \quad s^T s = \Delta^2.$$

Since $H$ is symmetric PD, so are $H + \alpha I \quad and \quad (H + \alpha I)^{-1},\ \forall \alpha \geq 0$. Thus, we have

$$g^T s = -g^T (H + \alpha I)^{-1} g < 0,\ \forall \alpha \geq 0.$$

Therefore, the curve $s(\alpha) = -(H + \alpha I)^{-1} g$ is descent direction for all $\alpha \geq 0$. Since H is symmetric , we can make the set of its eigenvectors an orthonormal set in $\mathbb{R}^n$. Thus, we expand g with eigenvectors $u_i, i = 1, 2, ..., n$ of H to show uniqueness. That is, $g = \sum_{i=1}^{n} c_i u_i$.

$$\Rightarrow (H + \alpha I)^{-1} g = \sum_{i=1}^{n} \frac{c_i}{\lambda_i + \alpha} u_i.$$

Or $\|(H+\alpha I)^{-1} g\|^2 = \sum_{i=1}^{n} \frac{c_i^2}{(\lambda_i+\alpha)^2} \Rightarrow \|(H+\alpha I)^{-1} g\|$ is monotonically decreasing function of $\alpha$. □

**Theorem 2.2.** *Consider the problem (2.1) and let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ be eigenvalues of $H$, where the iteration subscript $k$ is omitted. For any $\alpha > -\lambda_1$, $H + \alpha I$ is PD and the curve $s(\alpha) = -(H + \alpha I)^{-1} g$ is a descent direction for $f$. Moreover, either $\|s(0)\| < \Delta$ with $g^T s(0) < 0$ and $s(0)$ is a local minima of (2.1) or there exists $\alpha^* > -\lambda_n$ such that $\|s(\alpha^*)\| = \Delta$ and $s(\alpha^*)$ is a local minima of (2.1).*

*Proof.* Since H is symmetric, there exist an orthogonal matrix Q and a diagonal matrix D such that $H = Q^T D Q$ where $D = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ and columns of Q are corresponding eigenvectors of H. Then, for any $\alpha > -\lambda_1$,

$$\begin{aligned} &H + \alpha I = Q^T D Q + \alpha I = Q^T (D + \alpha I) Q \\ &\Rightarrow H + \alpha I = Q^T \Lambda Q, where \quad \Lambda = diag(\lambda_1 + \alpha, \lambda_2 + \alpha, ..., \lambda_n + \alpha) \\ &\Rightarrow H + \alpha I \ is \ PD (as \ \lambda_i + \alpha > 0, \forall \ i = 1, 2, ..., n). \end{aligned}$$

So, $s(\alpha)$ is a descent direction by the above Lemma. Thus, we have two constraints, namely, $s^T s \leq \Delta^2$ and $g^T s < 0$. If $\|s(0)\| < \Delta$, then $s(0)$ is a minimizer. Otherwise, consider the Lagrangian

$$L(s, \alpha, \mu) = f + g^T s + \frac{1}{2} s^T H s + \frac{1}{2}\alpha(s^T s + \epsilon^2 - \Delta^2) + \mu(g^T s + \delta^2).$$

Then, we obtain the following nonlinear system

$$L_s = Hs + \alpha s + (1 + \mu) g = 0 \tag{2.4}$$

$$2L_\alpha = s^T s + \epsilon^2 - \Delta^2 = 0 \tag{2.5}$$

$$L_\mu = g^T s + \delta^2 = 0 \tag{2.6}$$

$$L_\epsilon = \alpha \epsilon = 0 \tag{2.7}$$

$$L_\delta = 2\delta\mu = 0 \tag{2.8}$$

From (2.4), we get $s = -(1+\mu)(H+\alpha I)^{-1}g$. We can see that $\mu = 0$ satisfies $g^T s < 0$. Substituting s in to (2.4) gives $g^T(H+\alpha I)^{-1}g = \delta^2 \geq 0$. So, if $\epsilon \neq 0$, then we must have $\alpha = 0$ and $\| - H^{-1}g\| = \|s\| < \Delta$ with $g^T H^{-1} g \geq 0$. On the other hand , $\epsilon = 0$ implies $\| - (H+\alpha I)^{-1}g\| = \|s\| = \Delta$ with $g^T(H+\alpha I)^{-1}g \geq 0$. Then, expanding $g$ with orthonormalized eigenvectors of $H$ gives

$$\|(H+\alpha I)^{-1}g\|^2 = \sum_{i=1}^{n} \frac{c_i^2}{(\lambda_i + \alpha)^2}.$$

Therefore, $\|(H+\alpha I)^{-1}g\|$ is monotonically decreasing function of $\alpha$ for $\alpha > \lambda_i$, where $i$ is the first index such that $c_i \neq 0$. □

When $H_k$ is positive definite, we define the step to be the full Newton point if it is feasible. Otherwise, we search in the two dimensional subspace. That is, the subproblem (2.1) becomes:

$$\min m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T H_k s \quad s.t \quad \|s\| \leq \Delta_k,\ s \in span[s_k^d, s_k^N] \tag{2.9}$$

The Cauchy point $s_k^c$ is feasible for (2.9). Thus, as much as possible, we find an approximate solution resulting at least a positive fraction of a reduction in $m_k$ achieved by $s_k^c$ . We can see that the entire dogleg path (see chapter 1) lies in the subspace $S_k$. Therefore, the dogleg method strategy can be made slightly more sophisticated by widening the search for $s$ to the entire two-dimensional subspace $S_k$ in this case. Two dimensional subspace minimization method is also applied to handle indefinite Hessian matrices. As a result, it is called an extension of the dogleg method [5].

- ♢ Whenever $H_k$ is positive definite, it is not as such difficult unlike the case when $H_k$ is not positive definite because the Newton point is the unconstrained minimizer.

- ♢ If $H_k$ is not positive definite, then either it has **zero eigenvalue but no negative eigenvalue** or it has **negative eigenvalue** which is the most difficult case. When $H_k$ has zero eigenvalue but no negative eigenvalue, we take the Cauchy point $s_k^c$ to be our approximate minimizer.

When $H_k$ has negative eigenvalue, we find an approximate $\zeta_k$ to the most negative eigenvalue $\lambda_1^k$ of Hessian $H_k$ in order to form sufficiently positive definite matrix $(H_k + \alpha_k I)$ where $\alpha_k \in (|\zeta_k|, 2|\zeta_k|]$. Later on, we will see how such $\alpha_k$ is determined. Here, we have two subcases.

- ♢ The **first** one is when $\| - (H_k + \alpha_k I)^{-1} g_k\| \leq \Delta_k$. This case is called the **hard case** and in such situation, we discard the subspace search, instead we define the step to be $s = -(H_k + \alpha_k I)^{-1} g_k \quad$ if $\quad \| - (H_k + \alpha_k I)^{-1} g_k\| = \Delta_k$, and otherwise we use some technical improvement on s. Since s is a descent direction, we move in the direction of s till it reaches the boundary. That is,

$$s = -(H_k + \alpha_k I)^{-1} g_k + \gamma_k v_k$$

, for some vector $v_k$ in $\mathbb{R}^n$ and $\gamma_k > 0$ . Then, we can see that

$$\|s\|^2 = \|(H_k + \alpha_k I)^{-1} g_k\|^2 + \gamma_k^2 \|v\|^2 - 2\gamma_k v^T (H_k + \alpha_k I) g.$$

Since $\gamma_k > 0$, we consider $v_k$ such that $v_k^T (H_k + \alpha_k I)^{-1} g_k \leq 0$ *so that* $\|s\| \geq \|(H_k + \alpha_k I)^{-1} g_k\|$. Moreover, $\gamma_k > 0$ is chosen so that $\|s\| = \Delta$. We will see how $v_k$ *and* $\gamma_k$ are chosen after the following theorem.

$\diamondsuit$ The **second** case is whenever $\| - (H_k + \alpha_k I)^{-1} g_k\| > \Delta_k$. In such a case the inexact Newton direction $-(H_k + \alpha_k I)^{-1} g_k$ may not even unconstrained minimizer. Therefore, we search for s in the two dimensional subspace:
$S_k = span[s_k^d, -(H_k + \alpha_k I)^{-1} g_k] \quad for \quad some \quad \alpha_k \in (-\lambda_1^k, -2\lambda_1^k]$ where $\lambda_1^k$ is the smallest negative eigenvalue of $H_k$.

We can see that there are two main points to be discussed in this trust region method. These are:

i) A method to find $\alpha_k$ and $v_k$.

ii) A method to minimize $m_k$ over the two dimensional subspace $S_k$ defined above.

As we have said, in order to handle the indefinite cases, we seek directions of negative curvatures.

**Theorem 2.3.** *Suppose that $H_k$ has negative eigenvalue. If $\lambda_1^k$ is the smallest eigenvalue and $w_k$ a corresponding eigenvector of $H_k$, then $w_k$ is a direction of negative curvature of $f$ at $x_k$.*

*Proof.* $H_k w_k = \lambda_1^k w_k \Rightarrow w_k^T H_k w_k = \lambda_1^k \|w_k\|^2 < 0$. Therefore, $w_k$ is a negative curvature direction of $f$. □

**Remark 2.5.** *In practical algorithms, we include the **hard case** in a single case. To do so,consider the expression(where the subscript k is omitted):*

$$\gamma = -s^T q + \sqrt{(s^T q)^2 + \Delta^2 - \|s\|^2},$$

*where q is an orthonormalized eigenvector corresponding to the most negative eigenvalue of H. Here, $\gamma \geq 0$. Let $u = \gamma q$. Since q is a negative curvature direction, so is u, for $\gamma \neq 0$ and*

$$u^T (H + \alpha I)^{-1} g = -\gamma q^T s = (s^T q)^2 - s^T q \sqrt{(s^T q)^2 + \Delta^2 - \|s\|^2} \leq 0.$$

*Moreover, $\|s + u\|^2 = \|s\|^2 + \|u\|^2 + 2s^T u = \|s\|^2 + \gamma^2 + 2\gamma s^T q = \Delta^2$. Now we can see that, $u = 0$ when $\|s\| = \Delta$. Otherwise, we take $s + u = s + \gamma q$ to be our approximate minimizer.*

For large scale problems, it is costly to apply eigendecomposition and hence we have to use a numerical method to approximate eigenvalues and eigenvectors.

**Remark 2.6.** *Theorem 2.3 together with the following iterative method(Lanczos) tackles the* ***hard case****.*

There are various numerical techniques to approximate the smallest eigenvalue of a **real symmetric matrix** such as power method, Lanczos method etc. But, in this paper, we use the so called ***Lanczos iterative*** method.

### 2.3.1 Lanczos Iterative Method

We would like to find out the eigenvectors and eigenvalues. More specifically, we are interested in finding good estimates of the top and the bottom eigenvalues and the corresponding eigenvectors very quickly. Thus, we will use a method named Lanczos Iteration which at the end of any step gives a tri-diagonal matrix whose extreme eigenvalues approximate the extreme eigenvalues of a symmetric real matrix A. Under suitable initial conditions, the tridiagnoal matrix at the $m^{th}$ step has the same eigenvalues and eigenvectors of the original matrix. Being tri-diagonal, the eigendecomposition is easily carried out for that matrix. However, we seek only the smallest eigenvalue and the corresponding eigenvector.

**Rayleigh Quotient**

If $u$ is an eigenvector of the real symmetric matrix A, then $Au = \lambda u$ for some eigenvalue $\lambda$. In other words,$\min_{\lambda}\|(A - \lambda I)u\| = 0$ if $u$ is an eigenvector. If $u$ is not an eigenvector, then analogously, we can define an approximate eigenvalue by:

$$r(u) = \arg\min_{\lambda}\|(A - \lambda I)u\|$$

Here, $r(u)$ is called the Rayleigh quotient associated with the vector $u$.

**Theorem 2.4.** *Let $A$ be a real symmetric matrix of order $n$ and $u$ be any nonzero vector in $\mathbb{R}^n$ that is not an eigenvector corresponding to $\lambda$. Then,*

$$r(u) = \arg\min_{\lambda}\|(A - \lambda I)u\| = \frac{u^T Au}{u^T u}$$

*Proof.*

$$\begin{aligned}\|(A - \lambda I)u\|^2 &= \|(A - \frac{u^T Au}{u^T u}I + \frac{u^T Au}{u^T u}I - \lambda I)u\|^2 \\ &= \|(A - \frac{u^T Au}{u^T u}I)u\|^2 + \|(\frac{u^T Au}{u^T u}I - \lambda I)u\|^2\end{aligned}$$

$\Rightarrow \|(A - \lambda I)u\|$ is minimum when $\lambda = \frac{u^T Au}{u^T u}$.

□

Lanczos iteration terminates in $m$ iterations where each iteration involves estimating the smallest (largest) eigenvalue by minimizing(maximizing) the Rayleigh coefficient over vectors drawn from a suitable subspace. At each iteration, the dimension of the subspace involved in the optimization increases by 1. The sequence of subspaces used are the so called *Krylov subspaces* associated with a random initial vector.

## Krylov Subspaces

**Definition 2.3.** *Let $\mathbb{S}^n$ be space of real symmetric matrices of order n, $q \in \mathbb{R}^n$ and $A \in \mathbb{S}^n$. The order k Krylov subspace generated by A and q, and denoted by $\mathcal{K}(A, q, k)$ is the subspace:*

$$\mathcal{K}(A, q, k) = span(q, Aq, A^2q, ..., A^{k-1}q)$$

*and the Krylov matrix $K(A, q, k)$ is given by:*

$$K(A, q, k) = [q \quad Aq \quad A^2q \quad ... \quad A^{k-1}q]$$

## Tri-Diagonalization and Krylov Subspaces

Tri-Diagonalization and Krylov Subspaces are related by the following theorem.

**Theorem 2.5.** *Suppose the columns of $Q_k = [q_1 \quad q_2 \quad ... \quad q_k]$ form an orthonormal basis of $\mathcal{K}(A, q, k)$ and $A \in \mathbb{S}^n$. Then, $Q_k^T A Q_k = T$ is tri-diagonal.*

*Proof.* Let us consider the $ij^{th}$ entry $T_{ij}$ of $T$ for $i > j + 1$. $T_{ij} = q_i^T A q_j$.
Then, $Aq_j \in span\{q, Aq, A^2q, ..., A^jq\} = span\{q_1, ..., q_{j+1}\}$. Since $\{q_i\}_{i=1}^n$ form an orthonormal set, $q_i$ is orthogonal to $span\{q_1, ..., q_j, q_{j+1}\}$ for $i > j+1$. This means $q_i^T A q_j = 0$. Since $Q_k^T A Q_k$ is symmetric, so is $T$ . Hence, $T_{ij} = 0$ for $i < j - 1$ also. Therefore, $T$ is tri-diagonal. □

## Key Ideas behind Lanczos Iteration

Let $A \in \mathbb{S}^n$ and $\lambda_i$ be the $i^{th}$ smallest eigenvalue of $A$. That is,

$$\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n.$$

We seek to approximate $\lambda_1$.

**Theorem 2.6.** $\lambda_1 = \min_{y \neq 0} \frac{y^T A y}{y^T y}$ *and* $\lambda_n = \max_{y \neq 0} \frac{y^T A y}{y^T y}$

*Proof.* Since A is symmetric, the eigenvectors $\{u_i\}_{i=1}^n$ of A form an orthonormal basis of $\mathbb{R}^n$. Thus any $y \neq 0$ can be written as:

$$y = \sum_{i=1}^{n} a_i u_i.$$

Hence, $r(y) = \frac{y^T A y}{y^T y} = \frac{\sum_{i=1}^n a_i^2 \lambda_i}{a_i^2} \geq \lambda_1$ *and* $r(y) \leq \lambda_n$. Thus, $\lambda_1 \leq r(y) \leq \lambda_n$. Moreover, the extreme values $\lambda_1$ *and* $\lambda_n$ attained by setting $a_1 = 1$ with all other $a_i$ set to zero and $a_n = 0$ with all other $a_i$ set to zero, respectively. □

Let us define two quantities, $w_k$ and $W_k$ as follow:

$$w_k = \min_{0 \neq y \in \mathcal{V}_k} \frac{y^T A y}{y^T y} \text{ and } W_k = \max_{0 \neq y \in \mathcal{V}_k} \frac{y^T A y}{y^T y}$$

where $\mathcal{V}_k$ is $k$-dimensional subspace. Then, for any choice of increasing sequence of subspaces,$\mathcal{V}_k \subset \mathcal{V}_{k+1}$, $w_k$ is non-increasing and $w_n = \lambda_1$, and $W_k$ is non-decreasing with $W_n = \lambda_n$ as well.This is because, for each $k$, the optimization is performed over a larger domain than the previous. This shows that $w_k$ is an increasingly better approximate to $\lambda_1$, and $W_k$ to $\lambda_n$ as the step index $k$ increases. Lanczos iteration runs for m steps and in each step $k$, it augments the subspace $\mathcal{V}_k$ suitably to ensure the subsequent estimate of the extreme eigenvalue is good.To do so,we choose sequence of subspaces as follow. Let us rewrite $w_k$ and $W_k$ in terms of an orthonormal basis, given by the columns of $Q_k = [q_1 \quad q_2 \quad ... \quad q_k]$, of the subspace $\mathcal{V}_k$.

Any vector $u \neq 0$ in $\mathcal{V}_k$ can be written as $u = Q_k y$ where $y \in \mathbb{R}^{k\times 1}$. Since $Q_k^T Q_k = I$, we have

$$w_k = \min_{0\neq y\in\mathbb{R}^{k\times 1}} \frac{y^T Q_k^T A Q_k y}{y^T y} = \min_{0\neq y} \quad r(Q_k y) \text{ and } W_k = \max_{0\neq y\in\mathbb{R}^{k\times 1}} \frac{y^T Q_k^T A Q_k y}{y^T y} = \max_{0\neq y} \quad r(Q_k y)$$

Let the minimum $w_k$ and maximum $W_k$ be attained at $\overline{y}$ and $\underline{y}$ respectively.Let $u_k = Q_k \overline{y}$, $v_k = Q_k \underline{y}$. Then,

$$u_k, v_k \in span\{q_1, q_2, ..., q_k\}.$$

We then add $q_{k+1}$ to the set of orthonormal vectors $\{q_i\}_{i=1}^{k}$ such that $\{q_i\}_{i=1}^{k+1}$ is also orthonornal and the column space of $Q_{k+1}$ will be the subspace $\mathcal{V}_{k+1}$.
Since we are interested to minimize $w_k$ and maximize $W_k$ over suitable increasing sequences of subspaces (the larger contain the previous minimum rate of change, we want to have $\nabla r(u_k), -\nabla r(v_k) \in span\{q_1, q_2, ..., q_k, q_{k+1}\}$, where $\nabla r$ denotes the gradient vector at a certain vector argument.

**Remark 2.7.** *The gradient of the Rayleigh quotient $r(x)$ of a vector $x$ is:*

$$\nabla r(x) = \frac{2}{x^T x}(Ax - r(x)x).$$

Thus, $\nabla r(x) \in span\{x, Ax\}$. Since $u_k, v_k \in span\{q_1, q_2, ..., q_k\}$, we have

$$\nabla r(u_k), -\nabla r(v_k) \in span\{q_1, q_2, ..., q_k, Aq_1, Aq_2, ..., Aq_k\}$$

As mentioned above, we need to show:

$$\nabla r(u_k), \ -\nabla r(v_k) \in span\{q_1, \ q_2, ..., \ q_k, \ q_{k+1}\}$$

If we choose **Krylov** subspaces, that is, $\mathcal{V}_k = \mathcal{K}(A, q_1, k) = span[q_1, Aq_1, A^2 q_1, ..., A^{k-1} q_1]$ such that columns of $Q_k$ is an orthonormal basis for $\mathcal{K}(A, q_1, k)$. Then,

$$span\{q_1, q_2, ..., q_k, Aq_1, Aq_2, ..., Aq_k\} = span(q_1, Aq_1, A^2 q_1, ..., A^{k-1} q_1, A^k q_1)$$

$$\Rightarrow \nabla r(u_k), -\nabla r(v_k) \in span(q_1, Aq_1, A^2 q_1, ..., A^{k-1} q_1, A^k q_1) = span\{q_1, q_2, ..., q_k, q_{k+1}\}.$$

We are done.

**Remark 2.8.** *In general, one needs to orthonormalize the columns of the Krylov matrix $\mathbb{K}(A, q_1, k)$ to get $Q_k$ at every iteration. Hence, computing eigenvalues and eigenvectors of $A$ reduces, at the end of $m^{th}$ steps, to finding the eigendecomposition for tri-diagonal matrix $T$.*

**Lanczos Algorithm**

Let $A \in \mathbb{S}^n$. We seek to determine an orthogonal matrix $Q = [q_1 \quad q_2 \quad ... \quad q_n]$ and a tri-diagonal symmetric matrix $T$ such that

$$T = Q^T A Q.$$

Since $T$ is symmetric, we can write it as:

$$T = \begin{pmatrix} \delta_1 & \beta_1 & & & & 0 \\ \beta_1 & \delta_2 & \beta_2 & & & \\ & \beta_2 & \delta_3 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_{n-2} & \delta_{n-1} & \beta_{n-1} \\ 0 & & & & \beta_{n-1} & \delta_n \end{pmatrix}$$

Using orthogonality of $Q$,

$$T = Q^T A Q \Leftrightarrow AQ = QT$$

If we examine the $k^{th}$ column of $AQ$, we have:

$$\begin{aligned} Aq_k &= QT_k, where\ T_k = \begin{pmatrix} 0 & 0 & ... & \beta_{k-1} & \delta_k & \beta_k & 0 & ... & 0 \end{pmatrix}^T \\ &= \beta_{k-1} q_{k-1} + \delta_k q_k + \beta_k q_{k+1} \end{aligned}$$

Therefore,we can find Q recursively.We first set $\beta_0 q_0 = 0$. Since the set $\{q_i\}_{i=1}^n$ is orthonormal,we have

$$q_k^T A q_k = \delta_k \|q_k\|^2 = \delta_k$$

We now solve for $\beta_k q_{k+1}$ as follow.

$$\beta_k q_{k+1} = (A - \delta_k I) q_k - \beta_{k-1} q_{k-1}.$$

Let $r_k = (A - \delta_k I) q_k - \beta_{k-1} q_{k-1}$.Then for any $\beta_k \neq 0$,

$$q_{k+1} = \frac{r_k}{\beta_k}$$

where $\beta_k = \|r_k\|$. Let as examine in algorithmic way.

**Algorithm 2.1.** ***(Lanczos Iteration):***

***Initialize:*** $r_0 = 0$, $\beta_0 = 1$, $q_0 = 0$, $k = 0$

$$\begin{aligned} \textbf{\textit{while}} \quad & \beta_k \neq 0 \\ & q_{k+1} = \frac{r_k}{\beta_k} \\ & k = k + 1 \\ & \delta_k = q_k^T A q_k \\ & r_k = (A - \delta_k I) q_k - \beta_{k-1} q_{k-1} \\ & \beta_k = \|r_k\| \\ & end \quad \%while \end{aligned}$$

**Theorem 2.7.** *(Termination Criterion)*
*Let $A \in \mathbb{S}^n, q_1 \in \mathbb{R}^n$ and $\|q_1\| = 1$. Then, the Lanczos algorithm runs until iteration $m = rank\ \mathbb{K}(A, q_1, n)$. Moreover, for $i = 1, 2, ..., m$ we have $AQ_k = Q_k T_k + r_k e_k^T$, where*

$$T_k = \begin{pmatrix} \delta_1 & \beta_1 & & & & 0 \\ \beta_1 & \delta_2 & \beta_2 & & & \\ & \beta_2 & \delta_3 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_{k-2} & \delta_{k-1} & \beta_{k-1} \\ 0 & & & & \beta_{k-1} & \delta_k \end{pmatrix} \quad and \quad Q_k = \begin{bmatrix} q_1 & q_2 & ... & Q_k \end{bmatrix}.$$

*Proof.* See [2] □

### 2.3.2 The Two-Dimensional Subspace Subproblem

By this method,at each iteration,n-dimensional trust region subproblem is replaced by a 2-dimensional subproblem which is very simple to solve relative to the original subproblem for large n. So, the 2D subspace minimization technique highly reduces computational costs while solving **large scale** problems.
Now onwards, in this section, we omit the iteration subscript k for simplicity. Instead of (2.9), we can consider

$$\min m(s) = g^T s + \frac{1}{2} s^T H s \quad s.t \quad \|s\| \leq \Delta, s \in S = span[s^d, s^N] \tag{2.10}$$

**Main procedures:**

We seek to reduce the 2-dimensional subspace subproblem (2.10) to finding roots of a fourth degree polynomial. First, we form an $n \times 2$ matrix A using the steepest descent direction $s^d = -g$ and the Newton point $s^N$ (in case H is positive definite) or the direction of negative curvature otherwise. If these columns of A are linearly dependent, then $s = \alpha g$ for some $\alpha \in \mathbb{R}$ and so the quadratic model in (2.10) becomes to:

$$m(\alpha) = \|g\|^2 \alpha + \frac{1}{2} g^T H g \alpha^2,$$

and its unconstrained minimizer is $-\frac{\|g\|^2}{g^T H g} g$, which is the same us the unconstrianed minimizer of $m(s)$. Therefore,we consider the Cauchy point, $s^c$, defined in chapter one, for the seek of feasibility.
For the opposite case, we make these vectors(columns of A) orthonormal to each other by using Gram Schmidt orthogonalization process and form a new matrix M using the orthonormalized columns of A. That is $M = \begin{pmatrix} m_1 & m_2 \end{pmatrix}$, where $m_1 = \frac{g}{\|g\|}$ and $m_2 = \frac{w}{\|w\|}$, where $w = s^N - \frac{(s^N)^T g}{\|g\|^2} g$. Then, from linear algebra the span $S'$of these orthonormal

vectors is equal to $S = span[s^d, s^N]$. Thus, for any $s \in S, \exists q = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$ such that $s = Mq$.

Therefore, the subproblem (2.10) becomes:

$$min \quad m(q) = U^T q + \frac{1}{2} q^T G q \quad s.t \quad \|q\| \leq \Delta, \tag{2.11}$$

where $U = M^T g$ and $G = M^T H M$, is a $2 \times 2$ matrix.
Now this problem is too small (2-dimensional) with unknowns $x$ and $y$ which is easier to solve than the previous problem(n-dimensional).

Subproblem (2.11) can be reduced to finding roots of a *fourth degree polynomial* using some algebraic manipulations. Now, by Remark 2.1, the subspace subproblem (2.11) has a solution. Thus, such a solution lies either **in** the trust region,$\Omega$ or **on** the boundary of it. That is, we have two cases.

**Case1:** $G$ is positive definite and $\| - G^{-1}U\| \leq \Delta$
In this case $s = -G^{-1}U$.
**Case2:** $G$ is not positive definite or $\| - G^{-1}U\| > \Delta$
A solution s that lies on the boundary of the trust region satisfies the condition(s) of this case by the converse of Corollary 2.1. So, we can consider optimal solution of (2.10) that lies on the boundary of the trust region. In this case, (2.11) becomes a constrained problem with a single equality constraint in two variables, namely, $x \quad and \quad y$, and we can use the Lagrangian technique.
Let $U^T = \begin{pmatrix} b_1 & b_2 \end{pmatrix}$ and $G = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$.
Then,

$$\begin{aligned} U^T q + \frac{1}{2} q^T G q &= \begin{pmatrix} b_1 & b_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ &= b_1 x + b_2 y + \frac{1}{2}[a_{11}x^2 + a_{12}xy + a_{21}xy + a_{22}y^2] \\ &= \frac{1}{2}a_{11}x^2 + \frac{1}{2}a_{22}y^2 + \frac{1}{2}(a_{12} + a_{21})xy + b_1 x + b_2 y \end{aligned}$$

Multiplying both sides of the last equation by 2 gives:

$$2U^T q + q^T G q = a_{11}x^2 + a_{22}y^2 + 2a_{12}xy + b_1 x + b_2 y$$

Put $a_1 = a_{11}, a_2 = a_{12}, a_3 = a_{22}, a_4 = b_1 \quad and \quad a_5 = b_2$. Then,(2.10) becomes:

$$\min a_{11}x^2 + a_{22}y^2 + 2a_{12}xy + b_1 x + b_2 y \quad s.t \quad x^2 + y^2 = \Delta^2 \tag{2.12}$$

Then (2.12) can be reduced to a fourth degree polynomial equation using the Lagrangian approach. The Lagrangian is:

$$L(x, y, \mu) = a_1 x^2 + 2a_2 xy + a_3 y^2 + 2a_4 x + 2a_5 y + \mu(x^2 + y^2 - \Delta^2)$$

A minimizer of (2.12) should satisfy the following system of equations.

$$\frac{\partial L}{\partial x} = 2a_1x + 2a_2y + 2a_4 + 2\mu x = 0$$
$$\frac{\partial L}{\partial y} = 2a_2x + 2a_3y + 2a_5 + 2\mu y = 0$$
$$\frac{\partial L}{\partial \mu} = x^2 + y^2 - \Delta^2 = 0$$

After eliminating $\mu$, we obtain:

$$a_2x^2 + (a_3 - a_1)xy - a_2y^2 + a_5x - a_4y = 0$$

$$x^2 + y^2 = \Delta^2$$

To eliminate the last equation,we can use the parametrization:

$$x = \frac{2\Delta t}{1+t^2} \quad , \quad y = \frac{\Delta(1-t^2)}{1+t^2}, where \;\; t \in \mathbb{R} \tag{2.13}$$

The last required $4^{th}$ degree polynomial equation is:

$$c_1t^4 + c_2t^3 + c_3t^2 + c_4t + c_5 = 0 \tag{2.14}$$

where $c_1 = \Delta(-a_2\Delta + a_4), c_2 = 2\Delta((-a_1 + a_3)\Delta + a_5)$,
$c_3 = 6\Delta^2a_2, c_4 = 2\Delta((-a_1 + a_3)\Delta + a_5) \quad and \quad c_5 = -a_2\Delta^2 - a_4\Delta$

Now on, the problem is changed to finding real roots of (2.14). We use MATLAB to find roots of the polynomial and we take a root that results the most reduction in the reduced model function in (2.11). Then, we solve for x and y back by the parametrization used.

### 2.3.3 The 2D Subspace Minimization Algorithm

Byrd, Schnabel and Schultz [**7**] extended the minimization to the whole bidimensional subspace containing the dogleg path in such a way that also indefinite Hessian H can be used. The two-dimensional subspace minimization algorithm to find approximate solution of Newton trust region subproblem is stated below.

**Algorithm 2.2.** *(Two-dimensional subspace approximation of TRS):*

***Input:***
*initial guess*$(x_0)$, $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, $\eta \in [0, \frac{1}{4})$,
$tol > 0$*(suff small error tolerance)*,$g_0, H_0$, $k := 0$.
***Output:*** *approximate minimizer* $s \approx s^*$ *of the TRS(2.10)*
***1.*** *Approximating the most negative eigenvalue using algorithm 2.1*
$v \approx \lambda_1 := \min eig(H)$ *such that* $|v| \in (-\lambda_1, -2\lambda_1]$ *if* $\lambda_1 < 0$
***2.*** *Check definiteness of H*
$\alpha = |v|$

**if** $v > tol$ $\quad \%H \ \ is \ \ PD$
$s^N = -H^{-1}g$
***if*** $\|s^N\| \leq \Delta$
$s = s^N$
***return***
***else***
$S = span[g, -H^{-1}g] \ \ and$ ***go to (3)***
***end*** $\quad \%if$
**else if** $\alpha < tol$ $\quad \%(H \ \ is \ \ numerically \ \ singular)$
$s = s^c$
***return***
**else** $\quad \%H \ \ is \ \ indefinite$
$s = -(H + \alpha I)^{-1}g$
***if*** $\|s\| \leq \Delta$
$Let \ \ q \ \ be \ \ diretion \ \ of \ \ negative \ \ curvature \ \ and \ \ \|q\| = 1$
$\gamma = -s^T q + \sqrt{(s^T q)^2 + \Delta^2 - \|s\|^2}$
$Let \ \ v = \gamma q. Set \ \ s = s + v$
***return***
***else***
$S = span[g, s] \ \ and$ ***go to (3)***
***end*** $\quad \%if$
**end** $\quad \%if$
***3.*** *Let* $S = span[s_1, s_2]$ *with* $\|s_1\| = \|s_2\| = 1$ *and* $s_1^T s_2 = 0$.
$M = [s_1 \ \ s_1] \in \mathbb{R}^{n \times 2}$.
*Find the minimizer* $q^* = \begin{pmatrix} x \\ y \end{pmatrix}$ *of the model m in Subspace S:*

$$q^* = \underset{x^2+y^2 \leq \Delta^2}{\arg\min} f + g^T M \begin{pmatrix} x \\ y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x \\ y \end{pmatrix}^T M^T H M \begin{pmatrix} x \\ y \end{pmatrix}$$

$s = q^*$
$s^* = Mq^*$
***return***

Once v is available, the descent direction q can be the eigenvector corresponding to it. That is, $Hq \approx vq$.

**Remark 2.9.** *The practical algorithm of the 2D subspace minimization method is obtained by combining Algorithm 3.1 , Algorithm 2.1 and Algorithm 2.2 together.*

## 2.4 Convergence Analysis

In the general trust region method, we have seen that taking the Cauchy point as an approximate minimizer of (1.5) at each iteration has global convergence apart from its slow rate. Thus, **global convergence** of most trust region methods to approximately

solve (1.5) depends on the approximate solution obtaining at least as much decrease in the model function as the Cauchy point does.

In order to see the global convergence of a trust region method, we need to obtain an ***estimate of decrease*** in the model function $m_k$ achieved by the Cauchy point. We then use this estimate to show that the sequence of gradient vectors $\{g_k\}$ generated by algorithm 1.3 converges to zero. Such estimate of decrease is given as:

$$m_k(0) - m_k(s_k) \geq c_1 \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\} \tag{2.15}$$

for some $c_1 \in (0, 1]$, where $s_k$ is an approximate solution of (1.5) produced by a certain trust region method.

**Lemma 2.3.** *The Cauchy point satisfies (2.15) with $c_1 = \frac{1}{2}$. That is,*

$$m_k(0) - m_k(s_k^c) \geq \frac{1}{2} \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\} \tag{2.16}$$

*Proof.*
For simplicity, we drop the iteration subscript k. We consider two cases.
**Case 1:** $g^T B g \leq 0$
Here, we have

$$\begin{aligned} m(s^c) - m(0) &= m(\frac{-\Delta}{\|g\|} g) - f \\ &= -\frac{\Delta}{\|g\|} \|g\|^2 + \frac{\Delta^2}{\|g\|^2} g^T B g \\ &\leq -\Delta \|g\| \\ &\leq -\|g\| \min\{\Delta, \frac{\|g\|}{\|B\|}\} \\ &\leq -\frac{1}{2} \|g\| \min\{\Delta, \frac{\|g\|}{\|B\|}\} \end{aligned}$$

So,we are done in this case.
**Case 2:** $g^T B g > 0$
Here,we have two subcases.
**Case 2.1:** $\frac{\|g\|^3}{\Delta g^T B g} \leq 1$

$$\Rightarrow s^c = -\frac{\|g\|^2}{g^T B g} g.$$

Thus,

$$\begin{aligned} m(s^c) - m(0) &= -\frac{\|g\|^4}{g^T Bg} + \frac{1}{2} g^T Bg \frac{\|g\|^4}{(g^T Bg)^2} \\ &= -\frac{1}{2}\frac{\|g\|^4}{g^T Bg} \\ &\leq -\frac{1}{2}\frac{\|g\|^4}{\|g\|^2\|B\|} \quad (By \quad Cauchy \quad Schwartz \quad inequality) \\ &= -\frac{1}{2}\frac{\|g\|^2}{\|B\|} \\ &\leq -\frac{1}{2}\|g\| \min\{\Delta, \frac{\|g\|}{\|B\|}\} \end{aligned}$$

**Case 2.2:** $\frac{\|g\|^3}{\Delta g^T Bg} > 1$

$$\Rightarrow s^c = -\frac{\Delta}{\|g\|} g.$$

Thus,it is the same as case 1 above. This completes the proof. □

**Theorem 2.8.** *a) Let $s_k$ be any vector such that $\|s_k\| \leq \Delta_k$ and $m_k(0) - m_k(s_k) \geq c_2(m_k(0) - m_k(s_k^c))$, for some positive fraction $c_2$. Then, $s_k$ satisfies (2.9) with $c_1 = \frac{c_2}{2}$.*

*b) A step obtained by the two dimensional subspace minimization algorithm satisfies (2.15) with $c_1 = \frac{1}{2}$.*

*c) Let $s^*, s^{2D}, s^D$ be optimal solution, approximate solution by two-dimensional subspace minimization method and approximate solution by the dogleg method,respectively,of the Newton trust region subproblem (2.1)) with PD B.Then,*

$$m(s^*) \leq m(s^{2D}) \leq m(s^D) \leq m(s^c)$$

*where $s^c$ is the Cauchy point.*

*Proof.* a) Since $\|s_k\| \leq \Delta_k$, (2.16) gives the result.

b) Let $s_k^{2D}$ be an approximate solution obtained by 2D subspace algorithm.Since $m_k(s_k^{2D}) \leq m_k(s_k^c)$, (2.15) gives:

$$m_k(0) - m_k(s_k^{2D}) \geq \frac{1}{2}\|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\}.$$

c) While solving (2.1) exactly , its feasible set is $\Omega = \{s \in \mathbb{R}^n : \|s\| \leq \Delta\}$. Then,obviously $s^{2D}, s^D, s^c \in \Omega$.
Since $s^*$ is the minimizer of (2.1) over $\Omega$,we must have

$$m(s^*) \leq m(s^{2D}), m(s^*) \leq m(s^D) \quad and \quad m(s^*) \leq m(s^c)$$

While solving (2.1) using two dimensional subspace method,the feasible set is:

$$S^{2D} = \{s : \|s\| \leq \Delta, s \in span[g, -H^{-1}g]\}$$

We know that $s^c$ is the minimizer of (2.1) along the path $s(\tau) = -\tau\frac{\Delta}{\|g\|}g, 0 < \tau \leq 1$ and $s^D$ is the minimizer of (2.1) along the path

$$s(\tau) = \begin{cases} \tau s^u, & 0 \leq \tau \leq 1 \\ s^u + (\tau - 1)(s^N - s^u), & 1 \leq \tau \leq 2. \end{cases}$$

Therefore, $s^c, s^D \in S^{2D}$ and hence $m(s^{2D}) \leq m(s^D)$ and $m(s^{2D}) \leq m(s^c)$.
Consider the minimizer of (2.1) without any constraint, that is,

$$s^u = -\frac{g^T g}{g^T H g} g.$$

If $\|s^u\| > \Delta$ , then $\frac{\|g\|^3}{\Delta g^T H g} > 1$ and so $s^c = -\frac{\Delta}{\|s^u\|}s^u$.

$$\Rightarrow s^c \quad is \quad on \quad the \quad dogleg \quad path.$$
$$\Rightarrow m(s^{2D}) \leq m(s^c)$$

On the other hand if $\|s^u\| \leq \Delta$,then

$$s^c = s^u \in s(\tau)$$

Therefore,$m(s^{2D}) \leq m(s^c)$. Thus,we conclude that

$$m(s^*) \leq m(s^{2D}) \leq m(s^D) \leq m(s^c)$$

□

**Theorem 2.9.** *Assuming Algorithm* 1.3 *is applied with $B_k$ uniformly bounded in norm, $f$ is bounded below on the level set $S = \{x : f(x) \leq f(x_0)\}$ and Lipschitz continuous differentiable in the neighborhood $S(R_0) = \{x : \|x - y\| < R_0\}$, for some constant $R_0$, and the approximate solution $s_k$ of (1.5) satisfies (2.15) and $\|s_k\| \leq \sigma\Delta_k$ for some $\sigma > 0$, and $S$ is assumed to be bounded. Then*

*i. if $\eta = 0$ , then $\liminf_{k\to\infty}\|g_k\| = 0$ ,and*

*ii. if $\eta \in (0, \frac{1}{4})$, then $\lim_{k\to\infty}\|g_k\| = 0$*

*Proof.* As $B_k$ is uniformly bounded, there exists $\beta \in \mathbb{R}$ such that $\|B_k\| \leq \beta, \forall k \in \mathbb{R}^n$.

i. From (1.6),we have

$$|\rho_k - 1| = \frac{m_k(s_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)}$$

Recall Taylor's theorem

$$f(x_k + s_k) = f(x_k) + g(x_k + ts_k)^T s$$

for some $t \in (0,1)$. Thus,we have

$$f(x_k + s_k) = f_k + g_k^T s_k + \int_0^1 [g(x_k + ts_k) - g_k]^T s_k dt,$$

and from the definition of $m_k$

$$\begin{aligned} |m_k(s_k) - f(x_k + s_k)| &= |\frac{1}{2} s_k^T B_k s_k - \int_0^1 [g(x_k + ts_k) - g_k]^T s_k dt| \\ &\leq \frac{\beta}{2}\|s_k\|^2 + C(s_k)\|s_k\| \end{aligned} \tag{2.17}$$

We can see that $C(s_k)$ can be made arbitrarily small by restricting the size of $s_k$. Suppose for contradiction that there is $\epsilon > 0$ and a positive index K such that

$$\|g_k\| \geq \epsilon, \forall k \geq K \tag{2.18}$$

From (2.15), for any $k \geq K$, we have

$$m_k(0) - m_k(s_k) \geq c_1 \epsilon \; min\{\Delta_k, \frac{\epsilon}{\beta}\} \tag{2.19}$$

Since $\|s_k\| \leq \sigma\Delta_k$,(2.17) and (2.19) gives

$$|\rho_k - 1| \leq \frac{\sigma\Delta_k(\Delta_k(\beta/2) + C(s_k))}{2c_1\epsilon \; min(\Delta_k, \epsilon/\beta)} \tag{2.20}$$

This holds for all sufficiently small values $\Delta_k$. Thus,for such $\Delta_k$ with $\|s_k\| \leq \sigma\bar{\Delta}$, we can made $\bar{\Delta}$ small enough so that

$$\Delta_k < \frac{c_1\epsilon}{\beta\sigma} \tag{2.21}$$

Since $C(s_k)$ can be made arbitrarily small and $c_1\epsilon > \beta\sigma\Delta_k$, we can restrict $s_k$ so that

$$C(s_k) \leq \frac{c_1\epsilon - \beta\sigma\Delta_k}{2\sigma}$$

Thus,we have

$$\Delta_k(\beta/2) + C(s_k) \leq \frac{c_1\epsilon}{2\sigma} \tag{2.22}$$

Furthermore, if we made $\Delta_k \leq \sigma\bar{\Delta} \leq \frac{\epsilon}{\beta}$, (2.20) and (2.22) gives

$$|\rho_k - 1| \leq \frac{1}{4}$$

$\implies \rho_k > \frac{3}{4}$
Thus,by Algorithm 1.3 we have $\Delta_{k+1} \geq \Delta_k$ whenever $\Delta_k < \bar{\Delta}$. This shows that the reduction of $\Delta_k$(by a factor of $\frac{1}{4}$) can occur in our algorithm only when $\Delta_k \geq \bar{\Delta}$ and therefore

$$\Delta_k \geq \min\{\Delta_k, \frac{\bar{\Delta}}{4}\}, \forall k \geq K \tag{2.23}$$

Suppose that there is an infinite subsequence $I$ such that $\rho_k \geq \frac{1}{4}$ for $k \in I$. If $k \in I$ and $k \geq K$, we have from (2.19) that

$$\begin{aligned} f(x_k) - f(x_{k+1}) &= f(x_k) - f(x_k + s_k) \\ &\geq \frac{1}{4}(m_k(0) - m_k(s_k)) \\ &\geq \frac{1}{4} c_1 \epsilon \; min\{\Delta_k, \frac{\epsilon}{\beta}\} \end{aligned}$$

Since $f$ is bounded below , we must have :

$$\lim_{k \in I, k \to \infty} \Delta_k = 0,$$

contradicting (2.23). Hence no such infinite subsequence I can exist, and we must have $\rho_k < \frac{1}{4}$ for all k sufficiently large. In this case, $\Delta_k$ will eventually be reduced by a factor of $\frac{1}{4}$ at every iteration, and we have $\lim_{k\infty} \Delta_k = 0$, which again contradicts (2.20). Therefore, our original assertion (2.18) must be false, which completes the proof.

ii. Consider any $m$ such that $g_m \neq 0$. If we use $\theta$ to denote the Lipschitz constant for $g$ on the level set $S$, we have

$$\|g_k - g_m\| \leq \theta \|x - x_m\|, \forall x \in S$$

. Hence, by defining the scalars $\epsilon = \frac{1}{2}\|g_m\|, \quad R = \frac{\|g_m\|}{2\theta} = \frac{\epsilon}{\theta}$, and the ball

$$B(x_m, R) = \{x : \|x - x_m\| \leq R\},$$

we have the following.

$$x \in B(x_m, R) \Longrightarrow \|g(x)\| \geq \|g(x)\| - \|g(x) - g_m\| \geq \frac{1}{4}\|g_m\| = \epsilon.$$

If the entire sequence $\{x_k\}_{k \geq m}$ stays inside the ball $B(x_m, R)$, we would have $\|g_k\| \geq \epsilon > 0, \forall k \geq m$. But by the proof of $(i)$ above, this condition can never occur.Thus, the sequence $\{x_k\}_{k \geq m}$ eventually leaves the ball $B(x_m, R)$. Let the index $r \geq m$ be such that $x_{r+1}$ is the first iterate after $x_m$ outside $B(x_m, R)$. Since $\|g_k\| \geq \epsilon$ for $k = m, m+1, ..., r$, we can use (2.19) to get:

$$\begin{aligned} f(x_k) - f(x_{r+1}) &= \sum_{k=m}^{r} [f(x_k) - f(x_{k+1})] \\ &\geq \sum_{\substack{k=m \\ x_k \neq x_{k+1}}}^{r} \eta [m_k(0) - m_k(s_k)] \\ &\geq \sum_{\substack{k=m \\ x_k \neq x_{k+1}}}^{r} \eta c_1 \epsilon \; min\{\Delta_k, \frac{\epsilon}{\beta}\}, \end{aligned}$$

where the sum is limited to iterations on which a step is actually taken ($x_k \neq x_{k+1}$). If $\Delta_k \leq \frac{\epsilon}{\beta}, \forall k = m, m+1, ..., r$, we have

$$f(x_m) - f(x_{r+1}) \geq \eta c_1 \epsilon \sum_{\substack{k=m \\ x_k \neq x_{k+1}}}^{r} \Delta_k \geq \eta c_1 \epsilon R = \eta c_1 \epsilon^2 \frac{1}{\theta} \tag{2.24}$$

Otherwise, we have $\Delta_k > \frac{\epsilon}{\beta}$ for some $k = m, m+1, ..., r$, and so

$$f(x_m) - f(x_{r+1}) \geq \eta c_1 \epsilon^2 \frac{1}{\beta} \tag{2.25}$$

Since the sequence $\{f(x_k)\}_{k=0}^{\infty}$ is decreasing and bounded below, we have $f(x_k) \longrightarrow f^*, as\ k \to \infty$, for some $f^* > -\infty$. Therefore,using (2.24) and (2.25), we can write

$$\begin{aligned} f(x_k) - f^* \geq f(x_m) - f(x_{r+1}) &\geq \eta c_1 \epsilon^2\ min\{\frac{1}{\beta}, \frac{1}{\theta}\} \\ &= \frac{1}{4}\eta c_1\ min\{\frac{1}{\beta}, \frac{1}{\theta}\}\|g_m\|^2 \end{aligned} \tag{2.26}$$

$(2.26) \Longrightarrow \|g_m\|^2 \leq (\frac{1}{4}\eta c_1\ min\{\frac{1}{\beta}, \frac{1}{\theta}\})^{-1}(f(x_m) - f^*)$.
Thus, as $m \longrightarrow \infty, \|g_m\|^2 \longrightarrow 0$. Hence, $\lim_{m\to\infty} \|g_m\| = 0$.

□

Local fast rate convergence of a trust region method with $B_k = H_k$ usually associated with Newton's method. Steps that satisfy the trust region bound and becomes closer and closer to the true Newton step near the solution are said to be *asymptotically similar* to Newton steps. We show any algorithm of the form Algorithm 1.3 that produce such steps has superlinear convergence.

**Theorem 2.10.** *Let $f$ be twice Lipschitz continuously differentiable in a neighborhhod of a point $x^*$ at which second-order sufficient conditions are satisfied.Suppose the sequence $\{x_k\}$ converges to $x^*$ and that for all k sufficiently large, the trust-region algorithm based on (2.1) chooses steps $s_k$ that satisfy the Cauchy-point-based model reduction criterion (2.15) and are asymptotically similar to Newton steps $s_k^N$ whenever $\|s_k^N\| \leq \frac{1}{2}\Delta_k$, that is,*

$$\|s_k - s_k^N\| = o(\|s_k^N\|) \tag{2.27}$$

*Then,the trust-region bound $\Delta_k$ becomes inactive for all k sufficiently large and the sequence $x_k$ converges **superlinearly** to $x^*$.*

*Proof.* We show that $\|s_k^N\| \leq \frac{1}{2}\Delta_k$ , the TR bound is inactive and the rate of convergence of $\{x_k\}$ is superlinear, for all sufficiently large k.Then,we seek a lower bound for predicted reduction.Assume that k is large enough that the $o(\|s_k^N\|)$ term is less than $\|s_k^N\|$. When $\|s_k^N\| \leq \frac{1}{2}\Delta_k$, from (2.27) and the fact $\|x\| - \|y\| \leq \|x - y\|$, we have $\|s_k\| \leq \|s_k^N\| +$

$o(\|s_k^N\|) \leq 2\|s_k^N\|$. When $\|s_k^N\| \leq \frac{1}{2}\Delta_k$, we have $\|s_k\| \leq \Delta_k < 2\|s_k^N\|$. In both cases, we have

$$\|s_k\| \leq 2\|s_k^N\| \leq 2\|H_k^{-1}\|\|g_k\|,$$

and so $\|g_k\| \geq \frac{1}{2}\Delta_k/\|H_k^{-1}\|$. Then, by (2.15)

$$\begin{aligned} m_k(0) - m_k(s_k) &\geq c_1\|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|H_k\|}\} \\ &\geq c_1 \frac{\|s_k\|}{2\|H_k^{-1}\|} \min\{\|s_k\|, \frac{\|s_k\|}{2\|H_k\|\|H_k^{-1}\|}\} \\ &\geq c_1 \frac{\|s_k\|^2}{4\|H_k\|\|H_k^{-1}\|^2}. \end{aligned}$$

Since $x_k \to x^*$, using the continuity of $H_k$ and positive definiteness of $H(x^*)^{-1}$, for all sufficiently large k, we get

$$\frac{c_1}{4\|H_k\|\|H_k^{-1}\|^2} \geq \frac{c_1}{8\|H(x^*)\|\|H(x^*)^{-1}\|^2} = c_3,$$

where $c_3 > 0$. Hence, for all sufficiently large k, we have

$$m_k(0) - m_k(s_k) \geq c_3\|s_k\|^2 \tag{2.28}$$

From Taylor's theorem,we have that

$$f(x_k + s) = f_k + g_k^T s + \frac{1}{2} s^T \nabla^2 f(x_k + ts)s,$$

for some $t \in (0, 1)$. Using this and Lipschitz continuity of $H(x)$ near $x^*$, we have

$$\begin{aligned} &|f_k - f(x_k + s_k - (m_k(0) - m_k(s_k))| \\ &= |\frac{1}{2} s_k^T H_k s_k - \frac{1}{2}\int_0^1 s_k^T H(x_k + ts_k)s_k dt| \\ &= \frac{1}{2}\int_0^1 |s_k^T (H_k - H(x_k + ts_k))s_k| dt \\ &\leq \frac{\|s_k\|^2}{2}\int_0^1 \|H_k - H(x_k + ts_k)\| dt \\ &\leq \frac{L}{2}\|s_k\|^3 \end{aligned} \tag{2.29}$$

where $L > 0$ is the Lipschitz constant.Then, from (1.6), (2.27), (2.28),we get

$$|\rho_k - 1| \leq \frac{\|s_k\|^3 (L/2)}{c_3\|s_k\|^2} \leq \frac{L}{2c_3}\|s_k\| \leq \frac{L}{2c_3}\Delta_k \tag{2.30}$$

Since the TR radius can be reduced only if $\rho_k < \frac{1}{4}$(or some other fixed number less than 1), so from (2.30, the sequence $\{\Delta_k\}$ is bounded away from zero. Moreover, since $x_k \to x^*$, satisfying the $2^{nd}$ order sufficient condition, we must have $\|s_k^N\| \to 0$ and therefore $\|s_k\| \to 0$ by (2.27). Thus,the trust region bound is inactive for all k sufficiently large

, and the bound $\|s_k^N\| \leq \frac{1}{2}\Delta_k$ is eventually always satisfied.
To prove superlinear convergence, we use the quadratic convergence of Newton's method(see chapter one). That is

$$\|x_k + s_k^N - x^*\| = o(\|x_k - x^*\|^2),$$

which implies that $\|s_k^N\| = o(\|x_k - x^*\|)$. Then, by (2.27)

$$\begin{aligned}
&\|x_k + s_k - x^*\| \\
&\leq \|x_k + s_k^N - x^*\| + \|s_k^N - s_k\| \\
&= o(\|x_k - x^*\|^2) + o(\|s_k^N\|) = o(\|x_k - x^*\|).
\end{aligned}$$

□

**Remark 2.10.** *This theorem together with Theorem 2.7(b) and Theorem 2.9 gives convergence property of the 2D subspace minimization method.*

**Example:** Consider the following unconstrained problem

$$\min_{x\in\mathbb{R}^3} f(x), \tag{2.31}$$

where $f(x) = 1+(x_1-x_2)^2+(x_2-5)^4+(x_3-x_1)^2$. Let us solve (2.31) by the 2D subspace minimization algorithm using input data
$x_0 = [3 \quad 6 \quad 4]^T, \Delta_0 = 2, \bar{\Delta} = 100, \eta = 0.24, tol = toleig = 0.001$. Then,the gradient vector and Hessian matrix of $f$ at $x_0$ are $g_0 = [-8, 10, 2]^T$, and $H_0 = \begin{pmatrix} 4 & -2 & -2 \\ -2 & 14 & 0 \\ -2 & 0 & 2 \end{pmatrix}$ respectively,and $f_0 = f(x_0) = 12$. So, the quadratic model of $f$ is

$$m_0(s) = 12 + g_0^T s + \frac{1}{2} s^T H_0 s,$$

where $s = [s_1 \quad s_2 \quad s_3]^T \in \mathbb{R}^3$. The minimum eigenvalue of $H_0$ is $\lambda = 0.6768 > tol$, so ,$H_0$ is PD and hence invertible. $s_0^N = -H_0^{-1} g_0 = [2.6667 \quad -0.3333 \quad 1.6667]^T$ and $\|s_0^N\| = 3.1623 > \Delta_0$. Thus,we must search for $s$ in the 2D subspace $S_0 = span\{g_0, -H_0^{-1} g_0\}$. Let $A = [g_0 \quad -H_0^{-1} g_0]$. Then,$A$ is a $3 \times 2$ matrix whose columns are linearly independent. Hence,we can orthonormalize the columns of $A$ to form a new matrix $M = [m_1 \quad m_2]$ with the set $\{m_1, m_2\}$ is orthonormal,by using Gram Schmidt orthogonalization process.That is,

$$m_1 = \frac{g_0}{\|g_0\|} = [-0.6172 \quad 0.7715 \quad 0.1543]^T \quad and$$

$$m_2 = \frac{s_0^N - ((s_0^N)^T m_1) m_1}{\|s_0^N - ((s_0^N)^T m_1) m_1\|} = [0.6114 \quad 0.3468 \quad 0.7113]^T.$$

The corresponding subspace subproblem is

$$\min \quad m_0(s) \quad s.t \quad \|s\| \leq \Delta_0, s \in S_0.$$

Since $s \in S_0$, we can find $q = [v \;\; w]^T \in \mathbb{R}^2$ such that $s = Mq$. Then,$\|s\| = \|Mq\|, and$ the reduced 2-dimensional problem is

$$\min \;\; m_0^r(q) = f(Mq) + U^T q + \frac{1}{2} q^T G q \;\; s.t \;\; \|q\| \leq \Delta_0$$

,where $f(Mq) = 812.6777, U = M^T g_0 \;\; and \;\; G = M^T H_0 M.$

Now,$G = \begin{pmatrix} 12.1905 & 2.6305 \\ 2.6305 & 1.6034 \end{pmatrix}$, $U = \begin{pmatrix} 12.9615 & 0.0000 \end{pmatrix}$. The Newton point of this reduced problem is $s^N = -G^{-1}U = [-1.6459 \;\; 2.7002]$ with $\|s^N\| = 3.1623 > \Delta_0$. Then,by using the parametrization (2.13) in section 2.3.2 for $v \;\; and \;\; w$, we can change this problem in to finding roots of a fourth degree polynomial.So,$a_1 = 12.1905$,
$a_2 = 2.6305, a_3 = 1.6034, a_4 = 12.9615 \;\; and \;\; a_5 = 0.0000$. This implies that $c_1 = 15.4009$, $c_2 = -84.6964, c_3 = 63.1321, c_4 = -84.6964, \;\; and \;\; c_5 = -36.4450$. The required polynomial is

$$p(t) = c_1 t^4 + c_2 t^3 + c_3 t^2 + c_4 t + c_5.$$

The only real roots are,$r_1 = 4.9128 \;\; and \;\; r_2 = -0.3196$. Then,by the parametrization used for components of $q$, we have

$$v_1 = \frac{2\Delta_0 r_1}{1 + r_1^2} = 0.7818 \;\; and \;\; w_1 = \frac{\Delta_0(1 - r_1)}{1 + r_1^2} = -1.8409.$$

That is, $q_1 = [v_1 \;\; w_1]^T$. Similarly,using the root $r_2$, we get $q_2 = [v_2 \;\; w_2]^T = [-1.1600 \;\; 1.6293]^T$. Now,we compare the functional value using the reduced model function $m_0^r$ and we take the one with less value.

$$m_0^r(q_1) = 24.7898 \;\; and \;\; m_0^r(q_2) = 2.3232.$$

Thus,we must take $q = q_2$ and so the minimizer of the model for this iteration is

$$s = Mq_2 = [1.7120 \quad -0.3298 \;\; 0.9799]^T \;\; and \;\; \|s\| = 2 = \Delta_0.$$

Lastly,we use $\rho_0 = \frac{ared}{pred}$, defined in (1.6) and $\eta = 0.24$ to update $\Delta$ and $x$ respectively.Since $\rho_0 = 1.0136 > 0.75$ and $\|s\| = 2 = \Delta_0$, by Algorithm 1.3,

$$\Delta_1 = \min 2\Delta_0, 100 = 2\Delta_0 = 4.$$

Moreover,$\rho_0 > \eta \;\; and \;\; so, \;\; x_1 = x_0 + s = [4.7120 \;\; 5.6702 \;\; 4.9799]^T$, which is closer to the exact minimizer $x^* = [5 \;\; 5 \;\; 5]$ than $x_0$ because $\|x^* - x_0\| = 2.4495 > 0.7738 = \|x^* - x_1\|$.

In each of the remaining 7 iterations, the full Newton step exists and is feasible. Thus, we take the full Newton step at each iteration as shown in the table below.

| Iteration | $\Delta$ | $\rho$ | $\|sN\|$ | $\|s\|$ |
|---|---|---|---|---|
| 0 | 2.0000 | 1.0136 | 3.1623 | 2.0000 |
| 1 | 4.0000 | 1.0244 | 0.8987 | 0.8987 |
| 2 | 8.0000 | 1.2037 | 0.2579 | 0.2579 |
| 3 | 16.0000 | 1.2037 | 0.1720 | 0.1720 |
| 4 | 32.0000 | 1.2037 | 0.1146 | 0.1146 |
| 5 | 64.0000 | 1.2037 | 0.0764 | 0.0764 |
| 6 | 100.0000 | 1.2037 | 0.0510 | 0.0510 |
| 7 | 100.0000 | 1.2037 | 0.0340 | 0.0340 |

From this table, we see that the Newton step is feasible(using column 2 and column 4) and is successful(using column 3).

| *Iter* | $x_1$ | $x_2$ | $x_3$ | $s_1$ | $s_2$ | $s_3$ | $m(s)$ | $\|g\|$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 3.0000 | 6.0000 | 4.0000 | 1.7120 | -0.3298 | 0.9799 | 2.3232 | 12.9615 |
| 1 | 4.7120 | 5.6702 | 4.9799 | 0.7348 | -0.2234 | 0.4669 | 1.0672 | 4.0044 |
| 2 | 5.4468 | 5.4468 | 5.4468 | -0.1489 | -0.1489 | -0.1489 | 1.0133 | 0.3567 |
| 3 | 5.2978 | 5.2978 | 5.2978 | -0.0993 | -0.0993 | -0.0993 | 1.0026 | 0.1057 |
| 4 | 5.1986 | 5.1986 | 5.1986 | -0.0662 | -0.0662 | -0.0662 | 1.0005 | 0.0313 |
| 5 | 5.1324 | 5.1324 | 5.1324 | -0.0441 | -0.0441 | -0.0441 | 1.0001 | 0.0093 |
| 6 | 5.0883 | 5.0883 | 5.0883 | -0.0294 | -0.0294 | -0.0294 | 1.0000 | 0.0027 |
| 7 | 5.0588 | 5.0588 | 5.0588 | -0.0196 | -0.0196 | -0.0196 | 1.0000 | 0.0008 |

**Remark 2.11.** *We choose a sufficiently small tolerance to get a more efficient approximate minimizer.*

# Chapter 3

# Numerical Implementations

In this chapter, we report numerical results(implementations) of the two dimensional subspace minimization algorithm for Newton trust region method using MATLAB in order to evaluate its effectiveness and compare with the performance of an alternative method, namely, the dogleg method, and other common methods such as Newton's method and the steepest descent method. We obtain the **practical** algorithm (MATLAB code) of this method by combining **Algorithm 1.3 Algorithm 2.1** and **Algorithm 2.2** together. However, we can use eigen-decomposition(spectral decomposition) of the Hessian matrices H at each iteration to find the smallest eigenvalue instead of **Algorithm 2.1** whenever the problem is not too large. We use some selected test problems. Since the 2D method has global convergence property, we can choose a particular vector as an initial guess so as to make our interpretation easy.

**Notaton:** In our implementation, we use

i) $x_0$ and $\Delta$ to denote the initial guess and trust region radius respectively.

ii) *tol*, *Iter* and *mineig* to denote the error tolerance taken, iteration and minimum eigenvalue respectively.

iii) *toleig* to denote a bound for smallest eigenvalue obtained by the Lanczos method to determine the definiteness of that involved Hessian matrix.

iv) $\eta$ to denote a parameter in $[0, \frac{1}{4})$ which determines the acceptability of a step $s$ as defined in Algorithm 1.3.

v) $\rho$ to denote a parameter used to adjust the trust region radius $\Delta$ as defined in equation 1.3.

vi) NM, SDM, DM and 2DM to denote Newton's, steepest descent, dogleg and two dimensional subspace minimization methods respectively.

We consider the following test problems.

1. Let $x = [x_1 \;\; x_2 \;\; x_3]^T$ and
$f(x) = 1 + (x_1 - x_2)^2 + (x_2 - 5)^4 + (x_3 - x_1)^2$.

$$\min \; f(x) \;\; s.t \;\; x \in \mathbb{R}^3 \qquad (P1)$$

2. Let $x = [x_1 \;\; x_2 \;\; ... \;\; x_{200}]^T$ and
$f(x) = 1 + (x_1 + 1)^2 + (x_2 - x_1)^2 + (x_3 + 3)^2 + ... + (x_{199} + 199)^2 + (x_{200} - x_{199})^2$.

$$\min \;\; f(x) \;\; s.t \;\; x \in \mathbb{R}^{200} \qquad (P2)$$

3. Let $x = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^T$ and
$f(x) = (x_1 - 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^2$

$$\min \;\; f(x) \;\; s.t \;\; x \in \mathbb{R}^4 \qquad (P3)$$

4. Let $x = [x_1 \;\; x_2 \;\; x_3]^T$ and
$f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 + cos(\frac{x_3}{2})$

$$\min \;\; f(x) \;\; s.t \;\; x \in \mathbb{R}^3 \qquad (P4)$$

**Note:** Our interpretation is based on the above test problems and we consider four digits after the decimal point in most numeral values.

Newton's method to solve (1.4) is very fast provided that the Hessian matrix, $H_k$ of the objective function $f$ is positive definite for each k. To see this, consider

$$\min \;\; f(x) \;\; s.t \;\; x \in \mathbb{R}^3, \qquad (P1)$$

where $f(x) = 1 + (x_1 - x_2)^2 + (x_2 - 5)^4 + (x_3 - x_1)^2$.

One can determine $[5 \;\; 5 \;\; 5]^T$ is the only minimizer. Let us see the performances of Newton's, the steepest descent, the two dimensional subspace minimization methods and the dogleg method as well.

Taking $x_0 = [90 \;\; 10 \;\; 0]^T \;\; and \;\; tol = 10^{-6}$, we get the following output from MATLAB using Algorithm 1.2.

| $Iter$ | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 90.000 | 10.0000 | 0.0000 | 15126.0000 | 513.419906 |
| 1 | 8.3333 | 8.3333 | 8.3333 | 124.4568 | 148.148148 |
| 2 | 7.2222 | 7.2222 | 7.2222 | 25.3865 | 43.895748 |
| 3 | 6.4815 | 6.4815 | 6.4815 | 5.8171 | 13.006147 |
| 4 | 5.9877 | 5.9877 | 5.9877 | 1.9515 | 3.853673 |
| 5 | 5.6584 | 5.6584 | 5.6584 | 1.1880 | 1.141829 |
| 6 | 5.4390 | 5.4390 | 5.4390 | 1.0371 | 0.338320 |
| 7 | 5.2926 | 5.2926 | 5.2926 | 1.0073 | 0.100243 |
| 8 | 5.1951 | 5.1951 | 5.1951 | 1.0014 | 0.029702 |
| 9 | 5.1301 | 5.1301 | 5.1301 | 1.0003 | 0.008800 |
| 10 | 5.0867 | 5.0867 | 5.0867 | 1.0001 | 0.002608 |
| 11 | 5.0578 | 5.0578 | 5.0578 | 1.0000 | 0.000773 |
| 12 | 5.0385 | 5.0385 | 5.0385 | 1.0000 | 0.000229 |
| 13 | 5.0257 | 5.0257 | 5.0257 | 1.0000 | 0.000068 |
| 14 | 5.0171 | 5.0171 | 5.0171 | 1.0000 | 0.000020 |
| 15 | 5.0114 | 5.0114 | 5.0114 | 1.0000 | 0.000006 |
| 16 | 5.0076 | 5.0076 | 5.0076 | 1.0000 | 0.000002 |
| 17 | 5.0051 | 5.0051 | 5.0051 | 1.0000 | 0.000001 |
| 18 | 5.0034 | 5.0034 | 5.0034 | 1.0000 | 0.000000 |

Table 3.1: Newton's method for $(P1)$

However, the steepest descent method(Algorithm 1.1) with inexact step length and with data inputs $x_0 = [90 \;\; 10 \;\; 0]^T \;\; and \;\; tol = 10^{-6}, and \;\; Wolfe \;\; constants \;\; c_1 = 0.1, c_2 = 0.5$, requires 20559 iterations to obtain an approximate minimizer $[5.0070 \;\; 5.0070 \;\; 5.0070]^T$ of the exact solution $[5 \;\; 5 \;\; 5]^T$. Some of the iterations are shown below.

| $Iter$ | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 90.0000 | 10.0000 | 0.0000 | 15126.0000 | 513.419906 |
| 1 | 79.3750 | -0.6250 | 5.6250 | 12841.1917 | 936.240691 |
| 2 | 76.9727 | 6.1868 | 6.7773 | 9940.9993 | 342.645817 |
| 3 | 68.1613 | 10.4020 | 11.1646 | 7437.3287 | 575.261551 |
| 4 | 64.5752 | 2.3546 | 12.9457 | 6586.9754 | 319.229926 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 20557 | 5.0070 | 5.0070 | 5.0070 | 1.0000 | 0.000002 |
| 20558 | 5.0070 | 5.0070 | 5.0070 | 1.0000 | 0.000001 |
| 20559 | 5.0070 | 5.0070 | 5.0070 | 1.0000 | 0.000001 |

Table 3.2: The steepest descent method for $(P1)$

Similarly, the two dimensional subspace algorithm with the same initial guess, $x_0 = [90 \;\; 10 \;\; 0]^T$ and $tol = 10^{-6}$, and with $\eta = 0.24$, $\Delta_0 = 1$, $\bar{\Delta} = 100$ and $toleig = tol$, results the following MATLAB output.

| *Iter* | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 90.0000 | 10.0000 | 0.0000 | 15126.0000 | 513.419906 |
| 1 | 89.1826 | 9.5038 | 0.2926 | 14664.9693 | 433.282994 |
| 2 | 90.0587 | 10.9217 | -0.8129 | 15695.5546 | 775.024506 |
| 3 | 88.2928 | 10.0652 | -0.4279 | 14664.5145 | 524.410834 |
| 4 | 84.4765 | 9.1353 | 0.3278 | 13066.2791 | 384.114652 |
| 5 | 86.1618 | 14.3657 | -5.4862 | 18133.9006 | 3164.778080 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 22 | 5.0159 | 5.0159 | 5.0159 | 1.0000 | 0.000016 |
| 23 | 5.0106 | 5.0106 | 5.0106 | 1.0000 | 0.000005 |
| 24 | 5.0071 | 5.0071 | 5.0071 | 1.0000 | 0.000001 |
| 25 | 5.0047 | 5.0047 | 5.0047 | 1.0000 | 0.000000 |

Table 3.3: Two dimensional subspace minimization method for $(P1)$

While taking $x_0 = [90 \;\; 10 \;\; 0]^T$, the objective function of $(P1)$ has positive definite matrices at each iteration. So, in this case, all the three methods(NM, SDM, 2DM) works well as expected. However, as we have seen in the above tables, Newton's results a **more efficient minimizer** with **less number of iterations** compared to the SDM and the 2DM. When we see the SDM, it needs 20559 iterations which is very large compared to 18 *and* 25, iterations needed for NM and the 2DM respectively and even its minimizer $[5.0070 \;\; 5.0070 \;\; 5.0070]^T$ is less close to the exact minimizer $[5 \;\; 5 \;\; 5]^T$ compared to the minimizers $[5.0034 \;\; 5.0034 \; 5.0034]^T$ and $[5.0047 \;\; 5.0047 \; 5.0047]^T$ obtained by NM and the 2DM respectively.

To see the opposite case(with none positive definite Hessian(s)), consider the following.

$$\min \;\; f(x) \;\; s.t \;\; x \in \mathbb{R}^3, \qquad (P4)$$

where $f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 + cos(\frac{x_3}{2})$. Then,the gradient vector of $f$ is

$$g(x_1, x_2, x_3) = \begin{pmatrix} 4(x_1 - 2)^3 + 2(x_1 - 2x_2) \\ -4(x_1 - 2x_2) \\ -\frac{1}{2} sin(\frac{x_3}{2}) \end{pmatrix}.$$

Therefore, $[2 \;\; 1 \;\; 2n\pi], n \in \mathbb{Z}$ are stationary points and are all minimizers except for the case $n = 0$. We can find many points such that the Hessian matrices are not positive define at such points and hence we may have singular, negative definite or indefinite Hessian matrices. That means, the Newton point may **not available** or it **may not be a descent direction** and so Newton's method is not applicable in this case. If the Hessian matrices are indefinite with none zero determinant, then the Newton point exists but it may not be a descent direction which may lead to **a convergence to a stationary point that is not a minimizer**. In order to illustrate such situation,we consider to initial guesses $[0 \;\; 0 \;\; 0]^T$ and $[0 \;\; 0 \;\; \pi]^T$.

Let $x_0 = [0 \;\; 0 \;\; 0]^T$ and $tol = 10^{-6}$. At this point, using MATLAB with Algorithm 1.2, the Hessian matrices are indefinite and has no zero eigenvalue. We obtain the following.

| *Iter* | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 0.000 | 0.0000 | 0.0000 | 17.0000 | 32.0000 |
| 1 | 0.6667 | 0.3333 | 0.0000 | 4.1605 | 9.4815 |
| 2 | 1.1111 | 0.5556 | 0.0000 | 1.6243 | 2.8093 |
| 3 | 1.4074 | 0.7037 | 0.0000 | 1.1233 | 0.8324 |
| 4 | 1.6049 | 0.8025 | 0.0000 | 1.0244 | 0.2466 |
| 5 | 1.7366 | 0.8683 | 0.0000 | 1.0048 | 0.0731 |
| 6 | 1.8244 | 0.9122 | 0.0000 | 1.0010 | 0.0217 |
| 7 | 1.8829 | 0.9415 | 0.0000 | 1.0002 | 0.0064 |
| 8 | 1.9220 | 0.9610 | 0.0000 | 1.0000 | 0.0019 |
| 9 | 1.9480 | 0.9740 | 0.0000 | 1.0000 | 0.0006 |
| 10 | 1.9653 | 0.9827 | 0.0000 | 1.0000 | 0.0002 |
| 11 | 1.9769 | 0.9884 | 0.0000 | 1.0000 | 0.0000 |
| 12 | 1.9846 | 0.9923 | 0.0000 | 1.0000 | 0.0000 |

Table 3.4: Newton's method for $(P4)$

We can see that, NM converges to the stationary point $[1.9846 \quad 0.9923 \quad 0.0000]^T$ which is not a minimizer. We can mitigate this difficulty by using the steepest descent method since it does not use the Hessian matrices but it needs 323 iterations to obtain an approximate minimizer $[1.9716 \quad 0.9858 \quad -6.2832]^T$ with the same initial guess, tolerance and with Wolfe constants $c_1 = 0.1, c_2 = 0.5$. Some of the iterations are

| $Iter$ | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 0.0000 | 0.0000 | 0.0000 | 17.0000 | 32.0000 |
| 1 | 2.0000 | -0.0313 | -0.0313 | 5.2538 | 8.5039 |
| 2 | 1.7422 | 1.0000 | -0.0322 | 1.0708 | 1.0817 |
| 3 | 1.7830 | 0.8711 | -0.0332 | 1.0037 | 0.1634 |
| 4 | 1.7830 | 0.8915 | -0.0343 | 1.0021 | 0.0418 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 319 | 1.9714 | 0.9857 | -6.2832 | -1.0000 | 0.0001 |
| 320 | 1.9715 | 0.9858 | -6.2832 | -1.0000 | 0.0002 |
| 321 | 1.9715 | 0.9857 | -6.2832 | -1.0000 | 0.0002 |
| 323 | 1.9716 | 0.9858 | -6.2832 | -1.0000 | 0.0001 |

Table 3.5: The steepest descent method for $(P4)$

The two dimensional subspace method overcomes the drawbacks of both methods, NM and SDM. For this problem, taking the same initial point and tolerance with $\Delta_0 = 1$, $\eta = 0.24$, $\Delta_{max} = 100$ and $toleig = 10^{-4}$, we get the following.

| *Iter* | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 0.0000 | 0.0000 | 0.0000 | 17.0000 | 32.0000 |
| 1 | 0.6711 | 0.3464 | 0.6555 | 4.0662 | 9.4114 |
| 2 | 1.1193 | 0.5661 | 1.5220 | 1.3260 | 2.7679 |
| 3 | 1.4187 | 0.7127 | 3.4940 | -0.0611 | 0.9332 |
| 4 | 1.4291 | 0.7480 | 7.4939 | -0.7116 | 0.9004 |
| 5 | 1.6023 | 0.8011 | 6.1098 | -0.9712 | 0.2554 |
| 6 | 1.7348 | 0.8674 | 6.2836 | -0.9951 | 0.0746 |
| 7 | 1.8232 | 0.9116 | 6.2832 | -0.9990 | 0.0221 |
| 8 | 1.8822 | 0.9411 | 6.2832 | -0.9998 | 0.0065 |
| 9 | 1.9214 | 0.9607 | 6.2832 | -1.0000 | 0.0019 |
| 10 | 1.9476 | 0.9738 | 6.2832 | -1.0000 | 0.0006 |
| 11 | 1.9651 | 0.9825 | 6.2832 | -1.0000 | 0.0002 |
| 12 | 1.9767 | 0.9884 | 6.2832 | -1.0000 | 0.0001 |

Table 3.6: The two dimensional subspace minimization method for $(P4)$

In addition to fast convergence, the approximate minimizer obtained by 2DM is more efficient compared to that of SDM.

At the second initial guess $[0 \;\; 0 \;\; \pi]^T$, with the same other data inputs as above, the Hessian $H_0$ has an eigenvalue $-1.5308 \times 10^{-17}$ which is numerically singular. So, the Newton point is not available and hence Newton's method is not applicable. The SDM and 2DM need 329 and 11 iterations and yield approximate minimizers $[1.9723 \;\; 0.9861 \;\; 6.2832]^T$ and $[1.9787 \;\; 0.9893 \;\; 6.2832]^T$ respectively. Thus, the two dimensional subspace method is recommended to use instead of both.

Now on, we will compare the performances of the standard dogleg method(DM) and that of the two dimensional subspace minimization method(2DM) using the test problems listed above. First, let us the positive definite case. Consider $(P1)$, that is,

$$\min \quad f(x) \quad s.t \quad x \in \mathbb{R}^3, \qquad (P1)$$

where $f(x) = 1 + (x_1 - x_2)^2 + (x_2 - 5)^4 + (x_3 - x_1)^2$.
The dogleg algorithm with $x_0 = [0 \;\; 1 \;\; 0]^T$, $\Delta_0 = 2$, $\eta = 0.24$, $\Delta_{max} = 100$ and $tol = 10^{-6}$, yields a positive definite Hessian matrices at each iteration and we obtain the following output from MATLAB.

| $Iter$ | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 0.0000 | 1.0000 | 0.0000 | 258.0000 | 254.007874 |
| 1 | 1.0650 | 2.3204 | 1.0594 | 54.1351 | 74.495639 |
| 2 | 2.3412 | 3.1944 | 2.3272 | 12.3565 | 21.903500 |
| 3 | 3.7963 | 3.7963 | 3.7963 | 3.0994 | 6.976448 |
| 4 | 4.1975 | 4.1975 | 4.1975 | 1.4147 | 2.067096 |
| 5 | 4.4650 | 4.4650 | 4.4650 | 1.0819 | 0.612473 |
| 6 | 4.6433 | 4.6433 | 4.6433 | 1.0162 | 0.181473 |
| 7 | 4.7622 | 4.7622 | 4.7622 | 1.0032 | 0.053770 |
| 8 | 4.8415 | 4.8415 | 4.8415 | 1.0006 | 0.015932 |
| 9 | 4.8943 | 4.8943 | 4.8943 | 1.0001 | 0.004721 |
| 10 | 4.9295 | 4.9295 | 4.9295 | 1.0000 | 0.001399 |
| 11 | 4.9530 | 4.9530 | 4.9530 | 1.0000 | 0.000414 |
| 12 | 4.9687 | 4.9687 | 4.9687 | 1.0000 | 0.000123 |
| 13 | 4.9791 | 4.9791 | 4.9791 | 1.0000 | 0.000036 |
| 14 | 4.9861 | 4.9861 | 4.9861 | 1.0000 | 0.000011 |
| 15 | 4.9907 | 4.9907 | 4.9907 | 1.0000 | 0.000003 |
| 16 | 4.9938 | 4.9938 | 4.9938 | 1.0000 | 0.000001 |

Table 3.7: The dogleg method for $(P1)$ (positive definite case)

Using the same data inputs as Table 3.7 above and with $toleig = tol$, the 2DM yields the following.

| *Iter* | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 0.0000 | 1.0000 | 0.0000 | 258.0000 | 254.007874 |
| 1 | 1.1840 | 2.0973 | 1.1807 | 72.8255 | 96.018963 |
| 2 | 2.4633 | 2.9522 | 2.4584 | 18.8228 | 33.383271 |
| 3 | 3.6348 | 3.6348 | 3.6348 | 4.4733 | 10.176951 |
| 4 | 4.0899 | 4.0899 | 4.0899 | 1.6861 | 3.015393 |
| 5 | 4.3933 | 4.3933 | 4.3933 | 1.1355 | 0.893450 |
| 6 | 4.5955 | 4.5955 | 4.5955 | 1.0268 | 0.264726 |
| 7 | 4.7303 | 4.7303 | 4.7303 | 1.0053 | 0.078437 |
| 8 | 4.8202 | 4.8202 | 4.8202 | 1.0010 | 0.023241 |
| 9 | 4.8801 | 4.8801 | 4.8801 | 1.0002 | 0.006886 |
| 10 | 4.9201 | 4.9201 | 4.9201 | 1.0000 | 0.002040 |
| 11 | 4.9467 | 4.9467 | 4.9467 | 1.0000 | 0.000605 |
| 12 | 4.9645 | 4.9645 | 4.9645 | 1.0000 | 0.000179 |
| 13 | 4.9763 | 4.9763 | 4.9763 | 1.0000 | 0.000053 |
| 14 | 4.9842 | 4.9842 | 4.9842 | 1.0000 | 0.000016 |
| 15 | 4.9895 | 4.9895 | 4.9895 | 1.0000 | 0.000005 |
| 16 | 4.9930 | 4.9930 | 4.9930 | 1.0000 | 0.000001 |
| 17 | 4.9953 | 4.9953 | 4.9953 | 1.0000 | 0.000000 |

Table 3.8: The two dimensional subspace minimization method for $(P1)$

Observe that, for this problem, $(P1)$ at $x_0 = [0 \;\; 1 \;\; 0]^T$ the 2DM results an approximate minimizer $[4.9953 \;\; 4.9953 \;\; 4.9953]^T$ which is closer to the exact minimizer $[5 \;\; 5 \;\; 5]^T$ than an approximate minimizer $[4.9938 \;\; 4.9938 \;\; 4.9938]^T$ obtained by the DM with a negligible extra number of iteration, one.

From **Algorithm 1.4**, for iterations with none positive definite Hessian matrices,the DM takes its step to be the Cauchy point and so it is the similar to the SDM with a particular choice of step length. So, in such a case the DM may have slow convergence property. The two dimensional subspace minimization method mitigates this difficulty by determining the smallest eigenvalue and a corresponding eigenvector to form its 2D subspace and search on it. To illustrate this case, we can consider $(P4)$, that is,

$$\min \;\; f(x) \;\; s.t \;\; x \in \mathbb{R}^3,$$

where $f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 + cos(\frac{x_3}{2})$. Then, at the initial guess $x_0 = [0 \;\; 0 \;\; \frac{\pi}{5}]^T$ with $\Delta_0 = 1$, $\eta = 0.24$, $\Delta_{max} = 100$ and $tol = toleig = 10^{-6}$, the initial Hessian matrix $H_0$ has eigenvalues $50.3776, 7.6224$ and $-0.2378$ and hence it is indefinite. Moreover, the dogleg algorithm needs 24 iterations to converge and it results indefinite Hessian matrices up to iteration 13 which makes it slow. We can see the following table from MATLAB.

| *Iter* | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 0.0000 | 0.0000 | 0.6283 | 16.9511 | 32.000373 |
| 1 | 0.6400 | 0.0000 | 0.6314 | 4.7811 | 9.764334 |
| 2 | 1.0876 | 0.1216 | 0.6388 | 2.3555 | 4.030631 |
| 3 | 1.4850 | 0.7334 | 0.6672 | 1.0155 | 0.557667 |
| 4 | 1.6251 | 0.7528 | 0.7107 | 0.9716 | 0.516928 |
| 5 | 1.6413 | 0.8378 | 0.7416 | 0.9498 | 0.315305 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 21 | 1.9829 | 0.9914 | 6.2832 | -1.0000 | 0.000020 |
| 22 | 1.9886 | 0.9943 | 6.2832 | -1.0000 | 0.000006 |
| 23 | 1.9924 | 0.9962 | 6.2832 | -1.0000 | 0.000002 |
| 24 | 1.9949 | 0.9975 | 6.2832 | -1.0000 | 0.000001 |

Table 3.9: The dogleg method for $(P4)$ (indefinite case)

However, the 2DM method needs only 15 iterations with the same input data as shown below. We can see that, the distance from $x^* = [2\ 1\ 2\pi]^T$ to the approximated

| *Iter* | $x_1$ | $x_2$ | $x_3$ | $f(x)$ | $\|g\|$ |
|---|---|---|---|---|---|
| 0 | 0.0000 | 0.0000 | 0.6283 | 16.9511 | 32.000373 |
| 1 | 0.6708 | 0.3457 | 1.2844 | 3.9223 | 9.418275 |
| 2 | 1.1182 | 0.5646 | 3.2214 | 0.5649 | 2.799034 |
| 3 | 1.3367 | 0.5726 | 7.2154 | -0.6631 | 1.260659 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 12 | 1.9842 | 0.9921 | 6.2832 | -1.0000 | 0.000016 |
| 13 | 1.9894 | 0.9947 | 6.2832 | -1.0000 | 0.000005 |
| 14 | 1.9930 | 0.9965 | 6.2832 | -1.0000 | 0.000001 |
| 15 | 1.9953 | 0.9977 | 6.2832 | -1.0000 | 0.000000 |

Table 3.10: The 2DM for $(P4)$

minimizer obtained by the DM and 2DM are 0.0056 and 0.0052 respectively and hence the 2DM results a more efficient approximate with **less number of iterations** compared to the DM based on this problem. We can infer Algorithm 1.4 to see that we face similar difficulties for such cases(problems involving indefinite Hessian).

Let us see the performances of the 2DM and the DM based on the test problems listed above. Let $x^*$ denote exact minimizer and $x_{2D}$ and $x_D$ denote approximate minimizer obtained by the 2DM and DM respectively and n be the dimension of a problem. For both methods, we consider $tol = 10^{-6}$, $\Delta_0 = 1$, $\Delta_{max} = 100$ and $\eta = 0.24$ for each test problems. Denote the number of iterations needed for DM and 2DM by $I_D$ and $I_{2D}$ respectively, and the problem by $(Pi)$, for $i = 1, 2, 3, 4$.

| $(Pi)$ | $n$ | $x^*$ | $x_0$ | $x_D$ | $x_{2D}$ | $d_1$ | $d_2$ | $I_D$ | $I_{2D}$ |
|---|---|---|---|---|---|---|---|---|---|
| (P1) | 3 | $\begin{pmatrix}5\\5\\5\end{pmatrix}$ | $\begin{pmatrix}0\\0\\0\end{pmatrix}$ | $\begin{pmatrix}4.9938\\4.9938\\4.9938\end{pmatrix}$ | $\begin{pmatrix}4.9946\\4.9946\\4.9946\end{pmatrix}$ | 0.0107 | 0.0094 | 17 | 18 |
| | | | $\begin{pmatrix}20\\30\\40\end{pmatrix}$ | $\begin{pmatrix}5.0054\\5.0054\\5.0054\end{pmatrix}$ | $\begin{pmatrix}5.0050\\5.0050\\5.0050\end{pmatrix}$ | 0.0094 | 0.0086 | 23 | 24 |
| (P2) | 200 | $a$ | $b$ | $c$ | $d$ | $1\times10^{-6}$ | $1\times10^{-6}$ | 127 | 79 |
| (P3) | 4 | $\begin{pmatrix}0\\0\\0\\0\end{pmatrix}$ | $\begin{pmatrix}12\\21\\23\\16\end{pmatrix}$ | $\begin{pmatrix}0.0013\\-0.0001\\0.0019\\0.0019\end{pmatrix}$ | $\begin{pmatrix}0.0007\\-0.0001\\0.0017\\0.0017\end{pmatrix}$ | 0.0030 | 0.0024 | 26 | 24 |
| | | | $\begin{pmatrix}1\\2\\3\\4\end{pmatrix}$ | $\begin{pmatrix}0.0001\\-0.0000\\0.0019\\0.0019\end{pmatrix}$ | $\begin{pmatrix}0.0003\\-0.0000\\0.0014\\0.0014\end{pmatrix}$ | 0.0027 | 0.0020 | 18 | 19 |
| (P4) | 3 | $\begin{pmatrix}2\\1\\2\pi\end{pmatrix}$ | $\begin{pmatrix}0\\-1\\7\pi\times10^{-3}\end{pmatrix}$ | $\begin{pmatrix}2.0058\\1.0029\\6.2832\end{pmatrix}$ | $\begin{pmatrix}1.9950\\0.9975\\6.2832\end{pmatrix}$ | 0.0064 | 0.0056 | 68 | 15 |
| | | | $\begin{pmatrix}-2\\0.5\\\pi\times10^{-1}\end{pmatrix}$ | $\begin{pmatrix}1.9938\\0.9969\\6.2832\end{pmatrix}$ | $\begin{pmatrix}1.9951\\0.9975\\6.2832\end{pmatrix}$ | 0.0070 | 0.0055 | 40 | 17 |

Table 3.11: The two dimensional subspace minimization versus dogleg methods

Where $d_1 = \|x^* - x_D\|$, $d_2 = \|x^* - x_{2D}\|$

$a = [-1 \quad -1 \quad -3 \quad -3 \quad ... \quad -197 \quad -197 \quad -199 \quad -199]^T$ $(exact\ minimizer\ of\ (P2))$

$b = [0 \quad -2 \quad -2 \quad -4 \quad -4 \quad ... \quad -198 \quad -198 \quad -200]^T$ $(initial\ guess)$

$c \approx a \quad (approximate\ solution\ by\ DM)$

$d \approx a \quad (approximate\ solution\ by\ 2DM)$

From Table 3.11 above, if we consider row 1 and row 2, the Hessian matrices are $PD$ at each iteration in both methods, namely, DM and 2DM. For these rows, the 2DM results a relatively efficient approximate minimizer (see column 7 and 8) with a negligible extra number of iterations(see column 9 and 10) compared to the DM. From row 2, one can see that the 2DM is better for large scale problems (as it reduces the dimension of the problem from n to 2) than the DM. In row 4, both initial guesses produce an **indefinite** Hessian matrices and one can see that the 2DM is superior over the DM in this case (see columns 7, 8, 9 and 10 ).

# Conclusion

This thesis has described the two dimensional subspace minimization(2DM) approach for Newton trust region method to solve an unconstrained optimization problems. I have implemented my own MATLAB code for 2D subspace minimization method to solve the corresponding subspace subproblem (see Annex C).

In section (2.3), we have seen that the subspace of this method is spanned by the steepest descent and the full(or inexact) Newton directions to have global and fast convergence property respectively and this convergence property is achieved(see Theorem 2.9 and Theorem 2.10). This method reduces the dimension of the subproblem from n to 2 and then to finding roots of a fourth degree polynomial(see section (2.3.2)). Newton's method is better than the steepest descent and the 2DM when the Hessian matrices are positive definite(PD) at each iteration(see Tables 3.1, 3.2 and 3.3). But, Newton's method is not applicable when the Hessian matrices are not positive definite because in such a case the Newton point may not available or it may be not a descent direction (see Table 3.4). Since the steepest descent method(SDM) doesn't use Hessian matrices, it mitigates the difficulty of Newton's method caused by definiteness the Hessian matrices. However, it needs an unacceptable number of iterations(see Table 3.5). Although the standard dogleg method also overcomes the drawbacks of Newton's method(see Table 3.9 and Table 3.11), still it uses the Cauchy point when the Hessian matrices are **not positive definite**. That is, the dogleg method is same as the steepest descent method with a particular choice of step length for the case when the Hessian matrices are not positive definite. Based on the test problems, Table 3.11 shows that the **2DM** is better than the dogleg method as it modifies the convergence property. Table 3.11 has also shown that 2DM is better for large problem than the DM. Thus, we can conclude that **2DM is superior** over Newton's, the steepest descent, and the dogleg methods as expected. However, for large scale problems, defining the objective function, its gradient vector and Hessian matrix for MATLAB code may be difficult.

One can emphasize on the following for further work.

i. How can we decrease or avoid (if possible ) the cases that take the Cauchy point as a minimizer in 2D subspace minimization method to increase the rate of convergence?

ii. How can we define any large scale problem in the MATLAB code of 2D subspace minimization method?

# Annex A

MATLAB Code For Newton's and the steepest descent method

```
function Newton
% First,we have to define the objective function,its gradient vector and Hessian
% matrix as indicated at the bottom of this program Or alternatively
% We can save the objective function,its gradient vector and Hessian matrix
% using file names "f.m" , "grad.m" and "Hessian.m" respectively.
% We must have an appropriate choice of parameters.
clc;
clear all;
x=input('Enter the initial guess x=');
tol=input('Enter your error tolerance,tol=')
      obj=f(x);     %obj is the objective function to be minimized.
       g=grad(x);   % gragient of f
      H=Hessian(x);       %H is the hessian matrix of obj
       k=0;                   % k = # iterations
 if det(H)==0
      disp('Wrong initial guess.')
      return
  end
 while norm(g)>tol
       obj=func(x);
       g=grad(x);
       H=hessian(x);
       X=-inv(H)*g;
       x=x+X;
      k=k+1;
    end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function fun = f(x)
     fun="Our objective_function"; % We have to define
                                   %the objective function to be minimized here.
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 function y = grad(x)
      y="Gradient(f)" ; % We have to define
                        % the gradient vector of f here.

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    function H=Hessian(x)
        H=Hessian(f);  % We have to define
                       %the Hessian matrix of f here.
```

```
function steepestd
clc;
clear all;
x=input('enter the initial column vector x: ');
  c1=input('Enter 0<c1<1,c1=');
  c2=input('Enrer c1<c2<1,c2=');
  obj=f(x);      % func is the objective function to be minimized.
  g=grad(x);         % obj is the objective function value.
  k=0;                    % k = # iteration
  while  norm(g) > tol    %Tolerance=10^(-6).
    d = -g;                   % d is steepest descent direction
    t = 1;
    newobj = f(x + t*d);
    while t~=0&&(newobj-obj)/t > c1*g'*d
         t = t*c2;
         newobj = f(x + t*d);
    end
       x = x + t*d;
      obj=newobj;
      g=grad(x);
      k = k + 1;
  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     function fun = f(x)
     fun="Our objective_function"; % We have to define
                                   %objective function to be minimized here.
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        function y = grad(x)
         y="Gradient(f)" ; % We have to define the gradient vector of f here.
```

# Annex B

MATLAB Code For the Newton dogleg method

```
function dogleg
% First,we have to define the objective function,its gradient vector and Hessian
% matrix as indicated at the bottom of this program Or alternatively
% We can save the objective function,its gradient vector and Hessian matrix
% using file names "f.m" , "grad.m" and "Hessian.m" respectively.

clc;
clear all;
n=input('Enter the dimension of the problem,n=');
x_0=input('Enter the initial nx1 column vector x_0 = ');   % Initial guess.
 % We have to take a sufficiently small positive value.
tol=input('Enter an error tolerance,tol=');
eta=input('Enter a parameter in (0,0.25),eta='); % a parameter used to update x
% a positive parameter as a maximum upper bound to our trust region.
delmax=input('Enter the maximum TR bound,delmax=');
% We have to take a small positive parameter(not too small).
del=input('Enter a trust radius in (0,delmax),del=');
obj=f(x); g=grad(x);H=hess(x);
  k=0;
   while norm(g)>tol
       obj=f(x); g=grad(x);H=hess(x);
if (g'*H*g)<=0
         tau=1;
else
   tau=min((norm(g))^3/(del*g'*H*g),1);
end

sc=-((tau*del)/norm(g))*g; %Cauchy point

if min(eig(H))>0
    H=H;
   sn=-inv(H)*g;
   if norm(sN)<=del
       sk=sN;   %Newton point
   else
       if norm(sc)==del
           sk=sc;
       else
  b=sc'*(sc-sN);
  c= norm(sc-sN)^2;
  d=del^2-norm(sc)^2;
 a1=(b+sqrt(b^2+c*d))/c;
 a2=(b-sqrt(b^2+c*d))/c;
```

```
  if a1 >=0
   a= a1;
  else
    a=a2
  end
  sk=sc+a*(sn-sc);
       end
   end
    else
      sk=sc;
      tk=del*tau/norm(g);
end

   mk=f(x)+g'*sk+0.5*sk'*H*sk;

  rho=(f(x)- f(x+sk))/(f(x)-mk);

  norm(sk);
  if rho<0.25
     newdel=0.25*del;
  elseif rho>=0.75 && norm(sk)==del
      newdel=min(2*del,delmax);
       else
          newdel=del;
  end
        del=newdel;
        if rho>=eta
            newx=x+sk;

        else
            newx=x;
        end
        x=newx;
      k=k+1;
   end
```

# Annex C

MATLAB Code For Two Dimensional Subspace Minimization

```
% The subspace minimization approach for Newton trust-region method.
% First,we have to define the objective function,its gradient vector and Hessian
% matrix as indicated at the bottom of this program Or alternatively
% We can save the objective function,its gradient vector and Hessian matrix
% using file names "f.m" , "grad.m" and "Hessian.m" respectively.

function two_Dsubspace(Ageze A.)
clc;
clear all;
n=input('Enter the dimension of the problem,n=');
x_0=input('Enter the initial nx1 column vector x_0 = '); % Initial guess.
 % We have to take a sufficiently small positive value.
tol=input('Enter an error tolerance,tol=');
eta=input('Enter a parameter in (0,0.25),eta='); % a parameter used to update x
% A sufficiently small positive parameter used to determine
% definitness of the Hessian matrices H.
toleig=input('Enter a parameter,toleig=');
 % a positive parameter as a maximum upper bound to our trust region.
delmax=input('Enter the maximum TR bound,delmax=');
% We have to take a small positive parameter(not too small).
del=input('Enter a trust radius in (0,delmax),del=');

    obj=f(x_0);     % f is the objective function to be minimized and
                    %obj is the objective value.
    g=grad(x_0);   % gradient vector of f.
    H=Hessian(x_0);       %H is the Hessian matrix of the
                          %objective fumction.
  count=0;           % iteration counter
 while norm(g)>tol
    obj=f(x);
    g=gra(x);
    H=Hessian(x);
if (g'*H*g)<=0
    tau=1;
else
    tau=min((norm(g))^3/(del*g'*H*g),1);
end
sc=-((tau*del)/norm(g))*g;        % The Cauchy point.

% Lanzos iteration to approximate the smallest eigenvalue and corresponding
% eigenvector(direction of negative curvature.)

    k=length(H);
```

```
    r=rand(k,1);
    Q(:,1)=r/norm(r);
if k==1
    beta=[];
end
for j=1:k
  v=H*Q(:,j);
  alpha(j)=Q(:,j)'*v;
  v=v-alpha(j)*Q(:,j);
  if j>1
    v=v-beta(j-1)*Q(:,j-1);
  end
  if j<k
    beta(j)=norm(v);
    Q(:,j+1)=v/beta(j) ;
  end
end
   T=Q'*H*Q;  % The tridiagonal matrix approximating extreme
              %eigenvalues and eigenvectors of H.
   [V,D]=eig(T);
for i=1:k

if min(diag(D))==D(i,i)
d=V(:,i);    % eigenvector corresponding to the smallest eigenvalue.
end
end
 lambda_1=min(diag(D));          % the smallest eigenvalue of H.


%Check definiteness of H
if lambda_1>toleig             % H is PD
    sN=-inv(H)*g;           %Full Newton step
    if norm(sN)<=del
        s=sN;
        else
%Solving the reduced subspace subproblem(i.e the 4th degree polynomial)

       s1=g/norm(g);
        if sN==(sN'*s1)*s1
          s=sc;
           else
        s2=(sN-(sN'*s1)*s1)/norm(sN-(sN'*s1)*s1);      % Normalization
        W=[s1 s2];
        G=W'*H*W;
        U=g'*W;
        if min(eig(G))>toleig
```

```
            s2N=-inv(G)*U';                    % Newton point for the
                                               % reduced subproblem.
            if norm(s2N)<=del
                s=s2N;
            else
% Notation
      a1=G(1,1);
      a2=G(1,2);
      a3=G(2,2);
      a4=U(1,1);
      a5=U(1,2);
%Coefficients of the fourth degree polynomial.
c1=del*(a4-a2*del);
c2=2*del*(del*(a3-a1)+a5);
c3=6*del^2*a2;
c4=2*del*(del*(a3-a1)+a5);
c5=del*(-a2*del-a4);
% The fourth degree polynomial.
poly=[c1,c2,c3,c4,c5];           % the fourth degree polynomial.
r=roots(poly);
r=r(imag(r)==0) % it saves only real roots of poly.
j=length(r);
 for i=1:j
 a(i)=2*del*r(i)/(1+r(i)^2);
 b(i)=del*(1-r(i)^2)/(1+r(i)^2); % the parametrization used.
 pp=[a(i) b(i)]';
 mr = f(x)+U*p + 0.5*p'*G*p % The REDUCED QUADRATIC MODEL of f.
 end
 if min(mr)==mr
    q=[a(i) b(i)]'; % a root resulting the smallest value of mr
 end
 s=W*q;             % Minimizer of the reduced subspace subproblem.

               end              % if
           end              % if
        end          % if
    end              % if
elseif abs(lambda_1) < toleig    % H is numerically singular.
   s=sc;

else               % H is indefinite
    MN=-inv(H+2*lambda_1*eye(length(H)))*g;
    if norm(MN)<=del
    gamma=-MN'*d+sqrt((MN'*d)^2+del^2-(norm(MN))^2);
    v=gamma*d;
    s=MN+v;
```

```
    else
        s1=g/norm(g);
        if MN==(MN'*s1)*s1
          s=sc;
           else
        s2=(MN-(MN'*s1)*s1)/norm(MN-(MN'*s1)*s1);
        W=[s1 s2];
        G=W'*H*W;
        U=g'*W;
         if min(eig(G))>toleig
            s2N=-inv(G)*U';
            if norm(s2N)<=del
                s=s2N;
            else
       a1=G(1,1);
       a2=G(1,2);
       a3=G(2,2);
       a4=U(1,1);
       a5=U(1,2);
%Coefficients
     c1=del*(a4-a2*del);
     c2=2*del*(del*(a3-a1)+a5);
     c3=6*del^2*a2;
     c4=2*del*(del*(a3-a1)+a5);
     c5=del*(-a2*del-a4);
% The fourth degree polynomial.
% poly4=c1*f4(t(n)) + c2*f3(t(n)) +  c3*f2(t(n)) + c4*f1(t(n)) + c5;
     c=[c1,c2,c3,c4,c5]; % the polynomial poly4
     r=roots(c);
     r=r(imag(r)==0)
     j=length(r);
 for i=1:j
 a(i)=2*del*r(i)/(1+r(i)^2);b(i)=del*(1-r(i)^2)/(1+r(i)^2);pp=[a(i) b(i)]';
 mr = f(x)+U*pp + 0.5*pp'*G*pp;   % The REDUCED QUADRATIC MODEL.
 end
if min(mr)==mr
   p=[a(i) b(i)]';
end
s=W*p;
            end     %if
         end        %if
        end        %if
    end            %if
end               %if
   z=x+s;
  mk=f(x)+g'*s+0.5*s'*H*s; % Evaluating the quadratic model.
```

```
if (f(x)-mk)~=0
rho=(f(x)-f(z))/(f(x)-mk) %The ratio of actual reduction to predicted reduction.
% Update del
if rho<0.25
    newdel=0.25*del;
    del=newdel;
else

    if rho>0.75 && norm(s)<=del
        newdel=min(2*del,delmax);
        del=newdel;
    else
        del=del;
    end
end

%Updete x
if rho>eta
    newx=x+s;
   x=newx;
else
   x=x;
end      %if
end      %if
count=count+1;
 end     %while

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% We have to define the objective function to be minimized here.
     function fun = f(x)
     fun="Our objective_function";
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % We have to define the gradient vector of f here.
       function y = grad(x)
      y="Gradient(f)" ;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % We have to define the Hessian matrix of f here.
     function H=Hessian(x)
        H=Hessian(f);
```

# Bibliography


[1] Andrew R.Conn,Nicholas I.M.Goulg,Philippe L.Toint,Trust region methods,2000

[2] B.N. Parlett,The Symmetric Eigenvalue Problem (Prentice-Hall, Englewood Cliffs, NJ, 1980).

[3] C.T Kelley,Iterative methods for optimization,1999

[4] G. A. Shultz, R. B. Schnabel and R.H. Byrd, "A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties",1984

[5] J. Nocedal, S. Wright, Numerical Optimization, second edition, 2006

[6] Jennifer B. Erway and Philip E. Gill,A subspace minimization method for the trust region step,2009

[7] M.J.D. Powell, "A new algorithm for unconstrained optimization," in: J.B. Rosen, O. L. Mangasarian and K. Ritter, eds.,Nonlinear Programming (Academic Press, New York, 1970)

[8] R. H. Byrd, R. B. Schnabel, and G. A. Schultz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. Technical report, University of Colorado, 1986.

[9] Wenyu Sun,Optimization theory and methods nonlinear programming,2006

[10] W. W. Hager, Minimizing a quadratic over a sphere, SIAM J. Optim., 2001