SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Convergence Analysis of Quasi Newton methods and on the relation to Conjugate Gradient

av

Axel Olsson

2025 - No K28

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Convergence Analysis of Quasi Newton methods and on the relation to Conjugate Gradient

Axel Olsson

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Yishao Zhou

2025

Contents

Sammanfattning

Den här uppsatsen handlar om numerisk optimering, med särskilt fokus ligger på en quasi Newton metod kallad mL-BFGS. Inledningsvis presenteras grundläggande teori inom optimering, varefter en ingående analys av line search metoder genomförs. Ett bevis för global konvergens under standardantaganden presenteras för mL-BFGS, samt en detaljerad spektralanalys av den inversa Hessian-approximationen genomförs. Ett nytt resultat presenteras som visar hur egenvärdena hos starkt konvexa kvadratiska funktioner kan utnyttjas vid exakt line search för L-BFGS, vilket under vissa förutsättningar innebär att metoden tar identiska steg som conjugate gradient metoden.

Ett antal numeriska experiment genomförs där fokus ligger på att jämföra mL-BFGS med andra välkända metoder, speciellt då gradienten är inexakt, vilket genomförs genom att lägga till ett artificiellt brus. Experimenten undersöker även hur konditioneringen av Hessian matrisen påverkar konvergensegenskaperna hos de olika metoderna. Resultaten visar att mL-BFGS är en robust metod då gradienterna är inexakta och införandet av momentum ökar stabiliteten. Detta visar potential hos metoden för storskaliga optimeringsproblem, där det kan vara föredraget att stokastiskt estimera gradienten.

Abstract

This thesis investigates the convergence properties of a quasi Newton optimization method called mL-BFGS. A proof of global convergence under the standard assumptions for quasi Newton methods is presented. A new theoretical result is presented that extends the previous known relation between quasi Newton methods and Conjugate Gradient method. It shows in particular that if the memory of L-BFGS is defined to the number of unique eigenvalues of a strongly convex quadratic functions Hessian, and exact line search is used, the L-BFGS have identical iterates as Conjugate Gradient, for a certain initialization.

The convergence properties of the mL-BFGS algorithm is analyzed analytically as well as through numerical experiments. The experiments include comparisons with other popular first order methods, Nesterov's Accelerated Gradient Descent and the Heavy Ball method. The experiments focuses on how the methods compare, especially when the gradients are inexact, where artificial noise has been added to the exact gradient. The results of the experiments shows that momentum has a positive effect on convergence when inexact gradients are used, providing a more robust behavior compared with the other methods. Furthermore, the experiments show that the mL-BFGS is also more stable for problems where the Hessian is ill-conditioned and the gradients are inexact. This shows potential for the mL-BFGS in large scale stochastic optimization, where the gradients are approximated, which introduces noise.

1 Introduction

Numerical Optimization is a branch of mathematics that studies methods of finding optimal solutions for a given problem using numerical methods and algorithms. It plays an important role in various engineering applications, the methods are independent of the application as long as the problem can be formulated as an optimization problem. In recent years the rise of large language models and other deep neural networks have increased the increased scale of the objective functions that are optimized, which motivates the need for more efficient and stable algorithms.

This section reviews some of the background material required to follow the thesis. A brief introduction to unconstrained optimization is given and some of the notations used throughout the thesis are introduced. Global and local optimality conditions are later derived, followed by some classical results from convex analysis. Those results highlight some important properties of convex functions, an important class of functions that provide useful analytical tools used in later sections. Finally, a detailed description of line search methods is presented, a class of algorithms which the mL-BFGS as well as the other methods presented in this thesis are members of.

1.1 Unconstrained optimization

Let $f : \Sigma \to \mathbb{R}$ be a given objective function, where $\Omega \subseteq \mathbb{R}^n$ is the domain that one wants to find the minimum or maximum value of f over. These types of problems are typically divided into two categories, constrained and unconstrained problems, depending on the requirements on $x \in \Omega$. In unconstrained optimization, the only constraint on the set of feasible solutions is that they must belong to the domain of f i.e. $x \in \Omega$, while in constrained optimization the set of feasible solutions may be restricted to a subset of Ω. This thesis considers only unconstrained optimization problems, although the methods used in constrained optimization are similar [14]. Throughout the thesis the words *minimization* and *maximization* will be used to describe the type of optimization that is performed, and it should be noted that any problem may be rewritten as a *min* or *max* by simply evaluating $-f$ instead of f. The optimization problem may be written

$$\min_{x \in \Omega} f(x) \quad \text{or} \quad \max_{x \in \Omega} f(x) \tag{1.1}$$

depending on what is to be found. Any feasible $x \in \Omega$ that is a solution of (1.1) will throughout the thesis be denoted x^*, which depending on the problem satisfies

$$x^* \in \arg\min_{x \in \Omega} f(x) \quad \text{or} \quad x^* \in \arg\max_{x \in \Omega} f(x). \tag{1.2}$$

The next subsection introduces some mathematical properties that can be used to check if x is potentially is a solution of (1.1).

1.1.1 Characterization of a solution

In the cases where $\Omega \subseteq \mathbb{R}$ or $\Omega \subseteq \mathbb{R}^2$ it might be possible to determine the minimum of the function graphically. It is however not uncommon that the objective function is defined in higher dimensions, which requires other tools to characterize a solution to the problem (1.1). One example of this is neural networks where the number of parameters can be in billions and even larger [17].

It should be noted that the conditions that are presented below will typically be either a necessary or sufficient condition when the function is continuously differentiable. Lemma 1.1 is perhaps the one characterization that is most used throughout this paper as well as in the literature. For some functions it is a sufficient condition, but not for all functions. Firstly it will be presented in a general setup with less restrictive assumptions on the objective function.

Lemma 1.1 (First order necessary condition)**.** *If $x^* \in \mathbb{R}^n$ is a local minimizer of a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, then $\nabla f(x^*) = 0$.*

Proof. Given that x^* is a local minimizer it is true by definition that for all $y \in \mathbb{R}^n$ close enough to x^* that

$$f(x^*) \leq f(y) \tag{1.3}$$

For any $y \in \mathbb{R}^n$ in a neighborhood of x^* satisfying $\|y - x^*\| \leq r$ for some $r > 0$ where (1.3) is true. Since f is continuously differentiable a linear approximation of f at x^* may be written

$$f(y) = f(x^*) + (y - x^*)^T \nabla f(x^*) + o(\|y - x^*\|), \tag{1.4}$$

where $o(r)$ is defined as

$$\lim_{r \to 0^+} \frac{1}{r} o(r) = 0 \tag{1.5}$$
$$r(0) = 0.$$

Combining (1.3) and (1.4) yields

$$(y - x^*)^T \nabla f(x^*) + o(\|y - x^*\|) \geq 0. \tag{1.6}$$

Dividing (1.6) with $\|y - x^*\|$ for any $y \neq x^*$ we obtain the expression

$$\frac{1}{\|y - x^*\|}(y - x^*)^T \nabla f(x^*) + \frac{o(\|y - x^*\|)}{\|y - x^*\|} \geq 0, \tag{1.7}$$

which together with (1.5) implies that for any $s \in \mathbb{R}^n$

$$s^T \nabla f(x^*) \geq 0. \tag{1.8}$$

Consider $s = -h\nabla f(x^*)$ for some $h > 0$, substituting this s in (1.8) yields

$$-h\|\nabla f(x^*)\|^2 \geq 0, \tag{1.9}$$

where $\|\nabla f(x^*)\|^2 \geq 0$ by definition [1]. Thus (1.8) can only be true if $\nabla f(x^*) = 0$. □

Lemma 1.2 (Second order necessary condition)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable. If x^* is a local minimizer then $\nabla^2 f(x^*) \succeq 0$.*

Proof. Since it is known that x^* is a local minimizer, according to Taylor there must exist an $r > 0$ such that for any $z \in \mathbb{R}^n$ that fulfills $\|z - x^*\| \leq r$. Let $\langle \cdot, \cdot \rangle$ denote the Euclidean inner product in $\mathbb{R}^n$ and we can write

$$f(z) = f(x^*) + \langle \nabla f(x^*), z - x^* \rangle + \frac{1}{2}\langle \nabla^2 f(x^*)(z - x^*), z - x^* \rangle + o(\|z - x^*\|^2) \geq f(x^*). \tag{1.10}$$

From Lemma 1.1 it is known that $\nabla f(x^*) = 0$ which together with (1.10) yields

$$f(x^*) + \frac{1}{2}\langle \nabla^2 f(x^*)(z - x^*), z - x^* \rangle + o(\|z - x^*\|^2) \geq f(x^*). \tag{1.11}$$

Subtracting $f(x^*)$ from all sides of (1.11) and dividing by $\|z - x^*\|^2$ assuming $z \neq x^*$, the inequality can be rewritten

$$0 \leq \frac{1}{2\|z - x^*\|^2}\langle \nabla^2 f(x^*)(z - x^*), z - x^* \rangle + \frac{o(\|z - x^*\|^2)}{\|z - x^*\|^2}. \tag{1.12}$$

From (1.5) it is known that the last term will tend toward zero as $z \to x^*$. Hence, evaluating as $z \to x^*$ and neglecting constants yields that, for any direction s

$$\langle \nabla f(x^*)s, s \rangle \geq 0, \tag{1.13}$$

which implies that $\nabla^2 f(x^*) \succeq 0$ [1]. □

For any general continuously differentiable function Lemma 1.2 and 1.1 is not enough to guarantee that x^* is a local minimum [12]. In order to show that x^* is guaranteed to be a local minima the following sufficient condition is required.

Lemma 1.3 (Sufficient conditions for local optimal point)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable on $\mathbb{R}^n$. If a point $x^* \in \mathbb{R}^n$ satisfies both*

$$\nabla f(x^*) = 0 \quad \text{and} \quad \nabla^2 f(x^*) \succ 0 \tag{1.14}$$

then x^* is a strict local minimizer.

Proof. Since f is twice continuously differentiable we may according to Taylor's Theorem approximate f in a neighborhood of x^* as a quadratic function. Using the definition of the error term (1.5) and letting $z \in \mathbb{R}^n$ be any point close enough to x^* it is true that

$$f(z) = f(x^*) + \langle \nabla f(x^*), z - x^* \rangle + \frac{1}{2}\langle \nabla^2 f(x^*)(z - x^*), z - x^* \rangle + o(\|z - x^*\|^2). \tag{1.15}$$

Given that $\nabla f(x^*) = 0$ it may be simplified since one of the inner products vanishes

$$f(z) = f(x^*) + \frac{1}{2}\langle \nabla^2 f(x^*)(z - x^*), z - x^* \rangle + o(\|z - x^*\|^2). \tag{1.16}$$

Nesterov [12] provides an interesting proof to specifically show the existence of a non-zero radius r surrounding x^* in which, given the assumptions of x^* f must attain a *strict* minimum. From the definition 1.5 it is known that

$$\lim_{r \to 0} \frac{o(r^2)}{r^2} = 0, \tag{1.17}$$

from which it is known that there must exist an $\epsilon > 0$ such that for any positive $r \leq \epsilon$

$$\frac{o(r^2)}{r^2} \leq C, \tag{1.18}$$

where C is some constant. Given (1.18) it must also be true when taking the absolute value

$$\left| \frac{o(r^2)}{r^2} \right| \leq |C| \quad \to \quad |o(r^2)| \leq r^2 |C|. \tag{1.19}$$

Using this limit and the fact that the Hessian at x^*, $\nabla^2 f(x^*) \succ 0$ which by definition [1] for any non-zero vector $s \in \mathbb{R}^n$ fulfills

$$\langle \nabla^2 f(x^*)s, s \rangle > 0, \tag{1.20}$$

it is possible to relate the constant C to the eigenvalues of the Hessian $\nabla^2 f(x^*)$ [12]. Since it is known (1.20) that $\nabla^2 f(x^*) \succ 0$, all eigenvalues are strictly greater than zero, and furthermore that the smallest transformation of a vector $s \in \mathbb{R}^n$ through $\nabla^2 f(x^*)$ may be expressed by

$$\langle \nabla^2 f(x^*)s, s \rangle \geq \lambda_{\min}(\nabla^2 f(x^*))\langle s, s \rangle. \tag{1.21}$$

Using this smallest eigenvalue Nesterov defines the constant $|C|$

$$|C| = \frac{1}{4}\lambda_{\min}(\nabla^2 f(x^*)). \tag{1.22}$$

Using (1.22) in (1.19), we may for clarity express

$$-r^2 \frac{1}{4}\lambda_{\min}(\nabla^2 f(x^*)) \leq o(r^2) \leq r^2 \frac{1}{4}\lambda_{\min}(\nabla^2 f(x^*)), \tag{1.23}$$

that together with (1.18) and for any $z \in \mathbb{R}^n$ fulfilling $r = \|z - x^*\|$ such that $0 < r \leq \epsilon$ the following

$$f(z) = f(x^*) + \frac{1}{2}\langle \nabla^2 f(x^*)(z - x^*), z - x^* \rangle + o(\|z - x^*\|^2) \tag{1.24}$$

$$\geq f(x^*) + \frac{1}{2}\lambda_{\min}(\nabla^2 f(x^*))\langle z - x^*, z - x^* \rangle + o(\|z - x^*\|^2) \tag{1.25}$$

$$\geq f(x^*) + \frac{1}{4}\lambda_{\min}(\nabla^2 f(x^*))\langle z - x^*, z - x^* \rangle \geq f(x^*) \tag{1.26}$$

where in (1.26) the lower bound in (1.23) was used and in (1.25) relation (1.21). Therefore it is concluded that there exist an neighborhood of x^* in which it is a strict minimizer. □

The Lemmas 1.2 to 1.3 provide a useful set of checks that may both be used in analysis as well as in implementation of algorithms to check convergence. As previously mentioned in the introduction, the class of convex functions provide several useful properties that may now be shown.

1.2 Convex functions

One class of functions that is frequently studied in optimization is the class of convex functions, their properties provide important guarantees regarding solutions to optimization problems. As shown in section 1.1.1, it is possible to characterize *local* minimizers of continuous and differentiable functions using first- and second-order optimality conditions. This section show that the properties of convex functions allow those local criteria to also characterize global minimizers. This explains why convex functions are nice to work with in optimization. In order to make a formal definition of convex functions, convex sets must first be introduced and defined.

Definition 1.1 (Convex Set)**.** A set $\Omega \subseteq \mathbb{R}^n$ is called convex if, for any $\alpha \in [0, 1]$ satisfies [2]

$$\alpha x + (1 - \alpha)y \in \Omega \quad \forall x, y \in \Omega. \tag{1.27}$$

Geometrically, a convex set may be interpreted as a set in which any two elements can be connected through a straight line, with every point along the line segment lying entirely in the set. With this definition it is possible to formally define convex functions.

Definition 1.2 (Convex Function)**.** A function $f : \Omega \to \mathbb{R}$ is convex if, for all $x, y \in \Omega \subseteq \mathbb{R}^n$, where Ω is a convex set, satisfies the relation [2]

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y) \quad 0 \leq \lambda \leq 1. \tag{1.28}$$

Much of the literature on optimization focuses on convex functions, since the convergence analysis becomes more approachable. From definition 1.2 it is possible to derive two useful criteria for determining if a function is convex or not.

Lemma 1.4 (First order convexity condition)**.** *Let $f : \Omega \to \mathbb{R}$ be differentiable over a convex set $\Omega \subseteq \mathbb{R}^n$. We say that f is convex over Ω if and only if for all $x, y \in \Omega$ [2]*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x). \tag{1.29}$$

This result essentially say that the function may always be underestimated by the tangent plane or line (depending on Ω) of the function.

Corollary 1.4.1. *A convex, differentiable function $f : \Omega \to \mathbb{R}$ satisfies, for all $x, y \in \Omega \subseteq \mathbb{R}^n$*

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq 0 \tag{1.30}$$

Proof. Since f is convex over Ω, equation (1.29) holds for any two points $x, y \in \Omega$. We may therefore write

$$f(x) \geq f(y) + \nabla f(y)^T (x - y) \tag{1.31}$$

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \tag{1.32}$$

adding (1.31) and (1.32 yields the following expression

$$f(x) + f(y) \geq f(y) + f(x) + \nabla f(y)^T (x - y) + \nabla f(x)^T (y - x) \tag{1.33}$$

$$0 \geq \nabla f(y)^T (x - y) + \nabla f(x)^T (y - x). \tag{1.34}$$

Refactoring (1.34) we find

$$0 \leq \nabla f(y)^T (y - x) - \nabla f(x)^T (y - x) \tag{1.35}$$
$$0 \leq \langle \nabla f(y) - \nabla f(x), y - x \rangle. \tag{1.36}$$

□

Another widely used criteria for convexity is the second-order condition, stated below with along with a proof. It does motivate the need for some results from matrix analysis to analyze wether or not a function is convex.

Theorem 1.5 (Second order convexity condition)**.** *Let $f : \Omega \to \mathbb{R}$ be twice continuously differentiable over an open convex set $\Omega \subseteq \mathbb{R}^n$. The function f is convex on Ω if and only if its Hessian matrix is positive semi-definite*

$$\nabla^2 f(x) \succeq 0, \quad \forall x \in \Omega \tag{1.37}$$

Proof. We first prove the Theorem for a function $f : \mathbb{D} \to \mathbb{R}$ where $\mathbb{D} \subseteq \mathbb{R}$ is open and convex, and then use that result to generalize the proof for functions $f : \Omega \to \mathbb{R}$ by examining them when restricted to lines.

Let $f : \mathbb{D} \to \mathbb{R}$ be a convex function on $\mathbb{D}$. According to the first order condition (1.29) this implies that $\forall x, y \in \mathbb{D}$

$$f(x) \geq f(y) + f'(y)(x - y) \tag{1.38}$$
$$f(y) \geq f(x) + f'(x)(y - x). \tag{1.39}$$

Adding the two inequalities (1.38) & (1.39) yields

$$f(x) + f(y) \geq f(x) + f(y) + f'(y)(x - y) + f'(x)(y - x), \tag{1.40}$$

subtracting $f(x)$ and $f(y)$ from both sides of (1.40) gives the inequality

$$f'(x)(x - y) - f'(y)(x - y) \geq 0. \tag{1.41}$$

We may rewrite (1.41) by letting $h = x - y$

$$f'(y + h)(h) - f'(y)(h) \geq 0. \tag{1.42}$$

Dividing equation (1.42) with h^2 we find

$$\frac{f'(y + h) - f'(y)}{h} \geq 0, \tag{1.43}$$

evaluating the limit as $h \to 0^+$ shows that the second derivative is non negative

$$\lim_{h \to 0} \frac{f'(y + h) - f'(y)}{h} = f''(y) \geq 0, \quad \forall y \in \mathbb{D}. \tag{1.44}$$

This proves that any convex function $f : \mathbb{D} \to \mathbb{R}$ has non negative second derivative.

Now assume that (1.44) holds for all $y \in \mathbb{D}$ and h small enough so $y + h \in \mathbb{D}$. By Taylors formula [15] it is known that since f is twice continuously differentiable there exists a $\xi \in (y, y + h)$ such that

$$f(y + h) = f(y) + f'(y)h + \frac{f''(\xi)h^2}{2} \quad \xi \in (y, y + h) \tag{1.45}$$

Let $x = y + h$ be a small perturbation of x such that $y + h \in \mathbb{D}$, which is possible for any x since $\mathbb{D}$ is an open set. Since we assumed that $f''(x) \geq 0$ for all $x \in \mathbb{D}$ we may rewrite (1.45) in terms of x as

$$f(x) = f(y) + f'(y)(x - y) + \underbrace{\frac{f''(\xi)(x - y)^2}{2}}_{\geq 0} \tag{1.46}$$

$$f(x) \geq f(y) + f'(y)(x - y). \tag{1.47}$$

Thus we have shown that for a function if $f''(x) \geq 0$ for all $x \in \mathbb{D}$ then f is convex on $\mathbb{D}$. Therefore, it is shown that $f : \mathbb{R} \to \mathbb{R}$ is convex if and only if $f''(x) \geq 0$ for all $x \in \mathbb{D}$.

To generalize the result to functions $f : \Omega \to \mathbb{R}$, let the domain of f be $\Omega \subseteq \mathbb{R}^n$ and $x, y \in \Omega$. Let $t \in [0, 1]$, then $tx + (1 - t)y \in \Omega$. Let $g : \mathbb{R} \to \mathbb{R}$ be defined as the function f restricted along the line between x and y

$$g(t) = f(tx + (1 - t)y) \quad \forall x, y \in \Omega. \tag{1.48}$$

Assume that f is convex on Ω, then $g(t)$ is also convex by construction. We have shown that for any such function $g''(\cdot) \geq 0$. Evaluating the second derivative of g yields

$$g''(t) = (x - y)^T \nabla^2 f(tx + (1 - t)y)(x - y) \geq 0, \tag{1.49}$$

which proves that $\nabla^2 f(tx + (1 - t)y) \succeq 0$. Since x and y can be any two vectors in Ω, this proves the case for all $x, y \in \Omega$.

Assume now that $\nabla^2 f(x) \succeq 0$ for all $x \in \Omega$. Using the same function $g : \mathbb{D} \to \mathbb{R}$ with $t \in [0, 1]$ it is known from (1.49) that if $\nabla^2 f(x) \succeq 0$ for all $x \in \Omega$ then $g''(t) \geq 0$, $\forall t \in [0, 1]$ given that $x, y \in \Omega$. It was proven for one dimensional functions (1.47) using Taylors formula that if $g''(t) \geq 0$ then $g(t)$ is convex. Thus we have proven both directions for functions $f : \Omega \to \mathbb{R}$. □

Remark. The two Lemmas can be used both globally and locally on a subset of the full domain, and remains valid as long as the set Ω is convex and open.

As previously mentioned, convex functions are nice to work with in optimization. Perhaps the most useful property is the one that allows the local optimality condition to be generalized to hold globally.

Lemma 1.6 (Global optimality of convex functions)**.** *For a differentiable convex function f (1.2), any strict local optimal point (1.3) is also a unique global minimizer.*

Proof. Let x^* be a strict local minimizer of f. According to the first order necessary conditions (1.3) any local minimizer must fulfill $\nabla f(x^*) = 0$. Using the first order convexity condition in Lemma 1.4 yields

$$f(y) \geq f(x^*) + \langle \nabla f(x^*), (y - x^*) \rangle = f(x^*) \quad \forall y \in \Omega \tag{1.50}$$

□

Many times in optimization theory, it is useful to work with a class of functions that provide more information about the curvature compared to a general convex function. This is typically done by imposing an assumption that the Hessian $\nabla^2 f(x_k) \succeq mI$ for some $m > 0$ for all $x_k \in \Omega$. This provides trackable information about the eigenvalues of the Hessian, which can be used when quantifying convergence rates or showing global convergence of a method. A function f that satisfies the condition $\nabla^2 f(x_k) \succeq mI \; \forall x \in \Omega$ is said to be strongly convex on Ω [12].

Definition 1.3 (Strong convexity)**.** A continuously differentiable function $f : \Omega \to \mathbb{R}$ is strongly convex on Ω if there exists a constant $m > 0$ such that for all $x, y \in \Omega$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} m \|y - x\|^2. \tag{1.51}$$

Now we formally prove the previous statement regarding the bound on the smallest eigenvalue for strongly convex functions, here it is proven for any convex set $\Omega \subseteq \mathbb{R}^n$, but typically Ω will consist of the whole $\mathbb{R}^n$.

Lemma 1.7 (Second order condition for strong convexity)**.** *Let $f : \Omega \to \mathbb{R}$ be twice continuously differentiable on the open set Ω. We say that f is strongly convex if and only if for any $x \in \Omega$*

$$\nabla^2 f(x) \succeq mI. \tag{1.52}$$

Proof. Similarly as the second order proof of convexity it will first be done for scalar functions, then generalized to higher dimensions. Let $g : \mathbb{D} \to \mathbb{R}$ be strongly convex on the open, convex set $\mathbb{D} \subseteq \mathbb{R}$. By Definition 1.3 $g(\cdot)$ satisfies for any $x, y \in \mathbb{D}$

$$g(y) \geq g(x) + g'(x)(y - x) + \frac{1}{2}m(y - x)^2 \tag{1.53}$$

$$g(x) \geq g(y) + g'(y)(x - y) + \frac{1}{2}m(x - y)^2. \tag{1.54}$$

Adding the two inequalities and subtracting $g(x) + g(y)$ from both sides yields

$$0 \geq g'(x)(y - x) + \frac{1}{2}m(y - x)^2 + g'(y)(x - y) + \frac{1}{2}m(x - y)^2. \tag{1.55}$$

Refactoring and noting that $(x - y)^2 = (y - x)^2$, we may rewrite (1.55) as

$$g'(x)(x - y) - g'(y)(x - y) \geq m(x - y)^2. \tag{1.56}$$

Assume that $x \neq y$ and let $h = x - y$, which when substituted into (1.56) yields

$$g'(y + h)h - g'(y)h \geq mh^2. \tag{1.57}$$

Dividing (1.57) with h^2 and evaluate as $h \to 0^+$ yields

$$g''(y) \geq m. \tag{1.58}$$

To show sufficiency, assume that 1.58 holds for any $y \in \mathbb{D}$. Let $h > 0$ be a small enough constant such that $y + h \in \mathbb{D}$. By Taylors formula it is known that there exists a $\xi \in (y, y + h)$ such that [15]

$$g(y + h) = g(y) + g'(y)h + \frac{g''(\xi)h^2}{2}. \tag{1.59}$$

Since it was assumed that $g''(y) \geq m$ for all $y \in \mathbb{D}$, (1.59) may be bounded below as

$$g(y) + g'(y)h + \frac{g''(\xi)h^2}{2} \geq g(y) + g'(y)h + \frac{1}{2}mh^2. \tag{1.60}$$

Let $x = y + h$, substituting x into (1.60) the inequality may be rewritten as

$$g(x) \geq g(y) + g'(y)(x - y) + \frac{1}{2}m(x - y)^2. \tag{1.61}$$

Comparing (1.61) with the Definition 1.3 we can conclude that $g''(x) \geq m$ implies strong convexity, hence both directions have been shown.

We generalize the result by considering $f : \Omega \to \mathbb{R}$ evaluated along every possible line segment in the open convex set $\Omega \subseteq \mathbb{R}^n$. Let $t \in [0, 1]$, since Ω is a convex set it is known by Definition 1.1 that $tx + (1 - t)y \in \Omega$. Let $g(t) = f(tx + (1 - t)y)$ be the function defined as f evaluated along a line segment between any two points $x, y \in \Omega$. If f is assumed to be strongly convex i.e. f satisfies (1.51), then f is also strongly convex along any line in the domain Ω, which implies that $g(t)$ is strongly convex. Evaluating $g''(t)$ yields

$$g''(t) = (x - y)^T \nabla^2 f(tx + (1 - t)y)(x - y), \tag{1.62}$$

which by (1.58) let us conclude $(x - y)^T \nabla^2 f(tx + (1 - t)y)(x - y) \geq m$ and thus

$$\nabla^2 f(tx + (1 - t)y) \succeq mI. \tag{1.63}$$

If we instead assume that $\nabla^2 f(x) \succeq mI \ \forall x \in \Omega$, then by (1.62) $g''(t) \geq m \ \forall t \in [0,1]$. From the one dimensional case it is known that this implies that $g(t)$ is strongly convex. Since this holds for any $x, y \in \Omega$, it shows that it holds for any line segment within the domain Ω of f, and it can be concluded that

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} m \|y - x\|^2 \quad \forall x, y \in \Omega. \tag{1.64}$$

Thus, f satisfies the strong convexity condition (1.51) which was to be shown. □

Given the criteria used to analytically characterize convex- functions and sets, a graphical interpretation and how to visualize them might be found useful to build intuition. This also underscores why they are so nice to work with in optimization. We may connect the concepts of convex sets and convex functions geometrically by looking at the *epigraph* of a function f [2].

Definition 1.4 (Epigraph of a function)**.** The epigraph of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the set of pairs (x, t) defined as

$$\text{epi} f = \{(x, t) \,|\, x \in \Omega, \ t \in \mathbb{R}, \ f(x) \leq t\} \tag{1.65}$$

Using a scalar function f, the concept of convex sets and convex functions may now be visualized using the epigraph of f 1.4. As shown in figure 1, a piecewise defined function was constructed with the convex domain Ω where the function is convex clearly marked. The epigraph is only a convex set (1.1) for $x \in \Omega$, shown in green in the figure. For any x outside of Ω, the convexity property of f fails. This illustrates how the full domain of f, a subset of the domain Ω, and epigraph relates to the convexity properties.

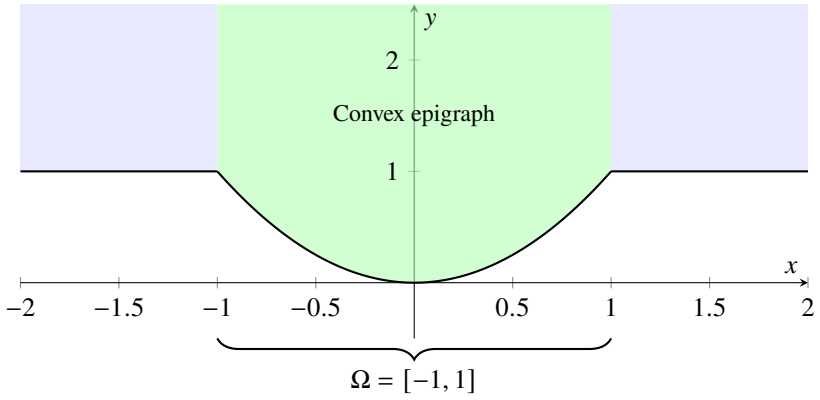


Figure 1: Piecewise function with convex epigraph only on Ω.

1.3 Quadratic Functions

In optimization as well as in other fields of mathematics, it is common practice to begin with a class of functions that are well understood and have properties that simplify the analysis. Throughout the thesis all functions will be assumed to be defined as $f : \mathbb{R}^n \to \mathbb{R}$, the available tools that will be used to analyze such functions depends on several properties, including smoothness, differentiability and convexity. One class of functions that provides an accessible theory for these properties are quadratic functions, which are easily analyzed using results from linear algebra.

Definition 1.5 (Quadratic Function)**.** A quadratic function is a function $f : \mathbb{R}^n \to \mathbb{R}$ of the form

$$f(x) = \frac{1}{2} x^T A x - b^T x + c \tag{1.66}$$

where $A \in \mathbb{R}^{n \times n}$ and is defined to be symmetric, i.e. $A = A^T$, $b \in \mathbb{R}^n$, and $c \in \mathbb{R}$.

The reason why quadratic functions are often initially chosen when doing analysis in optimization is that the analysis and derivations become easier than for general functions. The results from section 1.1.1 that describe the characterizations of a solution, becomes easy analytically. It is known from Lemma 1.4 that any solution to the optimization problem 1.1 must satisfy $\nabla f(x^*) = 0$. When f is a quadratic function this corresponds to solving the linear system $\nabla f(x) = Ax - b = 0$, where the number of solutions depends on the spectral properties of A. If A is positive definite i.e. $A \succ 0$, there is a unique solution x^*, which is also a unique global minimizer of f, see Lemma 1.6. If A is positive semi-definite i.e. $A \succeq 0$, there is an infinite number of solutions that satisfy the first-order condition.

1.4 Smoothness and Lipschitz continuity

In the convergence analysis of algorithms, it is often assumed that the objective function is smooth. It is also common to make assumptions about the smoothness of the derivatives of the objective function. In this section, some theory will be presented that will be used throughout the thesis.

Definition 1.6 (Lipschitz continuous function)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous with constant L if

$$\|f(x) - f(y)\| \leq L\|x - y\| \quad \forall\, x, y \in \mathbb{R}^n. \tag{1.67}$$

It should be noted that the Lipschitz constant L is a constant that is often used in convergence analysis to relate the rate of convergence in terms related to the smoothness of the objective function or its gradient.

Definition 1.7 (Lipschitz continuous derivative)**.** The p-th derivative of a function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant L if

$$\|\nabla^p f(x) - \nabla^p f(y)\| \leq L\|x - y\| \quad \forall\, x, y \in \mathbb{R}^n. \tag{1.68}$$

One important Lemma and Corollary for the thesis is presented and proved below. The Lemma is valid for any function that is twice continuously differentiable, but for quadratic functions it will yield results that is more easily studied using results from matrix analysis. Before proceeding with the Lemma and proof, the following definition is needed.

Definition 1.8 (Induced matrix norm)**.** Given a norm $\|\cdot\|$ on $\mathbb{R}^n$, the induced matrix norm $\|A\|$ for a matrix $A \in \mathbb{R}^{n\times n}$ is defined as

$$\|A\| = \max_{\|x\|=1} \|Ax\|. \tag{1.69}$$

Among its properties, the induced matrix norm fulfills the submultiplicativity $\|Ax\| \leq \|A\| \cdot \|x\|$ for any $x \in \mathbb{R}^n$, see e.g. [1].

Lemma 1.8 (Bound for norm of Hessian matrix for Lipschitz continuous gradient)**.** *A twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ has a Lipschitz continuous gradient with constant L if and only if*

$$\|\nabla^2 f(x)\| \leq L \quad \forall\, x \in \mathbb{R}^n. \tag{1.70}$$

Proof. First lets assume that equation (1.70) holds, then we must prove that $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for any $x, y \in \mathbb{R}^n$. Using the fundamental Theorem of analysis [15] it is known that

$$g(1) - g(0) = \int_0^1 g'(t)dt.$$

Let $g(t) = \nabla f(x + t(y - x))$, applying the chain rule yields

$$\nabla f(x) - \nabla f(y) = \int_0^1 \nabla^2 f(x + t(y - x))(y - x)dt = \Big(\int_0^1 \nabla^2 f(x + t(y - x))dt\Big)(y - x). \tag{1.71}$$

Note that the Hessian matrix defined by the integral is a matrix $\mathrm{M} \in \mathbb{R}^{n\times n}$. If the l_2 norm is used to measure the size each side of equation (1.71) the following holds

$$\|\nabla f(x) - \nabla f(y)\| = \|\Big(\int_0^1 \nabla^2 f(x + t(y - x))dt\Big)(y - x)\| \leq \|\Big(\int_0^1 \nabla^2 f(x + t(y - x))dt\Big)\| \, \|y - x\| \tag{1.72}$$

$$\|\Big(\int_0^1 \nabla^2 f(x + t(y - x))dt\Big)\| \, \|y - x\| \leq \int_0^1 \|\nabla^2 f(x + t(y - x))\|dt \, \|(y - x)\| \tag{1.73}$$

$$\int_0^1 \|\nabla^2 f(x + t(y - x))\|dt \, \|y - x\| \leq L\|x - y\|. \tag{1.74}$$

In (1.72) the submultiplicativity property of the induced matrix norm was used, in (1.73) the triangle inequality of norms $\|x + y\| \leq \|x\| + \|y\|$ and the continuity of the l_2 norm on f was used, see e.g. [1]). In the final step (1.74), since it was assumed that equation (1.70) holds for all x, therefore L must be an upper bound for the integral

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|. \tag{1.75}$$

To prove the opposite direction, the method used in [12] can be used. Assume that f is twice continuously differentiable with Lipschitz continuous gradient, then for any $v \in \mathbb{R}^n$ and $\alpha > 0$

$$\|\Big(\int_0^\alpha \nabla^2 f(x + tv)dt\Big)v\| = \|\nabla f(x + \alpha v) - \nabla f(x)\| \leq \alpha L\|v\| \tag{1.76}$$

$$\lim_{\alpha\to 0} \frac{1}{\alpha}\|\Big(\int_0^\alpha \nabla^2 f(x + tv)dt\Big)v\| = \|\nabla^2 f(x)\| \leq L\|v\|. \tag{1.77}$$

□

For quadratic functions it is possible to define an upper bound of the norm of the Hessian matrix using its largest singular value [1]. Since the Hessian of a quadratic function is the constant matrix A (1.66), it is possible to derive $\|A\|_2$ as follows.

Lemma 1.9 (The spectral norm is induced by the l_2 norm for a matrix $A \in \mathbb{R}^{n\times n}$)**.** *The largest singular value of a matrix $A \in \mathbb{R}^{n\times n}$ is an induced norm by the l_2 norm.*

Proof. Assume that a matrix $A \in \mathbb{R}^{n\times n}$ is defined with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k$, with $k \leq n$. Using the definition of induced norms (1.8) we may write

$$\|A\|_2 = \max_{\|x\|=1} \|Ax\|_2.$$

Using the singular value decomposition $A = U\Sigma V^T$ we may rewrite the equation as [1]

$$\max_{\|x\|=1} \|Ax\|_2 = \max_{\|x\|=1} \|U\Sigma V^T x\|_2 \tag{1.78}$$

$$\max_{\|x\|=1} \|U\Sigma V^T x\|_2 = \max_{\|x\|=1} \|\Sigma V^T x\|_2 \tag{1.79}$$

$$\max_{\|x\|=1} \|\Sigma V^T x\|_2 = \max_{\|y\|=1} \|\Sigma y\|_2 \leq \sigma_1 \tag{1.80}$$

In step (1.79) the fact that norms are unitary invariant was used, in (1.80) we defined $y = V^T x$ and since V is a unitary matrix the constraint was transferred on y with the simplification. Since $\Sigma = \text{diag}(\sigma_i)$, equality holds for the unit vector y that corresponds to the largest singular value, σ_1. □

Using the now established theory, the Lipschitz constant may be determined for quadratic functions.

Corollary 1.9.1 (Lipschitz continuous gradients of quadratic functions)**.** *A quadratic function $f : \mathbb{R}^n \to \mathbb{R}$ has Lipschitz continuous gradient with $L = \sigma_1$.*

Proof. According to definition 1.5, a quadratic function f have the form $f(x) = \frac{1}{2}x^T Ax - b^T x + c$ where $A = A^T$. The gradient and Hessian of f are given by $\nabla f(x) = Ax - b$ and $\nabla^2 f(x) = A$ respectively. It was shown in Lemma 1.8 that the norm of the Hessian matrix must be bounded above by L. And since $\nabla^2 f(x) = A$ the Hessian matrix of a quadratic function is constant, hence

$$\|A\|_2 \leq L. \tag{1.81}$$

Using the definition of induced matrix norms and Lemma 1.9 we conclude that

$$\|\nabla^2 f(x)\|_2 = \|A\|_2 = \sigma_1(A). \tag{1.82}$$

□

An important result from numerical linear algebra is the concept of condition numbers. For linear systems $Ax = b$ it can be seen as a way of quantifying the sensitivity of the system in the sense of how much a small perturbation of x affects the residual to b. This is something of interest in optimization especially when having inexact gradients for example due to the fact that the method runs in a stochastic setting. Since the Hessian matrix $A \in \mathbb{R}^{n\times n}$ of a quadratic function is square, the condition number $\kappa(A)$ may be defined in terms of its singular values [8].

Definition 1.9 (Condition number of a square matrix A)**.** Condition number $\kappa(A)$ of an invertible matrix $A \in \mathbb{R}^{n\times n}$

$$\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \tag{1.83}$$

Remark: For matrices A where not all eigenvalues are nonzero, the condition number may be defined as $\kappa(A) = \infty$. Different norms can be used in the definition, the goal is often not the definite number κ, it is rather a relative quantity to capture how well conditioned two problems are compared to each other [8].

One class of functions that is important in optimization theory is the twice continuously differentiable, strongly convex functions with Lipschitz continuous gradient. As shown in Lemma 1.3 and 1.8 any such function f satisfies both $\|\nabla^2 f(x)\| \leq L$ and $\nabla^2 f(x) \succeq m$ for some $L > 0$ and $m > 0$. Using Lemma 1.9 it is possible to define the bound

$$m \leq \|\nabla^2 f(x)\| \leq L \quad \forall x \in \Omega. \tag{1.84}$$

As will become clear later in the convergence analysis of various methods, this class of functions ensures that the behavior of the different methods can be analyzed using bounds on the worst case condition number. Since, for any $x \in \Omega$ equation (1.84) and (1.83) together gives the upper bound

$$\kappa(\nabla^2 f(x)) \leq \frac{L}{m}, \quad L \geq m > 0. \tag{1.85}$$

1.5 Line search and Wolfe-Conditions

There are different types of optimization methods that are suitable in different settings and for different problems. Direct solutions might be good in some circumstances, however it is not always possible nor feasible to calculate the exact minimizer in one step. The optimization methods that are studied in this thesis are called line search methods, they are popular for many tasks and are widely used in large scale optimization [14].

Line search methods can be represented in the following way, each iteration of the optimization process use the current point x_k and determines the next step as

$$x_{k+1} = x_k + \alpha_k p_k \tag{1.86}$$

where p_k is the direction of the step and α_k is the step size. Note that both p_k and α_k can vary between iterations, and how they are determined depends on the line search method being used. The goal is to iteratively minimize the objective function so that eventually the optimal value x^* is obtained. As mentioned briefly in previous sections, the point that minimizes the objective function is denoted x^*. The goal is therefor to repeat (1.86) from an initial guess x_0 until x^* is found.

In order to succeed with the task of decreasing the value of the objective function it is generally good that the direction p_k points in a direction that decreases the function value. Although, pointing in a direction that decreases the objective function is no guarantee that the global minima is reached, one of the core challenges in the subject. In the literature e.g. [14] the direction p_k is often defined as

$$p_k = -B_k^{-1}\nabla f(x_k) \text{ s.t} \quad B_k = B_k^T \tag{1.87}$$

where the definition of B_k depends on the algorithm being used. Consider for example when $B_k = I$, then the direction is the steepest descent $-\nabla f(x_k)$. Another popular family of of optimization algorithms that will be the focus of this thesis is the quasi Newton methods, where B_k tries to capture curvature information and that sense estimate Newtons method, hence the name.

After choosing p_k the step size α_k is to be considered. There are two general methods of doing it, which is the reason the method (1.86) is often specified as being exact- or inexact line search depending on how α_k is chosen. In exact line search, α_k is determined to minimize f along the direction p_k. In each iteration of (1.86) a function $\phi(\alpha)$ defined as

$$\phi(\alpha) \stackrel{\text{def}}{=} f(x_k + \alpha p_k) \tag{1.88}$$

is minimized with respect to α. Then $\arg\min_{\alpha\in\mathbb{R}} \phi(\alpha)$ is used to exactly minimize f along the search direction p_k.

This is not always useful in practice due to the computations required to determine the exact minimizer α_k. However, it provides useful insight into how the method works, and as will be shown later on it can show how different methods are related. In practice, it is often more convenient to work with inexact line search, see e.g. [14], where an algorithm is used to effectively select a step size that is good enough. In order to quantify *good enough* some criterion is used, and the algorithm typically starts trying a (relatively) large α_k to see if it satisfies the criterion's or not. If the current α does not fulfill the conditions, a new candidate α_k is checked, until hopefully one is found that may be used.

One popular method was proposed by Philip Wolfe that imposes two conditions on α. The first requirement ensures that the objective function is reduced enough and may be expressed

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1\alpha\nabla f(x_k)^T p_k, \tag{1.89}$$

where the constant $0 < c_1 < 1$. Choosing c_1 closer to zero for a more acceptable condition and closer to one if a more strict condition with a bigger decrease of f is required each step. The second condition is used to ensure that the slope of $\phi(\alpha)$ is at least $c_2 \in (c_1, 1)$ times steeper than the initial slope of ϕ. With (1.88) this can be interpreted as a requirement that f must at least decrease along p_k when a step of size α is taken. This curvature condition may be written

$$\nabla f(x_k + \alpha p_k)^T p_k \geq c_2\nabla f(x_k)^T p_k \tag{1.90}$$

Together equation (1.89) and (1.90) define the Wolfe conditions. And in each iteration (1.86) some algorithm of choosing an α_k satisfying the Wolfe conditions (1.89) and (1.90) is used [14].

1.5.1 Convergence of line search methods

In order to prove that a line search algorithm of the form (1.86) converges to a minimum, an intuitive method was proposed by Zoutendijk. Let θ_k be the angle between the search direction p_k and the steepest descent $-\nabla f_k$

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k|p_k\|}. \tag{1.91}$$

In order to prove convergence, a small Lemma is needed to analyze the result of Zoutendijk.

Lemma 1.10 (Boundness of series)**.** *Let θ_k be defined according to (1.91) and let $\|\nabla f(x_k\|$ denote the Euclidean norm of the gradient at x_k. If series defined as*

$$\sum_{k=0}^{\infty} \cos^2(\theta_k)\|\nabla f_k\|^2, \tag{1.92}$$

is convergent and if there furthermore exists an $\delta > 0$ such that $\cos(\theta_k) \geq \delta\, \forall\, k$ then $\lim_{k\to\infty} \|\nabla f_k\| \to 0$.

Proof. Given that the series (1.92) is convergent and as $\cos\theta_k$ must be bounded from zero, θ_k must be bounded away from $\pm 90°$ and thus never be orthogonal to the steepest descent (1.91). It is known from analysis that the terms in the series must decrease toward zero for the series to be convergent [18]. Therefore it must be the case that $\|\nabla f(x_k)\|^2 \to 0$ and thus $\|\nabla f(x_k)\| \to 0$. □

Zoutendijk proved that if a line search algorithm (1.86) where α_k satisfies the Wolfe conditions (1.89) is applied to an objective function f with certain properties, the following will hold.

Lemma 1.11 (Zoutendijks condition for line search methods using Wolfe condition)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function on an open set $\mathbb{S} \subseteq \mathbb{R}^n$ and have Lipschitz continuous gradient on the level set*

$$\mathbb{L} = \{x \in \mathbb{R}^n \,:\, f(x) \leq f(x_0)\} \subseteq \mathbb{S}.$$

Assume furthermore that f is bounded from below, so that there exist a constant $C \in \mathbb{R}$ so that $f(x) > C$ for all $x \in \mathbb{S}$. If a line search method (1.86) is used with initial guess x_0, and α_k satisfies the Wolfe conditions (1.89) and p_k is a descent direction for all iterations k, then

$$\sum_{k=0}^{\infty} \cos^2(\theta_k)\|\nabla f_k\|^2 < \infty \tag{1.93}$$

Proof. From the curvature condition (1.90) it is known that α_k satisfies

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k, \tag{1.94}$$

which together may be rewritten using (1.86) as

$$\nabla f(x_{k+1})^T p_k \geq c_2 \nabla f(x_k)^T p_k. \tag{1.95}$$

Subtracting $\nabla f(x_k)^T p_k$ from both sides of (1.95) yields

$$(\nabla f(x_{k+1})^T - \nabla f(x_k)^T) p_k \geq (c_2 - 1) f(x_k)^T p_k. \tag{1.96}$$

Since ∇f is Lipschitz continuous on $\mathbb{S}$, there exist a constant $L > 0$, such that

$$\|\nabla f(v) - \nabla f(y)\| \leq L\|v - y\|, \quad \forall\, v, y \in \mathbb{S}. \tag{1.97}$$

Since the Wolfe conditions require that $f(x_{+1}) \leq f(x_k)$, it ensures that $x_j \in \mathbb{L} \subseteq \mathbb{S}$ for all $j \geq 0$. Therefore it is true that

$$\|\nabla f(x_{k+1}) - \nabla f(x_k)\| \leq L\|x_{k+1} - x_k\|. \tag{1.98}$$

Using the Lipschitz continouity (1.97) and (1.86) the left hand side of (1.96) will satisfy

$$(\nabla f(x_{k+1})^T - \nabla f(x_k)^T)p_k \leq \alpha_k L\langle p_k, p_k\rangle = \alpha_k L\|p_k\|^2. \tag{1.99}$$

Given (1.99) and (1.96) we may write

$$(c_2 - 1)f(x_k)^T p_k \leq \alpha_k L\|p_k\|^2, \tag{1.100}$$

rearranging terms yields an expression for α_k

$$\alpha_k \geq \frac{(c_2 - 1)f(x_k)^T p_k}{L\|p_k\|^2}. \tag{1.101}$$

It is known from the assumptions that p_k is a descent direction i.e. $\nabla f(x_k)^T p_k < 0$, using the expression of α_k (1.101) and (1.86) in the first Wolfe condition (1.89) the following inequalities are derived

$$f(x_{k+1}) \leq f(x_k) + c_1\alpha_k\nabla f(x_k)^T p_k \leq f(x_k) + c_1\frac{(c_2 - 1)f(x_k)^T p_k}{L\|p_k\|^2}\nabla f(x_k)^T p_k, \tag{1.102}$$

where the fact that p_k is a descent direction was used. Using the definition of θ_k (1.91) we may be expressed equation (1.102) as

$$f(x_{k+1}) \leq f(x_k) - \frac{c_1(1 - c_2)}{L}\cos^2\theta_k\|\nabla f(x_k)\|^2. \tag{1.103}$$

Since this holds for all $0 \leq n \leq k$ it may be written

$$f(x_{k+1}) \leq f(x_0) - \frac{c_1(1 - c_2)}{L}\sum_{n=0}^{k}\cos^2\theta_n\|\nabla f(x_n)\|^2. \tag{1.104}$$

One of the assumptions was that f was bounded from below, which means

$$f(x_0) - f(x_{k+1}) < \infty, \quad \forall k \tag{1.105}$$

Therefore, the series in (1.104) must be convergent, i.e.

$$\sum_{k=0}^{\infty}\cos^2\theta_k\|\nabla f_k\|^2 < \infty. \tag{1.106}$$

□

Lemma 1.10 and 1.11 provides a powerful analytical tool for analyzing line search methods. By the discussion regarding classification of local and global minimizers it is, depending on the properties of f possible to at least prove necessary condition (1.3) using Zoutendijks result.

For strongly convex functions 1.3 it is possible to derive global convergence guarantees using Zoutendijks Lemma 1.11, since the Hessian is positive definite over the domain. Lemma 1.3 and 1.6 provide the proof for this claim. This combined with the fact that the Hessian is constant for quadratic functions will be used to prove global convergence results for the mL-BFGS method in later sections. Hence the tools are now in place to prove the following result

Theorem 1.12 (Global convergence of line search methods)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a strongly convex, continuously differentiable function on an open set $\mathbb{S} \subset \mathbb{R}$. Define the level set $\mathbb{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\} \subseteq \mathbb{S}$. If the method (1.86), where each α_k satisfies the Wolfe conditions (1.89) and (1.90) and p_k is a descent direction is used, then the line search will converge to the unique global minimum x^* if there exists a constant $\delta > 0$ such that $\cos\theta_k \geq \delta$.*

Proof. From Lemma 1.6 and 1.3 and the fact that $\nabla^2 f(x) \succ 0$ for all $x \in \mathbb{L}$, convergence to the unique global minimizer x^* will occur if $\nabla f(x_k) = 0$. Since the iterates (1.86) satisfies the Wolfe conditions and each p_k is a descent direction, it is known from Lemma 1.11 that

$$\sum_{k=0}^{\infty} \cos\theta_k{}^2 \|\nabla f_k\|^2 < \infty. \tag{1.107}$$

Furthermore, since $\cos\theta_k$ is bounded away from zero, it is known from Lemma 1.10 that $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$ which according to Lemma 1.6 together with the fact that $\nabla^2 f(x) \succ 0$ proves the Theorem. □

This concludes the introduction, now with the background needed to move forward with an in depth analysis of the different optimization methods.

2 Optimization algorithms

This section will describe a subset of the many popular methods available. The main focus will be on quasi Newton (QN) methods, both their motivation and relation to other methods. Since one of the QN methods incorporates momentum, two other momentum based methods considered to be fast gradient methods will also be introduced.

2.1 Conjugate gradient method

One method that is popular for solving linear systems of equations in an iterative manner is the Conjugate gradient (CG) method. As discussed in the introduction, finding the solution to a linear system of equations is equivalent to minimizing a strongly convex quadratic function. Strong convexity 1.3 implies that the quadratic function has one unique global minimizer, the solution to the linear system $Ax = b$ were $A \in \mathbb{R}^{n\times n}$. Since $A \succ 0$ the inverse A^{-1} is well defined, and the unique solution may be defined as $x^* = A^{-1}b$.

One interesting relation between the CG method and the family of quasi Newton methods where the update formula of B_k is in the Broyden class, is that for strongly convex quadratic functions they generate the same solutions if exact line search is used. It is a well studied phenomena that the BFGS algorithm fulfills this claim, see [14] or [6].

Given a strongly convex quadratic function $f(x) = \frac{1}{2}x^TAx - b^Tx + c$ where $A \succ 0$ and $A = A^T$ we define the exact step size as the minimizer of the one dimensional function $\phi(\alpha)$ defined as

$$\phi(\alpha) \overset{def}{=} f(x_k + \alpha p_k), \tag{2.1}$$

which for the strongly convex quadratic function may be expressed as

$$\phi(\alpha) = \frac{1}{2}(x_k + \alpha p_k)^T A(x_k + \alpha p_k) - b^T(x_k + \alpha p_k) + c. \tag{2.2}$$

By setting the derivative of ϕ to zero and using the fact that $-\nabla f_k^T = b^T - x_k^T A$ it is possible to express α as

$$\alpha = \frac{b^Tp_k - x_k^TAp_k}{p_k^TAp_k} = -\frac{\nabla f_k p_k}{p_k^TAp_k}. \tag{2.3}$$

One property that the conjugate gradient method does is to generate a set of A-conjugated vectors p,

$$\{p \in \mathbb{R}^n \mid p_i^TAp_j = 0,\ \forall\, i \neq j\ ,\ p \neq 0\}, \tag{2.4}$$

which are linearly independent. Since it is crucial for the motivation of the method this is proven below. If any vector p_k could be written as a combination of the others we would have

$$p_k = \sum_{i\neq k} \alpha_i p_i, \tag{2.5}$$

and then for some $j \neq k$ we would have

$$p_j^TAp_k = p_j^TA\sum_{i\neq k}\alpha_i p_i. \tag{2.6}$$

From the A-conjugacy the right hand side can be rewritten as

$$p_j^TA\sum_{i\neq k}\alpha_i p_i = p_j^TA\left(\alpha_j p_j\right) \neq 0, \tag{2.7}$$

where the last step follows from the positive definiteness of A since $\alpha_j \neq 0$ it is a contradiction, and thus the set of vectors $\{p\}$ must be linearly independent. This is the core idea behind the Conjugate Gradient method - since a set of n linearly independent vectors $p_n \in \mathbb{R}^n$ spans the whole $\mathbb{R}^n$, the solution may be expressed as a linear combination

$$x^* = x_0 + \sum_{i=0}^{n-1}\alpha_i p_i, \tag{2.8}$$

where x_0 is any initial starting point. The algorithm generates these conjugated directions in an iterative manner in accordance to (1.86) by initiating the search direction as $p_0 = -\nabla f_0$ and $r_0 = -p_0$, updating them with

$$p_k = -\nabla f_k + \beta_k p_{k-1}, \tag{2.9}$$

where β_k is chosen to

$$\beta_k = \frac{\nabla f_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}, \tag{2.10}$$

which implies that p_k is A-conjugated to p_{-1} (see [14]). As exact line search is used, each iteration will remove the error in one direction p_k. For a more detailed description of the method there are plenty of sources available, Nocedal and Wrights book [14] has an excellent discussion regarding Krylov subspaces and how it relates to the method.

An interesting property of the method is that for strongly convex quadratic functions the CG method and the BFGS methods will generate the same sequence of iterates $\{x_k\}\cdot$ This is briefly discussed in [14] and in much more detailed described in [6]. Since the L-BFGS method (or at least a modified version) is studied in this paper, a new insight into the method and how it relates to CG under certain assumptions is presented and proved in this paper. It should be noted though that the exact line search is not something that is particularity useful in practice, an implementation using the Wolfe condition or similar technique will outperform exact line search in practice [14]. The exact line search does however give good insight how different line search methods are related, and can probably motivate the new methods that arrived from the start.

2.2 Quasi Newton methods - motivation and derivation

In order to motivate and describe quasi Newton methods, it is necessary to revisit its origin, Newtons method. It is a line search method, where the search direction p_k in (1.86) is defined as

$$p_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k). \tag{2.11}$$

The iterative line search is thus, assuming that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable the iteration becomes

$$x_{k+1} = x_k - \alpha_k \nabla^2 f(x_k)^{-1} \nabla f(x_k). \tag{2.12}$$

However, it is known from matrix analysis that to have a well defined p_k in (2.12) certain requirements on the spectral properties of the Hessian $\nabla^2 f(x_k)$ must be satisfied. If $\nabla^2 f(x_k)$ is full rank, every eigenvalue is non-zero and the inverse is well defined. But as mentioned in the previous section on the convergence of line search methods, the Theorems and Lemmas rely on the fact that p_k must be a descent direction. So even if the inverse of the Hessian is well defined p_k must still be a descent direction, i.e. $\langle p_k, \nabla f(x_k) \rangle < 0$. Thus the requirements on the Hessian are stronger in order to have a well defined method for the convergence Theorems in the previous section. In order to guarantee that p_k is a descent direction, the Hessian must be positive definite i.e. $\nabla^2 f(x_k) \succ 0$. As stated in (1.21) this puts further constraints on the eigenvalues of the Hessian, being non-zero is not enough, they must all be positive. This is intuitive, since if all eigenvalues are positive and non-zero, the Hessian will only stretch or shrink any vector that is transformed, not change direction. Since the steepest descent is a decreasing direction by definition as long as it is not a stationary point, the direction p_k (2.11) will be a descent direction.

This implies that without any modifications of p_k Newtons method has strict restrictions on the objective function to be well defined and successfully converge to a minimum. Perhaps the biggest disadvantage of Newtons method in practice is the fact that the Hessian matrix has to be evaluated in each step, which for a function of many variables becomes a challenging task [14], this challenge is bigger than ever in the era of neural networks in which the objective function may have trillions of variables [17]. However, the reason to seek inspiration from Newton is its ability to converge quadratically.

Quasi Newton methods can be seen as an attempt to approximate Newtons method in a computationally cheaper and more stable way. Common for all such methods is, instead of computing and storing the full Hessian matrix, only first order information from ∇f is used to approximate curvature information, used to generate effective search

directions. The QN methods that are studied in this thesis are based on the BFGS method which can be derived by constructing a quadratic approximation of the objective function f at each iteration

$$m_k(p) = f(x_k) + \langle \nabla f(x_k), p \rangle + \frac{1}{2}\langle B(x_k)p, p \rangle, \tag{2.13}$$

where the matrix $B(x_k)$ is defined to be symmetric and positive definite. The update formula of BFGS instead works directly with the inverse $H(x_k) = B(x_k)^{-1}$. The search direction p_k is defined by $\arg\min_{p \in \mathbb{R}^n} m_k(p)$ in (2.13), which yields

$$p_k = -H(x_k)\nabla f(x_k), \quad H(x_k) \in \mathbb{R}^{n \times n}. \tag{2.14}$$

By comparing (2.14) and (2.11) it is clear that $H(x_k)$ replaces the true inverse Hessian used in the Newton direction. In order to ensure that the quadratic model (2.13) keeps some relevant information of the true objective function f, it is required that the gradient of (2.13) should be equal to the gradient of f for at least the current and previous step. The first condition at the current step follows trivially since $\nabla m_k(0) = \nabla f(x_k)$, the condition for the previous step is commonly referred to as the secant equation

$$\begin{aligned} H(x_{k+1})y_k = s_k \quad \text{where} \quad & s_k = x_{k+1} - x_k \\ & y_k = \nabla f(x_{k+1}) - \nabla f(x_k). \end{aligned} \tag{2.15}$$

Since $H(x_{k+1})$ is positive definite, this requires that $\langle s_k, y_k \rangle > 0$, which for general nonconvex functions is not always true. However, if the step length α_k is chosen to satisfy the Wolfe conditions (1.89) it will hold [14]. From all possible solutions to (2.15), a unique $B(x_{k+1})$ is chosen by solving the following optimization problem

$$\min_H \|H - H(x_k)\| \quad \text{s.t.} \quad H = H^T, \quad Hy_k = s_k. \tag{2.16}$$

The choice of norm in (2.16) affects the solution of the optimization problem and thus yields different update formulas of the inverse Hessian approximation $H(x_k)$. The BFGS update formula is found by minimizing (2.16) using the weighted Frobenius norm, which yields

$$H(x_{k+1}) = (I - \rho_k s_k y_k^T)H(x_k)(I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T \tag{2.17}$$

$$\rho_k = \frac{1}{y_k^T s_k}, \tag{2.18}$$

where y_k and s_k are defined in (2.15). Using the update formula of $H(x_k)$ (2.18) the full algorithm can thus be written

$$x_{k+1} = x_k - \alpha_k H(x_k)\nabla f(x_k), \tag{2.19}$$

where the step length α_k is chosen exact or inexact. It should also be noted that if the optimization (2.16) along with constraints was formulated over $B(x_k)$ instead of the inverse $H(x_k)$, the solution would yield another update formula, known to be less robust in practice [14].

2.3 L-BFGS

One popular method that is often used in large scale optimization is the L-BFGS method. It is more computationally and memory efficient compared to the BFGS, and can be seen as an approximation of the full BFGS. Algorithm (1) describes the method in detail [14].

Algorithm 1 L-BFGS

Input: Initial guess $x_0 \in \mathbb{R}^n$, maximum memory m, tolerance ϵ
Initialize $k \leftarrow 0$; $\{s, y\} \leftarrow []$
Compute $\nabla f(x_0) \in \mathbb{R}^n$
while $\|\nabla f_k\| > \epsilon$ **do**
    Choose $H_k^0 \in \mathbb{R}^{n \times n}$ ▷ Unlike BFGS, H_k^0 can vary $\forall k$.
    $p_k \leftarrow -H_k \nabla f_k \in \mathbb{R}^n$ ▷ $H_k \nabla f_k$ is returned from calling alg. (2) with $\{s, y\}$, H_k^0 as arguments.
    Choose step size $\alpha_k \in \mathbb{R}$ ▷ Using Wolfe conditions or exact line search.
    if $k > m$ **then**
        Pop $\{s_{k-m}, y_{k-m}\}$ from memory ▷ Ensures maximum m pairs $\{s, y\}$ in memory
    end if
    $x_{k+1} \leftarrow x_k + \alpha_k p_k \in \mathbb{R}^n$ ▷ The typical line search update.
    $s_k \leftarrow x_{k+1} - x_k \in \mathbb{R}^n$
    $y_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k) \in \mathbb{R}^n$
    Update $\{s, y\}$ with s_k and y_k.
    $k \leftarrow k + 1$
end while
Return: Approximate solution x^*

The main difference between the L-BFGS algorithm above and the full BFGS is the way that the inverted Hessian approximation H_k is computed. In the full BFGS, the previous H_k's are stored explicitly until convergence is reached. The main benefit of the L-BFGS algorithm is that the m latest vector pairs $\{s, y\}$ can be used to get a good enough approximation of H_k that the full BFGS algorithm calculates. The benefit is primarily noticeable for high dimensional problems, where the memory efficiency of storing only the vectors becomes noticeable in performance compared to storing the full matrix. This yields a trade-off between the number of calculations in the two loop recursion (2LR) and how many iterations until the solution x^* is found [14]. The 2LR algorithm is where the methods differ, this will be clear once the math behind the 2LR is unwrapped.

Algorithm 2 2LR

Input: Vector pairs $\{s, y\}$, Initial $H_k^0 \in \mathbb{R}^{n \times n}$
Compute $q = \nabla f(x_k) \in \mathbb{R}$
for $i = k - 1 \rightarrow k - m$ **do** ▷ First backward loop
    $\alpha_i \leftarrow \frac{1}{y_i^T s_i} s_i^T q \in \mathbb{R}$
    $q \leftarrow q - \alpha_i y_i \in \mathbb{R}^n$
end for
Initialize $r \leftarrow H_k^0 q \in \mathbb{R}^n$ ▷ This will be returned when $k = 0$.
for $i = k - m \rightarrow k - 1$ **do** ▷ First forward loop
    $\beta \leftarrow \frac{1}{y_i^T s_i} y_i^T r \in \mathbb{R}$
    $r \leftarrow r + s_i(\alpha_i - \beta) \in \mathbb{R}^n$
end for
Return: $r = H_k \nabla f_k \in \mathbb{R}^n$

It is worth noting that the 2LR above calculates the product $H_k \nabla f_k$, the reason for this is computational efficiency and is natural when looking at the full algorithm.

In the literature [14] as well as in more recent studies [6] the relation between the conjugate gradient and other quasi Newton methods are analyzed. However, there is one result that to the knowledge of the author have not been shown, regarding the L-BFGS and its relation to the conjugate gradient method. The result is mainly a theoretically interesting

result and can be seen as an extension of known results. It does provide some insight into how previous known results for the full BFGS method may be extended to the lighter, perhaps more frequently used L-BFGS algorithm.

Proposition 2.1. *Let $f(x) : \mathbb{R}^n \to \mathbb{R}$ be a strongly convex quadratic function. Let the number of distinct eigenvalues of the Hessian matrix $A \in \mathbb{R}^{n\times n}$ be denoted r. If $H_k^0 = I$ for all k, and exact line search is used in the L-BFGS algorithm (1), it will using only r vector pairs $\{s, y\}$ in memory converge to the exact solution (assuming exact arithmetic). Furthermore, it will do so with identical steps $\{x_k\}$ as Conjugate Gradient.*

Proof. It is known that the Conjugate Gradient method will under the given assumptions on f converge in at most r iterations, see Theorem 5.4 in [14]. Furthermore, as shown in e.g. [6] and [14] there the Broyden family of quasi Newton methods will generate the same sequence of points $\{x_k\}$ if the initial inverted Hessian approximation H_0 is equal to the identity matrix I. It is therefore sufficient to prove that the L-BFGS, with each $H_k^0 = I$ yield the same iterates $\{x_k\}$ as the BFGS. Given that CG converges in at most r iterations, BFGS must also converge in r iterations. The difference between the BFGS and L-BFGS algorithms lies in how the approximated inverted Hessian H_k is computed. Assuming exact line search, we may express the two iterations as

$$x_{k+1} = x_k - \alpha_k H_k^{\text{BFGS}} \nabla f_k \tag{2.20}$$

$$x_{k+1} = x_k - \alpha_k H_k^{\text{L-BFGS}} \nabla f_k \tag{2.21}$$

thus if it can prove that $H_k^{\text{BFGS}} = H_k^{\text{L-BFGS}}$ for every step until the BFGS converges, the same result can be concluded for L-BFGS. In BFGS the update formula of the inverse Hessian can be expressed [14]

$$H_{k+1} = \left(I - \rho_k y_k s_k^T\right)^T H_k \left(I - \rho_k y_k s_k^T\right)^T + \rho_k s_k s_k^T \tag{2.22}$$

where $\rho_k = \frac{1}{s_k^T y_k}$. To enhance readability the standard notation [14] of V_k will be used

$$V_k = \left(I - \rho_k y_k s_k^T\right). \tag{2.23}$$

If the recursive formula is written out for a couple of steps it is clearer how it relates to the L-BFGS, so if it is done for a few steps, it may be written

$$H_{k+1}^{\text{BFGS}} = V_k^T H_k V_k + \rho_k s_k s_k^T \tag{2.24}$$

$$= V_k^T \left(V_{k-1}^T H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T\right) V_k + \rho_k s_k s_k^T \tag{2.25}$$

$$= V_k^T V_{k-1}^T \left(V_{k-2}^T H_{k-2} V_{k-2} + \rho_{k-2} s_{k-2} s_{k-2}^T\right) V_{k-1} V_k + \rho_{k-1} V_k^T (s_{k-1} s_{k-1}^T) V_k + \rho_k s_k s_k^T \tag{2.26}$$

$$= \dots \tag{2.27}$$

which continues until H_0 is written out explicitly. To make the notation more compact the following notation will be used, let $A_i \in \mathbb{R}^{n\times n}$

$$\overset{\curvearrowleft}{\prod_{i=0}^{k}} A_i = A_k A_{k-1} A_{k-2} \dots A_0 \quad \text{and} \quad \overset{\curvearrowright}{\prod_{i=0}^{k}} A_i = A_0 A_1 A_2 \dots A_k \tag{2.28}$$

Using the notation above the BFGS formula may be expressed as

$$H_{k+1}^{\text{BFGS}} = \left(\overset{\curvearrowleft}{\prod_{i=0}^{k}} V_i^T\right) H_0 \left(\overset{\curvearrowright}{\prod_{i=0}^{k}} V_i\right) + \rho_0 \left(\overset{\curvearrowleft}{\prod_{i=1}^{k}} V_i^T\right) s_0 s_0^T \left(\overset{\curvearrowright}{\prod_{i=1}^{k}} V_i\right) \tag{2.29}$$

$$+ \rho_1 \left(\overset{\curvearrowleft}{\prod_{i=2}^{k}} V_i^T\right) s_1 s_1^T \left(\overset{\curvearrowright}{\prod_{i=2}^{k}} V_i\right) + \dots \tag{2.30}$$

$$+ \rho_k s_k s_k^T \tag{2.31}$$

Note that there is only one H_0 that is used for all iterations, which differs from how the 2LR in L-BFGS where it can vary between iterations. The L-BFGS algorithm will using the 2LR algorithm (2) calculate $H_k^{\text{L-BFGS}} \nabla f_k$ directly, however, an expression for $H_{k+1}^{\text{L-BFGS}}$ using m vectors in memory can be written

$$
\begin{aligned}
H_k^{\text{L-BFGS}} = &\left(\prod_{i=k-m}^{\overleftarrow{k-1}} V_i^T \right) H_k^0 \left(\prod_{i=k-m}^{\overrightarrow{k-1}} V_i \right) + \rho_{k-m} \left(\prod_{i=k-m+1}^{\overleftarrow{k-1}} V_i^T \right) s_{k-m} s_{k-m}^T \left(\prod_{i=k-m+1}^{\overrightarrow{k-1}} V_i \right) \\
&+ \rho_{k-m+1} \left(\prod_{i=k-m+2}^{\overleftarrow{k-1}} V_i^T \right) s_{k-m+1} s_{k-m+1}^T \left(\prod_{i=k-m+2}^{\overrightarrow{k-1}} V_i \right) + \dots \\
&+ \rho_{k-1} s_{k-1} s_{k-1}^T
\end{aligned}
\tag{2.32}
$$

where the recursive expression in (2.22) is being written out explicitly for the $m-1$ latest vector pairs. The main difference between this formula and the one in BFGS is that H_k^0 does not (by default) contain any previous information, and if $k > m$ some previous information will be lost compared to the full BFGS.

As been shown in [14] and [6] BFGS must converge in r iterations if $H_0 = I$, since the Conjugate Gradient does so. If the memory in L-BFGS is set to r, the update formula (2.32) of the iterates will for all iterations up to r yield exactly the same H_k as the BFGS method does, since no vector pairs are dropped from memory. Thus, comparing the written out expressions for H_k^{BFGS} and $H_k^{\text{L-BFGS}}$ it must therefore be that (as long as $H_k^0 = I$)

$$
H_r^{\text{L-BFGS}} = \left(\prod_{i=(r-1)-(r-1)}^{\overleftarrow{r-1}} V_i^T \right) H_k^0 \left(\prod_{i=(r-1)-(r-1)}^{\overrightarrow{r-1}} V_i \right) \tag{2.33}
$$

$$
+ \rho_{(r-1)-(r-1)} \left(\prod_{i=(r-1)-(r-1)+1}^{\overleftarrow{r-1}} V_i^T \right) s_{(r-1)-(r-1)} s_{(r-1)-(r-1)}^T \left(\prod_{i=(r-1)-(r-1)+1}^{\overrightarrow{r-1}} V_i \right) \tag{2.34}
$$

$$
+ \rho_{(r-1)-(r-1)+1} \left(\prod_{i=(r-1)-(r-1)+2}^{\overleftarrow{r-1}} V_i^T \right) s_{(r-1)-(r-1)+1} s_{(r-1)-(r-1)+1}^T \left(\prod_{i=(r-1)-(r-1)+2}^{\overrightarrow{r-1}} V_i \right) + \dots \tag{2.35}
$$

$$
+ \rho_{r-1} s_{r-1} s_{r-1}^T \tag{2.36}
$$

which since $(r-1)-(r-1) = 0$ is identical to the one generated by BFGS with $k+1 = r$ in (2.31). This proves that the r iterates will yield the same sequence of search directions as BFGS, and from the assumption of exact line search and the definition of the sequence $\{x_k\}$ (2.21) they must have the same iterates. Thus, with the results from [14] and [6] it is proven that L-BFGS given the assumptions has the same iterates as Conjugate Gradient.

□

2.4 mL-BFGS

In a recent paper, Niu et al. [13] introduced a modified version of the limited memory BFGS algorithm called mL-BFGS. Similar to the regular L-BFGS algorithm, it uses a limited number of vector pairs $\{s_k, y_k\}$ to approximate the inverse Hessian. The difference between the two methods is that mL-BFGS incorporates momentum to the vector pairs used to update the inverse Hessian approximation (2). Niu et al. demonstrated that the method works well in a stochastic setting, showing robust performance when the gradients used for y_k were noisy. The method thus shows strong potential for large scale problems where a stochastic setup is to be preferred or when gradients are noisy. They

define the momentum as

$$M_{x_{k+1}} = \beta M_{x_k} + (1-\beta)x_{k+1}, \tag{2.37}$$
$$M_{g_{k+1}} = \beta M_{g_k} + (1-\beta)g_{k+1} \tag{2.38}$$

where the momentum term $\beta \in [0, 1]$. Using equation (2.37) and (2.38) the vectors s_k and y_k will be updated in a differently than in the standard L-BFGS, the momentum based vectors will be denoted $\tilde{s}_k$ and $\tilde{y}_k$. Their update formulas are defined using the smoothed momentum terms as

$$\tilde{s}_k = M_{x_{k+1}} - M_{x_k}, \tag{2.39}$$
$$\tilde{y}_k = M_{g_{k+1}} - M_{g_k} \tag{2.40}$$

Together, the full momentum based L-BFGS algorithm may be formulated as shown in (3).

Algorithm 3 mL-BFGS

Input: Initial guess $x_0 \in \mathbb{R}^n$, momentum coefficient $\beta \in [0, 1]$, memory m, tolerance ϵ
Initialize $k \leftarrow 0$; $\{\tilde{s}, \tilde{y}\} \leftarrow []$; $M_{x_0} \leftarrow x_0$; $M_{g_0} \leftarrow \nabla f(x_0)$
while $\|\nabla f_k\| > \epsilon$ **do**
    Choose $H_k^0 \in \mathbb{R}^{n\times n}$ ▷ Initial Hessian approximation
    $p_k \leftarrow -H_k \nabla f_k$ ▷ Use 2LR (Algorithm 2) with at most m pairs $\{\tilde{s}, \tilde{y}\}$
    Choose step size α_k ▷ Using Wolfe conditions or exact line search
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    Update momentum iterates:
    $M_{x_{k+1}} \leftarrow \beta M_{x_k} + (1-\beta)x_{k+1}$ ▷ See equation (2.38) and (2.37)
    $M_{g_{k+1}} \leftarrow \beta M_{g_k} + (1-\beta)\nabla f(x_{k+1})$
    $\tilde{s}_k \leftarrow M_{x_{k+1}} - M_{x_k}$ ▷ This is what differs the method from the standard L-BFGS
    $\tilde{y}_k \leftarrow M_{g_{k+1}} - M_{g_k}$
    if $k \geq m$ **then**
        Pop $\{\tilde{s}_{k-m}, \tilde{y}_{k-m}\}$ from memory
    end if
    Append $\{\tilde{s}_k, \tilde{y}_k\}$ to memory
    $k \leftarrow k + 1$
end while
Return: Approximate solution x^*

Remark: In the article ([13]) the authors suggest having momentum defined as $\beta = 0.9$.

2.4.1 Global Convergence of mL-BFGS

In order to prove global convergence of the mL-BFGS algorithm the results from the introductory section of line search methods will be used. In particular Theorem 1.12 will be used, as it provides a sufficient condition in terms of the angle between the search direction p_k and the steepest descent. The Theorem states that if the angle θ_k (1.91) is bounded away from $\pm 90°$ then the sequence of gradients must converge to zero. As previously proven, this means that the iterates (1.86) will under certain assumptions regarding the objective function approach a global minimum, given that α_k satisfies the Wolfe conditions (1.89). Therefore it will be assumed that all α_k satisfies those conditions. It should also be noted that the proof follows analogously the proof by Nocedal and Liu [11] used when they proved the global convergence of the regular L-BFGS, but with the necessary modifications to incorporate the momentum.

In order to prove the global convergence of the method it must first be proven that the update formula used in algorithm (2) is well defined with the modified vector pairs. It will first be done for m strongly convex quadratic functions with Lipschitz continous gradient with constant L. Recall from the definitions 1.3 and (1.8 that these properties implies that the Hessian is bounded by m and L.

Lemma 2.2 (Curvature condition with momentum for quadratic functions)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a strongly convex 1.3 quadratic function*

$$f(x) = \frac{1}{2}x^T Ax - b^T + c \quad A = A^T, \tag{2.41}$$

with Lipschitz continuous gradient. Let the Hessian be bounded with

$$mI \preceq A \preceq MI, \tag{2.42}$$

for some $m > 0$. If the sequence of vector pairs $\{s_k, y_k\}$ generated through a line search on f satisfies

$$y_k^T s_k \geq m\|s_k\|^2 \quad \forall k, \tag{2.43}$$

then the modified vector pairs that incorporates momentum defined in (2.57) and (2.58) satisfies

$$\tilde{y}_k^T \tilde{s}_k \geq \kappa\|\tilde{s}_k\|^2. \tag{2.44}$$

Proof. The proof follows the one in [13], using induction. For the base case we have

$$\tilde{y}_0^T \tilde{s}_0 = (1-\beta)^2 y_0^T s_0 \geq (1-\beta)^2 m\|s_0\|^2. \tag{2.45}$$

Now assume that the relation (2.44) holds for

$$\tilde{y}_{k-1}^T \tilde{s}_{k-1} \geq \epsilon\|\tilde{s}_{k-1}\|^2. \tag{2.46}$$

for some $\epsilon > 0$. As shown in [13] the secant equation (2.15) is fulfilled for all vector pairs

$$\tilde{y}_n = A\tilde{s}_n \quad \forall n. \tag{2.47}$$

Thus, for the $k-th$ vector pair we have

$$\begin{aligned}
\tilde{y}_k^T \tilde{s}_k &= (\beta\tilde{y}_{k-1} + (1-\beta)y_k)^T (\beta\tilde{s}_{k-1} + (1-\beta)s_k) \\
&= (\beta A\tilde{s}_{k-1} + (1-\beta)As_k)^T (\beta\tilde{s}_{k-1} + (1-\beta)s_k) \\
&= \beta^2 \tilde{s}_{k-1}^T A\tilde{s}_{k-1} + \beta(1-\beta)\tilde{s}_{k-1}^T As_k + \beta(1-\beta)s_k^T A\tilde{s}_{k-1} + (1-\beta)^2 s_k^T As_k \\
&= (\beta\tilde{s}_{k-1} + (1-\beta)s_k)^T A(\beta\tilde{s}_{k-1} + (1-\beta)s_k).
\end{aligned} \tag{2.48}$$

By comparing with the definition of $\tilde{s}_k$ (2.39), it is clear that (2.48) may be rewritten

$$\tilde{s}_{k-1}^T A\tilde{s}_{k-1}, \tag{2.49}$$

which according to [13] and the hypothesis (2.46) is exactly equal to

$$\tilde{s}_{k-1}^T \tilde{y}_{k-1} \geq \epsilon\|\tilde{s}_{k-1}\|^2. \tag{2.50}$$

It is proven that the vector pairs satisfies the curvature condition. □

In order to generalize the result for non quadratic functions the following strategy will be used: If it is known that the series of vectors $\{s_k, y_k\}$ satisfies the secant equation, then the L-BFGS algorithm will be well defined. The next proposal (2.3) examines this. It should be noted, that the reason it becomes much more technical and long is because the Hessian is not assumed to be constant. It is possible to define the product $\tilde{y}_k^T \tilde{s}_k$ using the notation

$$\tilde{s}_k = \sum_{j=0}^{k} (1-\beta)\beta^{k-j} s_j \tag{2.51}$$

and

$$\tilde{y}_k = \sum_{j=0}^{k} (1-\beta)\beta^{k-j} y_j \tag{2.52}$$

which is equivalent to the definition (2.39) and (2.40). However, when defining the inner product of these two vectors the mixed indices is hard to keep track of. The notation gives an easy way to interpret how much the previous points will affect the next. For a detailed discussion about filtering and smoothening of signals see for example [16].

It should be noted that the following proposal examines if there for any strongly convex function with bounded Hessian exists a $\beta \in (0, 1)$ such that the curvature condition holds in the mL-BFGS. Since it is known that the L-BFGS is well defined for any such function, and therefore the secant equation also holds without momentum by default.

Proposition 2.3. *Curvature condition with momentum Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable, strongly convex function 1.3 with bounded Hessian according to*

$$mI \preceq \nabla^2 f(x) \preceq MI. \tag{2.53}$$

If the sequence of vector pairs $\{s_k, y_k\}$ generated through a line search method on f satisfies

$$y_k^T s_k \geq m\|s_k\|^2 \quad \forall k, \tag{2.54}$$

for some $m > 0$. If there exist a β in the interval $(0, 1)$ that satisfies

$$\frac{m}{(M^2+1)c+m} > \beta, \tag{2.55}$$

where c determines the scaling factor between the maximum norm of s_j in memory and the current vector s_k. Then the curvature condition is satisfied with momentum

$$\tilde{y}_k^T \tilde{s}_k > 0 \quad \forall k. \tag{2.56}$$

Proof. Using equation (2.38) and (2.40) (and similarly for $\tilde{s}_k$) the vectors that incorporates momentum may be expressed as

$$\tilde{y}_k = \beta\tilde{y}_{k-1} + (1-\beta)y_k \tag{2.57}$$
$$\tilde{s}_k = \beta\tilde{s}_{k-1} + (1-\beta)s_k. \tag{2.58}$$

We prove (2.56) by induction and show that if β is chosen within a specific interval, the result will hold.

The base case is true since plugging in $k = 0$ and using equation (2.57) and (2.58) yields

$$\tilde{y}_0^T \tilde{s}_0 = (1-\beta)^2 y_0^T s_0 \geq (1-\beta)^2 m\|s_0\|^2. \tag{2.59}$$

Now assume that the relation (2.56) holds for

$$\tilde{y}_{k-1}^T \tilde{s}_{k-1} \geq \epsilon\|\tilde{s}_{k-1}\|^2, \tag{2.60}$$

for some $\epsilon > 0$. Then it must be proven in the inductive step that it holds also for the k-th vector pair.

$$\begin{aligned} \tilde{y}_k^T \tilde{s}_k &= (\beta\tilde{y}_{k-1} + (1-\beta)y_k)^T (\beta\tilde{s}_{k-1} + (1-\beta)s_k) && (2.61)\\ &= \beta^2 \tilde{y}_{k-1}^T \tilde{s}_{k-1} + \beta(1-\beta)(y_k^T \tilde{s}_{k-1} + \tilde{y}_{k-1}^T s_k) + (1-\beta)^2 y_k^T s_k \\ &\geq \beta^2 \epsilon\|\tilde{s}_{k-1}\|^2 + (1-\beta)^2 m\|s_k\|^2 + \beta(1-\beta)(y_k^T \tilde{s}_{k-1} + \tilde{y}_{k-1}^T s_k). && (2.62) \end{aligned}$$

In (2.62) the lower bound from the hypothesis (2.60) and the assumptions (2.54) was used. From the lower bound (2.62) it is clear that the two first terms are positive. The last part with mixed terms needs further analysis, the goal

is to bound it from below. In order to make it easier to analyze the two mixed terms in the last parenthesis, Cauchy Schwartz and Young's inequality may be used [19]. The first term $y^T \tilde{s}_{k-1}$ may be bounded using Cauchy Schwartz

$$|y_k^T \tilde{s}_{k-1}| \leq \|y_k\| \|\tilde{s}_{k-1}\|, \tag{2.63}$$

and similarly

$$|\tilde{y}_{k-1}^T s_k| \leq \|\tilde{y}_{k-1}\| \|s_k\|. \tag{2.64}$$

From Young's inequality it is known that for any nonnegative $a, b \in \mathbb{R}$ we have

$$ab \leq \frac{a^2}{2} + \frac{b^2}{2}, \tag{2.65}$$

which, given some $A > 0$ lets us define $a = \frac{\|y_k\|}{\sqrt{A}}$ and $b = \sqrt{A}\|\tilde{s}_{k-1}\|$ which for (2.63) yields

$$\|y_k\| \|\tilde{s}_{k-1}\| \leq \frac{1}{2A}\|y_k\|^2 + \frac{A}{2}\|\tilde{s}_{k-1}\|^2, \tag{2.66}$$

and similarly if we set $b = \frac{\|s_k\|}{\sqrt{A}}$ and $a = \sqrt{A}\|\tilde{y}_{k-1}\|$ equation (2.64) may be bounded as

$$\|\tilde{y}_{k-1}\| \|s_k\| \leq \frac{A}{2}\|\tilde{y}_{k-1}\|^2 + \frac{1}{2A}\|s_k\|^2. \tag{2.67}$$

Using the upper bound on the Hessian and the definition of y_k and s_k it is known that $\|y_k\| <= M\|s_k\|$. Thus equation (2.66) may be bounded further

$$\frac{1}{2A}\|y_k\|^2 + \frac{A}{2}\|\tilde{s}_{k-1}\|^2 \leq \frac{1}{2A}M^2\|s_k\|^2 + \frac{A}{2}\|\tilde{s}_{k-1}\|^2. \tag{2.68}$$

Since no explicit upper bound of $\|\tilde{y}_{k-1}\|$ or $\|\tilde{s}_{k-1}\|$ new upper limits must be constructed. From (2.51) it is known that

$$\tilde{s}_{k-1} = \sum_{j=0}^{k-1}(1-\beta)\beta^{k-j-1}s_j. \tag{2.69}$$

By evaluating the norm of both sides of (2.69) we may from the triangular inequality obtain

$$\|\tilde{s}_{k-1}\| \leq \sum_{j=0}^{k-1}(1-\beta)\beta^{k-j-1}\|s_j\|. \tag{2.70}$$

The right hand side may further be bounded above, since the summation is a convex operation [2] we obtain

$$\sum_{j=0}^{k-1}(1-\beta)\beta^{k-j-1}\|s_j\| \leq \max_{j \leq k-1}\|s_j\| = L, \tag{2.71}$$

where $L \in \mathbb{R}$ is a bounded constant since we assume that optimization problem is well defined and the step lengths are finite. Substituting (2.71) into (2.68) yields

$$\frac{1}{2A}\|y_k\|^2 + \frac{A}{2}\|\tilde{s}_{k-1}\|^2 \leq \frac{1}{2A}M^2\|s_k\|^2 + \frac{A}{2}L^2. \tag{2.72}$$

Similarly we may bound $\|\tilde{y}_{k-1}\|$ using (2.52). From the definition we know

$$\tilde{y}_{k-1} = \sum_{j=0}^{k-1}(1-\beta)\beta^{k-j-1}y_j, \tag{2.73}$$

and from applying the triangle inequality in (2.73) we obtain

$$\|\tilde{y}_{k-1}\| \leq \sum_{j=0}^{k-1}(1-\beta)\beta^{k-j-1}\|y_j\|. \tag{2.74}$$

Again, using the upper bound on the Hessian we have the bound $y_j \leq M\|s_j\| \forall j$. Substituting this into (2.74) yields

$$\|\tilde{y}_{k-1}\| \leq \sum_{j=0}^{k-1}(1-\beta)\beta^{k-j-1}M\|s_j\|. \tag{2.75}$$

Since the linear combination is a convex operation, it will therefore never exceed the biggest element [2], thus we obtain

$$\|\tilde{y}_{k-1}\| \leq M \max_{j\leq k-1}\|s_j\| = ML. \tag{2.76}$$

Using this in (2.67) yields

$$\|\tilde{y}_{k-1}\|\|s_k\| \leq \frac{A}{2}(ML)^2 + \frac{1}{2A}\|s_k\|^2. \tag{2.77}$$

We therefore obtain

$$\begin{aligned} |y_k^T\tilde{s}_{k-1}| &\leq \|y_k\|\,\|\tilde{s}_{k-1}\| \leq \frac{1}{2A}\|y_k\|^2 + \frac{A}{2}\|\tilde{s}_{k-1}\|^2 \leq \frac{M^2}{2A}\|s_k\|^2 + \frac{A}{2}L^2 \\ |\tilde{y}_{k-1}^T s_k| &\leq \|\tilde{y}_{k-1}\|\,\|s_k\| \leq \frac{A}{2}\|\tilde{y}_{k-1}\|^2 + \frac{1}{2A}\|s_k\|^2 \leq \frac{A\,M^2}{2}L^2 + \frac{1}{2A}\|s_k\|^2, \end{aligned} \tag{2.78}$$

which together yields

$$|y_k^T\tilde{s}_{k-1}| + |\tilde{y}_{k-1}^T s_k| \leq \frac{M^2+1}{2A}\|s_k\|^2 + \frac{A\,(1+M^2)}{2}L^2. \tag{2.79}$$

Equation (2.79) may be transformed into a lower bound

$$y_k^T\tilde{s}_{k-1} + \tilde{y}_{k-1}^T s_k \geq -\frac{M^2+1}{2A}\|s_k\|^2 - \frac{A\,(1+M^2)}{2}L^2, \tag{2.80}$$

which when substituted into (2.62) yields a lower bound that we wish to prove is greater than zero. Since Young's inequality let's us choose the constant A, we may optimize it to minimize the negative contribution from (2.80). Because the other terms in (2.62) are positive we only need to minimize equation (2.80). Let $\phi(A)\,\mathbb{R} \to \mathbb{R}$ be defined as

$$\phi(A) = y_k^T\tilde{s}_{k-1} + \tilde{y}_{k-1}^T s_k \geq -\frac{M^2+1}{2A}\|s_k\|^2 - \frac{A\,(1+M^2)}{2}L^2. \tag{2.81}$$

We obtain the minimizer $\arg\min_{A\in\mathbb{R}}\phi(\cdot)$ by evaluating it's stationary points. Minimizing (2.81) with respect to A yields

$$\frac{d\phi}{dA} = \frac{M^2+1}{2A^2}\|s_k\|^2 - \frac{1+M^2}{2}L^2 \tag{2.82}$$

Solving (2.82) when evaluated at zero obtains the positive root

$$\begin{aligned} \frac{M^2+1}{A^2}\|s_k\|^2 &= (1+M^2)L^2 \\ A^2 &= \frac{\|s_k\|^2}{L^2} \\ A &= \frac{\|s_k\|}{L} = \frac{1}{c}. \end{aligned} \tag{2.83}$$

Substituting the minimizer A (2.83) into (2.80) we obtain

$$y_k^T \tilde{s}_{k-1} + \tilde{y}_{k-1}^T s_k \geq -\frac{(M^2+1)c}{2}\|s_k\|^2 - \frac{(M^2+1)c}{2}\|s_k\|^2 = -(M^2+1)c\|s_k\|^2. \tag{2.84}$$

Now we may substitute the lower bound (2.84) into the expression for $\tilde{y}_k^T \tilde{s}_k$ (2.62), which yields

$$\begin{aligned}
\tilde{y}_k^T \tilde{s}_k &\geq \beta^2\epsilon\|\tilde{s}_{k-1}\|^2 + (1-\beta)^2 m\|s_k\|^2 + \beta(1-\beta)(y_k^T \tilde{s}_{k-1} + \tilde{y}_{k-1}^T s_k) \\
&\geq \beta^2\epsilon\|\tilde{s}_{k-1}\|^2 + (1-\beta)^2 m\|s_k\|^2 - \beta(1-\beta)(M^2+1)c\|s_k\|^2 \\
&= \beta^2\epsilon\|\tilde{s}_{k-1}\|^2 + \left((1-\beta)^2 m - \beta(1-\beta)(M^2+1)c\right)\|s_k\|^2
\end{aligned} \tag{2.85}$$

It is known from the hypothesis that $\|\tilde{s}_{k-1}\| > 0$, so it is enough to give a condition for when the second term in (2.85) is greater than zero

$$(1-\beta)^2 m - \beta(1-\beta)(M^2+1)c > 0. \tag{2.86}$$

Rearranging terms we find that

$$\begin{aligned}
(1-\beta)m - \beta(M^2+1)c &> 0 \\
m - \beta\left(m + (M^2+1)c\right) &> 0 \\
\frac{m}{(M^2+1)c + m} &> \beta.
\end{aligned} \tag{2.87}$$

Thus we conclude, if β is chosen to be less than the right hand side of (2.87) then the update secant equation is true also with momentum. Note that the chosen constant c that describes the relation between the maximum norm of $\|s\|$ over all previous vectors and the current $\|s_k\|$. This implies that the condition (2.87) is to be evaluated per iteration. It should also be noted, if M is much greater than m so that the condition number is large, the interval of valid β will become smaller. □

Remark: In this proof the Hessian is allowed to vary, for the quadratic case the analysis becomes simpler since the Hessian is constant, see Lemma 2.2. If there exists a β in that interval, the curvature condition is satisfied for all functions that fulfills the assumptions. If it does not exist any β in the interval, it is another motivation, as mentioned in the paper [13] that introduced the method, to use damping to ensure the curvature condition holds.

The reason why the curvature condition $y^T s > 0$ is examined is because of two reasons. If $y^T s = 0$ the update formula would simply not be defined, see (2.18). The other reason is that the global convergence proofs that are based on Zoutendijk´s result requires that the search direction p_k is a descent direction. If the inner product $y^T s < 0$, it is not guaranteed that the update formula (2.18) generates a positive definite matrix. If ρ_k in the update formula is negative, the rank one part $\rho_k s_k s_k^T$ might break the requirement that for any nonzero $v \in \mathbb{R}^n$ the condition $v^T H_{k+1} v^T > 0$ must hold, which is easily seen by writing out the full update formula (2.18). If it is not positive definite, the search direction p_k will not be a descent direction (2.14), which is required in the global convergence analysis. Note that the update formula referred to is the classical BFGS update formula, but the requirements remain the same for L-BFGS and mL-BFGS.

Now to prove the global convergence, some minor results and identities will be useful. Firstly, assume that $x, y, v, u \in R^n$ and let $I \in \mathbb{R}^{n\times n}$ denote the identity matrix. Then it is true that [14]

$$\det\left(I + xy^T + uv^T\right) = (1 + y^T x)(1 + v^T u) - (x^T v)(y^T u). \tag{2.88}$$

The upper bound on the Hessian used in Theorem (2.4) ensures that the Theorem is valid for any strongly convex function with Lipschitz continuous gradient, see (1.84).

Theorem 2.4 (Global convergence of mL-BFGS)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function where the Hessian matrix satisfies, for some constants $m > 0$ and $M > 0$*

$$mI \preceq \nabla^2 f(x) \preceq MI, \quad \forall x \in \mathbb{D}, \tag{2.89}$$

where $\mathbb{D} = \{x \in R^n | f(x) \leq f(x_0)\}$ and x_0 the initial point. Assume further that the step size α_k satisfies Wolfe conditions (1.89) and (1.90) every iteration of a line search. If the vector pairs $\{\tilde{s}_k, \tilde{y}_k\}$ satisfies the curvature condition for all k, then the mL-BFGS algorithm will converge to the global minimum x^*.

Proof. From (2.89) it is know that for any $y \in \mathbb{R}^n$

$$m\|y\|^2 \leq y^T \nabla^2 f(x) y \leq M\|y\|^2, \tag{2.90}$$

for any $x \in D$. Since the every α_k satisfies the Wolfe conditions it is known from Zoutendijk's result that if the angle θ_k between the search direction p_k and the steepest descent is bounded away from 90 degrees, the method converges globally. See Theorem 1.12 for a proof. Since the vector pairs are assumed to satisfy the curvature condition $\tilde{y}_k^T \tilde{s}_k \geq \kappa > 0$, the update formula of H_{k+1} is well defined. As mentioned in Nocedal and Liu´s article [11], this ensures that the Sherman Morrison Woodbury can be used to express the update formula in an easier way for this analysis. The modified update formula may be expressed as

$$B_k^{l+1} = B_k^l - \frac{B_k^l \tilde{s}_{jl} \tilde{s}_{jl}^T B_k^l}{\tilde{s}_{jl}^T B_k^l \tilde{s}_{jl}} + \frac{\tilde{y}_{jl} \tilde{y}_{jl}^T}{\tilde{y}_{jl}^T \tilde{s}_{jl}}. \tag{2.91}$$

The number of pairs $\{\tilde{s}_i\,,\, \tilde{y}_i\}_{i=0}^{\tilde{m}-1}$ used is determined according to $\tilde{m} := min\{k+1, m\}$. Since the update will continue over the pairs until $B_{k+1} = B_k^{\tilde{m}}$, that is the expression that will be worked with. To show that the update ensures positive definite B_k it is therefore necessary to bound its eigenvalues using a technique with the determinant and trace. The determinant of the updated $B_k^{\tilde{m}}$ in equation (2.91) may be expressed

$$\det(B_k^{\tilde{m}}) = \det(B_k^{\tilde{m}-1} - \frac{B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)} \tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}} + \frac{\tilde{y}_{j(\tilde{m}-1)} \tilde{y}_{j(\tilde{m}-1)}^T}{\tilde{y}_{j(\tilde{m}-1)}^T \tilde{s}_{j(\tilde{m}-1)}}), \tag{2.92}$$

which may be rewritten by factoring out $B_k^{\tilde{m}-1}$ as

$$\det(B_k^{\tilde{m}}) = \det(B_k^{\tilde{m}-1} \cdot \left(I - \frac{\tilde{s}_{j(\tilde{m}-1)} \tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}} + \frac{(B_k^{\tilde{m}-1})^{-1} \tilde{y}_{j(\tilde{m}-1)} \tilde{y}_{j(\tilde{m}-1)}^T}{\tilde{y}_{j(\tilde{m}-1)}^T \tilde{s}_{j(\tilde{m}-1)}} \right)). \tag{2.93}$$

It is known that $\det(AB) = \det(A)\det(B)$ [1] and it is therefore possible to rewrite equation (2.93) accordingly. The expression inside of the parenthesis is of the form (2.88) with

$$x = \frac{\tilde{s}_{j(\tilde{m}-1)}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}} \quad ; \quad y^T = \tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1}$$
$$u = \frac{(B_k^{\tilde{m}-1})^{-1} \tilde{y}_{j(\tilde{m}-1)}}{\tilde{y}_{j(\tilde{m}-1)}^T \tilde{s}_{j(\tilde{m}-1)}} \quad ; \quad v^T = \tilde{y}_{j(\tilde{m}-1)}^T,$$

and since

$$y^T x = -\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \frac{\tilde{s}_{j(\tilde{m}-1)}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}} = -1, \tag{2.94}$$

we may rewrite the determinant of the inner expression as

$$\det(I + xy^T + uv^T) = -(x^T v)(y^T u) = \frac{\tilde{s}_{j(\tilde{m}-1)}^T \tilde{y}_{j(\tilde{m}-1)}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}}. \tag{2.95}$$

Gathering all of this together we find that

$$\det(B_k^{\tilde{m}}) = \det(B_k^{\tilde{m}-1}) \cdot \frac{\tilde{s}_{j(\tilde{m}-1)}^T \tilde{y}_{j(\tilde{m}-1)}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}}. \tag{2.96}$$

Since the same argument will hold recursively for all $i \in [0, \tilde{m}-1]$ and the assumption that B_k^0 is positive definite the expression may be written [11] as

$$\det(B_k^{\tilde{m}}) = \det(B_k^0) \prod_{n=0}^{\tilde{m}-1} \frac{\tilde{s}_{jn}^T \tilde{y}_{jn}}{\tilde{s}_{jn}^T B_k^n \tilde{s}_{jn}}. \tag{2.97}$$

From the assumptions the vector pairs $\{\tilde{s}_k, \tilde{y}_k\}$ satisfies $\langle \tilde{s}_k, \tilde{y}_k \rangle \geq \kappa > 0$. Therefore, the right hand side may be bounded as

$$\det(B_k^{\tilde{m}}) = \det(B_k^0) \prod_{n=0}^{\tilde{m}-1} \frac{\tilde{s}_{jn}^T \tilde{y}_{jn}}{\tilde{s}_{jn}^T B_k^n \tilde{s}_{jn}} \geq \det(B_k^0) \prod_{n=0}^{\tilde{m}-1} \frac{\kappa}{\tilde{s}_{jn}^T B_k^n \tilde{s}_{jn}} > 0. \tag{2.98}$$

The bound in (2.98) ensures that the product of the eigenvalues must be greater than zero, thus no eigenvalue can be zero. Now it must also be shown that no eigenvalue is unbounded above, which is done using the trace below. We are searching for an expression of

$$Tr(B_k^{\tilde{m}}) = Tr(B_k^{\tilde{m}-1} - \frac{B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)} \tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}} + \frac{\tilde{y}_{j(\tilde{m}-1)} \tilde{y}_{j(\tilde{m}-1)}^T}{\tilde{y}_{j(\tilde{m}-1)}^T \tilde{s}_{j(\tilde{m}-1)}}). \tag{2.99}$$

Using the fact that Trace is a linear operator, we may separate the terms and write [1]

$$Tr(B_k^{\tilde{m}}) = Tr(B_k^{\tilde{m}-1}) - Tr(\frac{B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)} \tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}}) + Tr(\frac{\tilde{y}_{j(\tilde{m}-1)} \tilde{y}_{j(\tilde{m}-1)}^T}{\tilde{y}_{j(\tilde{m}-1)}^T \tilde{s}_{j(\tilde{m}-1)}}). \tag{2.100}$$

Now, note that

$$Tr(\frac{B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)} \tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1}}{\tilde{s}_{j(\tilde{m}-1)}^T B_k^{\tilde{m}-1} \tilde{s}_{j(\tilde{m}-1)}}) > 0, \tag{2.101}$$

which together with (2.100) we may construct the inequality

$$Tr(B_k^{\tilde{m}}) \leq Tr(B_k^{\tilde{m}-1}) + Tr(\frac{\tilde{y}_{j(\tilde{m}-1)} \tilde{y}_{j(\tilde{m}-1)}^T}{\tilde{y}_{j(\tilde{m}-1)}^T \tilde{s}_{j(\tilde{m}-1)}}). \tag{2.102}$$

By expanding the terms recursively using the technique above where the negative part is removed, we may write

$$Tr(B_k^{\tilde{m}}) \leq Tr(B_k^0) + Tr\left(\sum_{n=0}^{\tilde{m}-1} \frac{\tilde{y}_{jn} \tilde{y}_{jn}^T}{\tilde{y}_{jn}^T \tilde{s}_{jn}}\right) = Tr(B_k^0) + \sum_{n=0}^{\tilde{m}-1} \frac{\|\tilde{y}_{jn}\|^2}{\tilde{y}_{jn}^T \tilde{s}_{jn}}. \tag{2.103}$$

Since $\tilde{y}_k^T \tilde{s}_k \geq \kappa$ we may rewrite the right hand side as

$$Tr(B_k^{\tilde{m}}) \leq Tr(B_k^0) + \sum_{n=0}^{\tilde{m}-1} \frac{\|\tilde{y}_{jn}\|^2}{\tilde{y}_{jn}^T \tilde{s}_{jn}} \leq Tr(B_k^0) + \tilde{m} \frac{\|\tilde{y}_{jn}\|^2}{\kappa}. \tag{2.104}$$

Since the Wolfe condition ensures that the iterates are inside of D and that the function is assumed to be bounded from below, the right hand side is finite, therefore it is shown that the eigenvalues are bounded from above as well. Therefore, the Hessian approximation from the update formula have bounded eigenvalues so that $0 < \lambda_i(B) < \infty$

which implies that the search direction $p_k = -B_k^{-1}\nabla f(x_k)$ (2.14) will be a descent direction. Recall the definition of θ_k (1.91) which together with the search direction for the mL-BFGS may be written

$$\cos\theta_k = \frac{-\nabla f(x_k)^T p_k}{\|\nabla f(x_k)\|\|p_k\|} = \frac{-\nabla f(x_k)^T B_k^{-1}\nabla f(x_k)}{\|\nabla f(x_k)\|\|B_k^{-1}\nabla f(x_k)\|}. \tag{2.105}$$

Since the eigenvalues are bounded as discussed earlier so that B_k is positive definite, it can be concluded that

$$\cos\theta_k = \frac{-\nabla f(x_k)^T B_k^{-1}\nabla f(x_k)}{\|\nabla f(x_k)\|\|B_k^{-1}\nabla f(x_k)\|} \geq \delta > 0. \tag{2.106}$$

From Theorem 1.12 it is known that since $\cos\theta_k > 0$ for all k, and the fact that α_k satisfies the Wolfe conditions, we conclude that

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0, \tag{2.107}$$

guaranteeing convergence to global minimum x^*. □

Remark: This Theorem holds for strongly convex, twice continuously differentiable functions with Lipschitz continuous gradient, see (1.84).

2.5 Spectral analysis of the inverse Hessian approximation in BFGS

It is common practice in numerical linear algebra to analyze the conditioning of problems, as that affects the sensitivity of the system, see e.g. [8] or [1]. One of the main objectives in this thesis is to understand and compare different methods when the gradients are inexact, which usually is the case when running the optimization method stochastically in large scale optimization [13]. This section aims to understand how the conditioning of H_{k+1} is affected by the geometry of $\{s_k, y_k\}$ as well as providing some intuition on how one ill conditioned update H_k might affect later updates. Furthermore, a short analysis on how the search direction p_k might be affected when the gradient is inexact depending on the condition number of H_k.

Before proceeding, it should be mentioned that a common practice is to use damping or similar techniques to enforce positive definiteness of the approximated Hessian H_k. This is particularly crucial when the objective function is non convex or when using the method stochastically. When the curvature condition $y^T s > 0$ breaks the convergence guarantees of Theorem 2.4 fails to hold. [3] and [13] among others discusses how damping and other techniques may be used. In [5] the authors provide a full spectral analysis of the Hessian approximation B_k.

Recall the BFGS update formula (2.18)

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T,$$

where $V_k = (I - \rho_k y_k s_k^T)$ and $\rho = \frac{1}{s_k^T y_k}$. As mentioned in the introduction, the condition number $\kappa(H_{k+1})$ may be described using the smallest and largest singular value of H_{k+1}, see Definition 1.9. Remember that for the BFGS family of quasi Newton, H_{k+1} is symmetric and positive definite by construction (assuming $y^T s > 0$), hence $\kappa(H_{k+1})$ may be defined as the quotient of the largest and smallest eigenvalue of H_{k+1}, i.e.

$$\kappa(H_{k+1}) = \frac{\lambda_{\max}(H_{k+1})}{\lambda_{\min}(H_{k+1})}. \tag{2.108}$$

Since H_k is assumed to be symmetric and positive definite, the Loewner partial order be may be used, hence it can be concluded that $\lambda_{\min}(H_k)I \preceq H_k \preceq \lambda_{\max}(H_k)I$ [1]. This allows the spectral analysis of the update formula of H_{k+1} to be done in terms of upper and lower bounds, since

$$V_k^T(\lambda_{\min}(H_k)I)V_k + \rho_k s_k s_k^T \preceq H_{k+1} \preceq V_k^T(\lambda_{\max}(H_k)I)V_k + \rho_k s_k s_k^T. \tag{2.109}$$

Refactoring (2.109) yields

$$\lambda_{\min}(H_k)V_k^TV_k + \rho_k s_k s_k^T \preceq H_{k+1} \preceq \lambda_{\max}(H_k)V_k^TV_k + \rho_k s_k s_k^T. \tag{2.110}$$

In order to determine an expression for the condition number of H_{k+1}, it is helpful to work with bounds of the largest and smallest eigenvalues of H_{k+1}. One Lemma from matrix analysis that provides insight into how the eigenvalues of two Hermitian (symmetric) matrices relate to each other is the following. Since only real valued matrices are being studied in this thesis the result from [1] will be shown over the field $\mathbb{R}^{n\times n}$.

Lemma 2.5 (Eigenvalue inequality of symmetric matrices)**.** *Let $A, B \in \mathbb{R}^{n\times n}$ both be symmetric matrices, i.e. $A = A^T$ and $B = B^T$. Let the eigenvalues be ordered as $\lambda_1(A) \leq \lambda_2(A) \leq \cdots \leq \lambda_n(A)$ and $\lambda_1(B) \leq \lambda_2(B) \leq \cdots \leq \lambda_n(B)$. If $A \preceq B$, then $\lambda_i(A) \leq \lambda_i(B)$ for all $i = 1, \ldots, n$.*

Proof. One way of proving the result is by using Weyls Theorem, as shown in [1]. However, a more direct proof uses the Courant-Fisher Theorem, it states that any Hermitian matrix $\mathbb{M} \in \mathbb{C}^{n\times n}$ with ordered eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ the i-th eigenvalue can be expressed as the minimum value attained over a subspace $\mathbb{S} \subset \mathbb{C}^n$, of the maximum Rayleigh quotient $\mathbb{M}$ for unit vectors in $\mathbb{S}$

$$\lambda_i(\mathbb{M}) = \min_{\substack{\mathbb{S}\subset\mathbb{C}^n \\ \dim\mathbb{S}=i}} \max_{\substack{x\in\mathbb{S} \\ \|x\|=1}} x^*\mathbb{M}x. \tag{2.111}$$

Since $B \succeq A$, it follows that $M = B - A \succeq 0$. Therefore it is known that the maximum Rayleigh quotient inequality

$$\max_{\|x\|=1} x^T(M + A)x = \max_{\|x\|=1} x^TMx + x^TAx \geq \max_{\|x\|=1} x^TAx, \tag{2.112}$$

since $M \succeq 0$. Substituting $M + A = B$ and evaluating the min over all $x \in \mathbb{S} \subset \mathbb{R}^n$ we find by Courant-Fisher (2.111) that

$$\min_{\substack{\mathbb{S}\subset\mathbb{R}^n \\ \dim\mathbb{S}=i}} \max_{\substack{x\in\mathbb{S} \\ \|x\|=1}} x^T(M + A)x \geq \min_{\substack{\mathbb{S}\subset\mathbb{R}^n \\ \dim\mathbb{S}=i}} \max_{\substack{x\in\mathbb{S} \\ \|x\|=1}} x^TAx \tag{2.113}$$

$$\min_{\substack{\mathbb{S}\subset\mathbb{R}^n \\ \dim\mathbb{S}=i}} \max_{\substack{x\in\mathbb{S} \\ \|x\|=1}} x^TBx \geq \min_{\substack{\mathbb{S}\subset\mathbb{R}^n \\ \dim\mathbb{S}=i}} \max_{\substack{x\in\mathbb{S} \\ \|x\|=1}} x^TAx, \tag{2.114}$$

which by (2.111) yields $\lambda_i(B) \geq \lambda_i(A)$. □

One imidiate result from Lemma 2.5 that is used throughout the spectral analysis of H_{k+1} is the following Corollary.

Corollary 2.5.1 (Extremal eigenvalue inequality of symmetric matrices)**.** *Let $A, B \in \mathbb{R}^{n\times n}$ be two symmetric matrices where $A \preceq B$. Then $\lambda_{\min}(A) \leq \lambda_{\min}(B)$ and $\lambda_{\max}(A) \leq \lambda_{\max}(B)$.*

Proof. Let the eigenvalues of A and B be ordered such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Use Lemma 2.5 with $i = 1$ and $i = n$ respectively, which proves the corollary. □

Using the Corollary 2.5.1 allows to construct two inequalities from equation (2.110)

$$\lambda_{\min}(\lambda_{\min}(H_k)V_k^TV_k + \rho_k s_k s_k^T) \leq \lambda_{\min}(H_{k+1}) \tag{2.115}$$

$$\lambda_{\max}(\lambda_{\max}(H_k)V_k^TV_k + \rho_k s_k s_k^T) \geq \lambda_{\max}(H_{k+1}), \tag{2.116}$$

which are of the form $\lambda_{\min}(c_1V_k^TV_k + \rho_k s_k s_k^T) \leq \lambda_{\min}(H_{k+1})$ and $\lambda_{\max}(c_2V_k^TV_k + \rho_k s_k s_k^T) \geq \lambda_{\max}(H_{k+1})$. In order to determine the upper and lower bounds in (2.115) and (2.116), the method used in the proof of Lemma 1 in [3] is used. Since the matrix in both the upper and lower bound is of the form

$$M = c_1V_k^TV_k + \rho_k s_k s_k^T, \tag{2.117}$$

where $c_1 > 0$, the largest and smallest eigenvalue of M can be studied to determine (2.115) and (2.116). The authors of [3] present an analysis of the eigenvalues of M under certain assumptions, which will be reused in this thesis. Before proceeding, recall that the term $V_k^T V_k$ may be expressed in terms of s_k and y_k as

$$V_k^T V_k = (I - \rho_k s_k y_k^T)(I - \rho_k y_k s_k^T) = I - \rho_k \left(y_k s_k^T + s_k y_k^T\right) + \rho_k^2 \|y_k\|^2 s_k s_k^T. \tag{2.118}$$

Following Lemma 1 in [3], assume that the vector pair $\{s_k, y_k\}$ satisfy $s_k^T y_k \geq \mu \|s_k\|^2$ for some $\mu > 0$. Furthermore the vector is assumed to satisfy $\|y_k\| \leq L\|s\|$ for some $L > 0$. Lastly, assume that the constant $c_1 > 0$. Using these assumptions, their derivation was done as follows.

Since $M \in \mathbb{R}^{n \times n}$ is symmetric, it follows from the spectral Theorem that M has n real eigenvalues and has an orthonormal eigenbasis. It is known from linear algebra that any eigenvalue $\lambda \in \mathbb{R}$ and associated eigenvector $z \in \mathbb{R}^n$ of M satisfies the eigenvalue equation $Mz = \lambda z$, which is analyzed for the two cases when the vectors s_k and y_k are colinear respectively non-colinear. Throughout the analysis the relation (2.118) is used when working with M (2.117), since that ensures that M is expressed in terms of the vectors s_k, y_k and known constants.

Assume first that $y_k = \alpha s_k$ for some $\alpha > 0$ i.e. that s_k and y_k are colinear. Any vector $z \in \mathbb{R}^n$ that is perpendicular to s_k, satisfies, by the eigenvalue equation $Mz = \lambda z$

$$Mz = (c_1 \left(I - \rho_k \alpha \left(s_k s_k^T + s_k s_k^T\right) + \rho_k^2 \|s_k\|^2 \alpha^2 s_k s_k^T\right) + \rho_k s_k s_k^T) z = c_1 z, \tag{2.119}$$

since $z^T s_k = 0$. It follows that any such z is an eigenvector associated with the eigenvalue $\lambda = c_1$ and since there are $n - 1$ such orthogonal vectors to s, the eigenvalue $\lambda = c_1$ have multiplicity $n - 1$. Now consider the vector $z = s_k$, still assuming that $y_k = \alpha s_k$

$$\begin{aligned} Mz = (c_1 V_k^T V_k + \rho_k s_k s_k^T) s &= (c_1 \left(I - \rho_k \alpha \left(s_k s_k^T + s_k s_k^T\right) + \rho_k^2 \|s_k\|^2 \alpha^2 s_k s_k^T\right) + \rho_k s_k s_k^T) s_k \\ &= (c_1 \left(1 - 2\rho_k \alpha \|s_k\|^2 + \rho_k^2 \|s_k\|^4 \alpha^2\right) + \rho_k \|s_k\|^2) s_k \\ &= \rho_k \|s_k\|^2 s_k = \lambda s_k, \end{aligned} \tag{2.120}$$

where $\rho_k = \frac{1}{\alpha \|s_k\|^2}$ was used in the simplification. Equation (2.120) tells that $\rho_k \|s_k\|^2$ is an eigenvalue associated with the eigenvector $z = s_k$. Given the assumption $s_k^T y_k \geq \mu \|s_k\|^2$ and $\|y_k\| \leq L\|s_k\|$, using Cauchy Schwarz and the definition of $\rho_k = \frac{1}{s_k^T y_k}$ it may be concluded that λ in (2.120) satisfies

$$\frac{1}{L} \leq \lambda \leq \frac{1}{\mu}. \tag{2.121}$$

Since s_k and y_k are colinear, there are $n - 1$ eigenvectors with corresponding eigenvalue c_1 and one eigenvector s_k with eigenvalue λ bounded according to (2.121).

Continuing, the authors of [3] then considers the case when s_k and y_k are not colinear. Any vector $z \in \mathbb{R}^n$ that is perpendicular s_k and y_k, which are $n - 2$ vectors, satisfies

$$Mz = c_1 z. \tag{2.122}$$

Using the fact that the M is symmetric, the remaining two eigenvectors and associated eigenvalues may be found by considering an $u \in \mathbb{R}^n$, $u = s_k + \gamma y_k$ for some $\gamma \neq 0$. Evaluating $M(s_k + \gamma y_k) = \lambda(s_k + \gamma y_k)$ yields

$$M(s_k + \gamma y_k) = \left((c_1 (I - \rho_k \left(y_k s_k^T + s_k y_k^T\right) + \rho_k^2 \|y_k\|^2 s_k s_k^T) + \rho_k s_k s_k^T\right) (s_k + \gamma y_k) \tag{2.123}$$

$$= \left(c_1 \rho_k^2 \|y_k\|^2 \|s_k\|^2 + \gamma + \rho_k \|s_k\|^2\right) s_k - c_1 \rho_k \|s_k\|^2 y_k, \tag{2.124}$$

which yields the two equations

$$\lambda = c_1 \rho_k^2 \|y_k\|^2 \|s_k\|^2 + \gamma + \rho_k \|s_k\|^2 \tag{2.125}$$

$$\gamma \lambda = -c_1 \rho_k \|s_k\|^2. \tag{2.126}$$

From (2.125) it is known that $\gamma = \lambda - c_1\rho_k^2\|y_k\|^2\|s_k\|^2 - \rho_k\|s_k\|^2$, by substituting this into (2.126) and rearranging terms, a polynomial $p(\lambda)$ of degree two may be constructed as

$$p(\lambda) = \lambda^2 - \lambda\rho_k\|s_k\|^2\big(1 + c_1\rho_k\|y_k\|^2\big) + \rho_k\|s_k\|^2c_1 = 0. \tag{2.127}$$

Evaluating the second derivative of the polynomial $p''(\lambda) = 2$, which by Theorem 1.5 ensures that $p(\cdot)$ is (strongly) convex. Hence, using the fact that any convex function is an over estimator of any tangent line, see Lemma 1.4, the authors found a lower bound using a tangent line at $\lambda = 0$

$$l(\lambda) = p(0) + p'(0)\lambda = c_1\rho_k\|s_k\|^2 - \lambda\rho_k\|s_k\|^2(1 + c_1\rho_k\|y_k\|^2). \tag{2.128}$$

Evaluating when $l(\lambda_0) = 0$ yields

$$\lambda_0 = \frac{c_1}{1 + c_1\rho_k\|y_k\|^2}, \tag{2.129}$$

which, when using the assumptions on s_k and y_k and since $p(\lambda) \geq l(\lambda)\ \forall\lambda$ ensures

$$\lambda \geq \lambda_0 \geq \frac{c_1}{1 + \frac{c_1}{\mu}L^2} > 0. \tag{2.130}$$

The authors of [3] continue by first noting that the largest eigenvalue is a root of the polynomial (2.127) which by the quadratic formula may be determined as

$$\lambda_2 = \frac{\rho_k\|s_k\|^2(1 + c_1\rho_k\|y_k\|^2) + \sqrt{\rho_k^2\|s_k\|^4(1 + c_1\rho_k\|y_k\|^2)^2 - 4c_1\rho_k\|s_k\|^2}}{2}. \tag{2.131}$$

Since all eigenvalues are real it follows from (2.131) that $\rho_k^2\|s_k\|^4(1 + c_1\rho_k\|y_k\|^2)^2 - 4c_1\rho_k\|s_k\|^2 \geq 0$, which by the identity $\sqrt{a^2 - b} \leq a - \frac{b}{2a}$ yields an upper bound

$$\lambda_2 \leq \rho_k\|s_k\|^2(1 + c_1\rho_k\|y_k\|^2) - \frac{c_1}{1 + c_1\rho_k\|y_k\|^2}. \tag{2.132}$$

Using the assumptions that $s_k^Ty_k \geq \mu\|s_k\|^2$ and Cauchy Schwarz it is known that $\frac{1}{\|s\|\|y\|} \leq \rho_k \leq \frac{1}{\mu\|s_k\|^2}$, and since $\|y_k\| \leq L\|s_k\|$

$$\lambda_2 \leq \rho_k\|s_k\|^2(1 + c_1\rho_k\|y_k\|^2) - \frac{c_1}{1 + c_1\rho_k\|y_k\|^2} \leq \frac{1}{\mu} + \frac{c_1}{\mu^2}L^2 - \frac{c_1}{1 + \frac{c_1}{\mu}L^2}. \tag{2.133}$$

Thus, considering both the colinear and non-colinear case, it is concluded from (2.130) and (2.121) that the smallest eigenvalue $\lambda_{\min}(M)$ must be bounded below as

$$\lambda_{\min}(M) \geq \min\Big\{\frac{1}{L}, \frac{c_1}{1 + \frac{c_1}{\mu}L^2}\Big\}. \tag{2.134}$$

Similarly, the largest eigenvalue $\lambda_{\max}(M)$ may be bounded above using equation (2.121) and (2.133), covering both the colinear and non-colinear case as

$$\lambda_{\max}(M) \leq \max\Big\{\frac{1}{\mu}, \frac{1}{\mu} + \frac{c_1}{\mu^2}L^2 - \frac{c_1}{1 + \frac{c_1}{\mu}L^2}\Big\}. \tag{2.135}$$

2.5.1 Eigenvalue inequalities on inverse Hessian approximation

Recall again the BFGS update formula in (2.18), if the objective function f satisfies $s_k^T y_k > \mu \|s_k\|^2$, and that its gradient is Lipschitz continuous i.e.

$$\|\nabla f(x) - \nabla f(z)\| \leq L\|x - z\| \quad \forall x, y \in \mathrm{dom} f.$$

Recall from (2.15) that $s_k = x_k - x_{k-1}$ and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$, thus the relation $\|y_k\| \leq L\|s_k\|$ follow from the Lipschitz continuous gradient. If the previous iteration H_k had bounded eigenvalues, such that $\lambda_{\min}(H_k) \geq \epsilon$ for some $\epsilon > 0$, and $\lambda_{\max}(H_k) \leq \ell$ for some constant $\ell > 0$, the bounds on the eigenvalues (2.134) and (2.135) of M may be applied. The bounds on H_{k+1} (2.115) (2.116) may therefore be defined as

$$\lambda_{\min}(H_{k+1}) \geq \lambda_{\min}(\lambda_{\min}(H_k)V_k^T V_k + \rho_k s_k s_k^T) \geq \min\Big\{\frac{1}{L}, \frac{\lambda_{\min}(H_k)}{1 + \frac{\lambda_{\min}(H_k)}{\mu}L^2}\Big\} \tag{2.136}$$

$$\lambda_{\max}(H_{k+1}) \leq \lambda_{\max}(\lambda_{\max}(H_k)V_k^T V_k + \rho_k s_k s_k^T) \leq \max\Big\{\frac{1}{\mu}, \frac{1}{\mu} + \frac{\lambda_{\max}(H_k)}{\mu^2}L^2 - \frac{\lambda_{\max}(H_k)}{1 + \frac{\lambda_{\max}(H_k)}{\mu}L^2}\Big\}. \tag{2.137}$$

In order to see how the conditioning of H_{k+1} depends on the previous matrix, we may use the bounds (2.137) and (2.136). Consider again the two cases where the vector pairs $\{s_k, y_k\}$ either are colinear or non co-linear. Starting with the colinear case, then it was determined that the smallest eigenvalue $\lambda_{\min}(H_{k+1}) \geq \frac{1}{L}$ where $L > 0$ is the Lipschitz constant of the gradient. The largest eigenvalue in the colinear case was shown to be $\lambda_{\max}(H_{k+1}) \leq \frac{1}{\mu}$. This means that the condition number of H_{k+1} (2.108) may be bounded as

$$\kappa(H_{k+1}) = \frac{\lambda_{\max}(H_{k+1})}{\lambda_{\min}(H_{k+1})} \leq \frac{L}{\mu}. \tag{2.138}$$

In the case where the two vectors $\{s_k, y_k\}$ are not colinear, it is known from (2.136) and (2.137) that $\kappa(H_{k+1})$ may be bounded as

$$\kappa(H_{k+1}) = \frac{\lambda_{\max}(H_{k+1})}{\lambda_{\min}(H_{k+1})} \leq \frac{\frac{1}{\mu} + \frac{\lambda_{\max}(H_k)}{\mu^2}L^2 - \frac{\lambda_{\max}(H_k)}{1+\frac{\lambda_{\max}(H_k)}{\mu}L^2}}{\frac{\lambda_{\min}(H_k)}{1+\frac{\lambda_{\min}(H_k)}{\mu}L^2}}. \tag{2.139}$$

In the colinear case (2.138) the upper bound is only dependent of the Lipschitz constant L and the curvature constant μ. In the other case where the vectors $\{s_k, y_k\}$ are linearly independent, the bound (2.139) became less easy to interpret. However, a simple analysis may be performed by defining

$$A(\lambda_{\min}(H_k), \lambda_{\max}(H_k)) \overset{\text{def}}{=} \frac{\frac{1}{\mu} + \frac{\lambda_{\max}(H_k)}{\mu^2}L^2 - \frac{\lambda_{\max}(H_k)}{1+\frac{\lambda_{\max}(H_k)}{\mu}L^2}}{\frac{\lambda_{\min}(H_k)}{1+\frac{\lambda_{\min}(H_k)}{\mu}L^2}}. \tag{2.140}$$

Varying $\lambda_{\min\,/\,\max}(H_k)$ to see how they affects A provides some insight on how previous conditioning may propagate and affect the conditioning of $\kappa(H_{k+1})$. Consider $A(\alpha\lambda_{\max}, \lambda_{\min})$ where $\alpha > 0$, the effect of different scaling of $\lambda_{\max}$ may be determined by evaluating

$$A(\alpha\lambda_{\max}, \lambda_{\min}) - A(\lambda_{\max}, \lambda_{\min}) = (\alpha - 1)\frac{\frac{\lambda_{\max}(H_k)}{\mu^2}L^2 - \frac{\lambda_{\max}(H_k)}{1+\frac{\lambda_{\max}(H_k)}{\mu}L^2}}{\frac{\lambda_{\min}}{1+\frac{\lambda_{\min}}{\mu}L^2}}, \tag{2.141}$$

which shows that for an increase in the largest eigenvalue ($\alpha > 1$) yields a higher upper bound (2.139). A decrease in largest eigenvalue ($\alpha < 1$) leads to a lower upper bound. The analysis can be done for the smaller eigenvalue $\lambda_{\min}$ as well, showing that a decrease in the lower eigenvalue (assuming constant $\lambda_{\max}$) results in a smaller lower bound of (2.139). This simple analysis show that the guarantees regarding condition number of H_{k+1} is dependent of the conditioning of the previous iteration, which shows how bad updates might propagate.

2.5.2 Condition number analysis

The other part of the analysis is to understand how the parameters μ and L and geometry of the vectors s_k and y_k affect condition of H_{k+1}. From the previous assumptions about the vectors, it is known that $s_k^T y_k \geq \mu \|s_k\|^2$, and the Lipschitz continuity of the gradient yields $\|y_k\| \leq L\|s_k\|$. Using the Cauchy-Schwartz inequality together it is known that

$$\|s_k\|\|y_k\| \geq s_k^T y_k \geq \mu \|s_k\|^2,$$

which together with the Lipschitz continuity of the gradient yields

$$\|s_k\|^2 L \geq s_k^T y_k \geq \mu \|s_k\|^2. \tag{2.142}$$

Recall that the angle θ_k between two vectors $\{s_k, y_k\}$ can be defined, similarly as previously mentioned in (1.91), as

$$\cos\theta_k = \frac{s_k^T y_k}{\|s_k\|\|y_k\|}, \tag{2.143}$$

which together with the previous equations form the inequality

$$\cos\theta_k \geq \frac{\mu\|s_k\|^2}{\|s_k\|\|y_k\|} \geq \frac{\mu\|s_k\|^2}{L\|s_k\|^2} = \frac{\mu}{L}. \tag{2.144}$$

Since the angle θ_k between the vectors s_k, y_k bounds the smoothness μ and Lipschitz constant L, it will according to (2.139) have impact on the conditioning $\kappa(H_{k+1})$. If the vector pair $\{s_k, y_k\}$ become almost orthogonal, such that the angle θ_k between the two vectors get close to $\pm 90°$, the parameters μ must either be very small, or L large, or both. According to the expression of $\kappa(H_{k+1})$ (2.139), the inequality $\cos\theta_k \geq \frac{\mu}{L}$ yield as $\cos\theta_k$ gets small the numerator

$$\frac{1}{\mu} + \frac{\lambda_{\max}(H_k)}{\mu^2}L^2 - \frac{\lambda_{\max}(H_k)}{1 + \frac{\lambda_{\max}(H_k)}{\mu}L^2} = \frac{1}{\mu}(1 + \frac{\lambda_{\max}(H_k)}{\mu}L^2) - \mu\frac{\lambda_{\max}(H_k)}{\mu + \lambda_{\max}(H_k)L^2}, \tag{2.145}$$

the first term $\frac{1}{\mu}(1+\frac{\lambda_{\max}(H_k)}{\mu}L^2)$ becomes large and the second term $\mu\frac{\lambda_{\max}(H_k)}{\mu+\lambda_{\max}(H_k)L^2}$ becomes small. Hence the numerator of (2.139) grows as $\cos\theta_k$ approaches zero, but note that it is bounded from zero. Analyzing the denominator of (2.139)

$$\frac{\lambda_{\min}(H_k)}{1 + \frac{\lambda_{\min}(H_k)}{\mu}L^2} = \mu\frac{\lambda_{\min}(H_k)}{\mu + \lambda_{\min}(H_k)L^2}, \tag{2.146}$$

it is obvious that as $\frac{\mu}{L}$ approaches zero the whole expression becomes smaller. The simple analysis in (2.145) and (2.146) implies that the upper bound of the condition number $\kappa(H_{k+1})$ increases as the angle θ_k between the vectors $\{s_k, y_k\}$ approaches $\pm 90°$.

To summarize, the spectral analysis shows how the upper bound of the condition number $\kappa(H_{k+1})$ depends on the conditioning of the previous iteration $\kappa(H_k)$, as well as showing how the constants μ and L affects the conditioning of the Hessian approximation.

Recall the search direction of the BFGS methods (2.14) $p_k = -H_k\nabla f(x_k)$. Consider the case where the gradient is inexact, which may be represented using some noise $\epsilon \in \mathbb{R}^n$ such that the noisy direction $\hat{p}_k$ can be expressed

$$\hat{p}_k = -H_k(\nabla f(x_k) + \epsilon). \tag{2.147}$$

To analyze how a given ϵ affects the search direction p_k, we may consider the relative error of p_k and $\hat{p}_k$. The relative error can be computed as (see e.g. chapter 5.8 in [1])

$$\frac{\|p_k - \hat{p}_k\|}{\|p_k\|}. \tag{2.148}$$

Using (2.14) and (2.147) it is possible to write the relative error (2.148) as

$$\frac{\|p_k - \hat{p}_k\|}{\|p_k\|} = \frac{\|H_k \epsilon\|}{\|H_k \nabla f(x_k)\|}. \tag{2.149}$$

Assuming that the norm $\|\cdot\|$ is the spectral norm, we may use the Cauchy Schwarz inequality to bound the numerator of (2.149) as

$$\|H_k \epsilon\| \leq \lambda_{\max}(H_k)\|\epsilon\|, \tag{2.150}$$

since H_k is a symmetric real matrix. The denominator of (2.149) can be bounded from below, which can seen by using the spectral Theorem. Let $Q \in \mathbb{R}^{n\times n}$ be an orthogonal matrix and $\Lambda \in \mathbb{R}^{n\times n}$ be a diagonal matrix with the eigenvalues of H_k on the diagonal. Then $H_k = Q\Lambda Q^T$, which allows the denominator to be bounded as

$$\|H_k \nabla f(x_k)\| = \|Q\Lambda Q^T \nabla f(x_k)\| \tag{2.151}$$

$$= \|\Lambda \nabla f(x_k)\| \geq \lambda_{\min}(H_k)\|\nabla f(x_k)\|. \tag{2.152}$$

Using equation (2.150) and (2.152) it is therefore possible to quantify the relative error (2.148) in terms of the condition number of H_k

$$\frac{\|p_k - \hat{p}_k\|}{\|p_k\|} = \frac{\|H_k \epsilon\|}{\|H_k \nabla f(x_k)\|} \leq \kappa(H_k)\frac{\|\epsilon\|}{\|\nabla f(x_k)\|}. \tag{2.153}$$

It can thus be concluded that if the gradient is inexact, the relative error of the search direction might become larger if H_k is ill conditioned.

2.6 Nesterov accelerated gradient descent (NAGD) and Heavy ball

In this subsection two popular first order methods, Heavy Ball and NAGD will be presented. They will be analyzed for strongly convex quadratic functions. These methods will serve as useful baselines when comparing with the QN methods presented previously. Both methods are modified versions of the gradient descent, one of the simplest optimization methods available. However, they offer improved convergence properties compared to the regular gradient descent.

2.6.1 Heavy Ball

A well known variant of gradient descent is the Heavy Ball method. It modifies the classical gradient descent so that the search direction incorporates information from previous directions through momentum. This is done with to improve the stability and convergence speed. The memory from previous directions is referred to as momentum since it essentially tries to capture the effect that momentum of a heavy ball rolling down a slope has. Momentum in the mechanical sense is a property that depends on the velocity and mass, it quantifies a system's resistance to changes in motion. In optimization this idea is introduced to stabilize the path of the algorithm, making it less sensitive to gradient changes and noise by smoothening them out. Furthermore it improves convergence rates compared with the classical gradient descent [12]. The update formula follows the same structure as other line search methods described in this paper with the general form (1.86). The iterates are defined as shown in (2.154) and (2.155), where β is the momentum and α is the step size

$$x_{k+1} = x_k - \alpha p_k \tag{2.154}$$

$$p_{k+1} = \nabla f(x_{k+1}) + \beta p_k \tag{2.155}$$

In classical mechanics and structural engineering stability of systems subject to different loads is often analyzed in order to improve design. One very important aspect is to analyze the natural frequencies of the system, at which the system might resonate more than what is acceptable. The Heavy Ball method may also be analyzed in a similar way in order to select the optimal choices of parameters α and β, to ensure the best convergence properties and performance [17].

when the function is strongly convex and has Lipschitz continuous gradient Given a strongly convex 1.3, quadratic function $f(x) = \frac{1}{2}x^T Ax - b^T x$ with Lipschitz continuous gradient, the update formula (2.154) may be expressed as

$$x_{k+1} = x_k - \alpha \left(\nabla f(x_k) + \beta p_{k-1}\right) = x_k - \alpha \left(Ax_k - b + \beta p_{k-1}\right). \tag{2.156}$$

Using the update formula (2.154) for the previous step x_k we may express p_{k-1} in terms of x and α, since $x_k = x_{k-1} - \alpha p_{k-1}$, which together with (2.156) yield

$$\begin{aligned} x_{k+1} &= -\alpha \left(Ax_k - b + \beta \frac{1}{\alpha}(x_{k-1} - x_k)\right) \\ &= (1+\beta)x_k - \alpha A x_k + \alpha b - \beta x_{k-1}. \end{aligned} \tag{2.157}$$

The strong convexity of f implies that $A \succ mI \succ 0$ i.e. all eigenvalues are greater than zero. The unique minimizer $\arg\min_{x\in\mathbb{R}^n} f(x)$ is therefore $x^* = A^{-1}b$. In order to easier obtain the optimal parameters for the update formula (2.157), it is preferred to define the iterates in terms of the residual in terms of the residual $r_k = x_k - x^*$. Using the fact that we may write

$$\alpha(Ax_k - b) = \alpha A(x_k - x^*) = \alpha A r_k,$$

which lets us rewrite the update formula (2.157) as

$$\begin{aligned} x_{k+1} &= (1+\beta)x_k - \alpha A x_k + \alpha b - \beta x_{k-1} \\ x_{k+1} &= (1+\beta)x_k - \alpha A r_k - \beta x_{k-1} \\ r_{k+1} &= (1+\beta)r_k - \alpha A r_k - \beta r_{k-1} \end{aligned} \tag{2.158}$$

where in (2.158) the definition of r_k was used, canceling all the x^*. The sequence of residuals in (2.158) may then be rewritten in matrix form. Let $y_k = \begin{bmatrix} r_k \\ r_{k-1} \end{bmatrix}$, we may then write

$$y_{k+1} = \begin{pmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{pmatrix} y_k \tag{2.159}$$

Since the goal is to decrease the residuals as quickly as possible, it is known from the recursive definition that

$$y_{k+1} = \begin{pmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{pmatrix} \cdots \begin{pmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{pmatrix} y_0 = \begin{pmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{pmatrix}^{k+1} y_0 \tag{2.160}$$

which will converge only if the largest eigenvalue of the matrix is less than one, see [1] for a discussion on that topic. It can be shown that the optimal choices of β and α are given by

$$\alpha = \frac{4}{\left(\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}\right)^2} \quad \text{and} \quad \beta = \left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right)^2, \tag{2.161}$$

where λ denotes an eigenvalue of the Hessian matrix A, see [17] or [4]. Since the the matrix is assumed to be strongly convex and smooth, both α and β are well defined.

2.6.2 Nesterov accelerated gradient descent

Another popular variant of the gradient descent was introduced by Nesterov, this method further improves the convergence properties achieved by the Heavy Ball method [17]. The method is similar to the Heavy Ball in the sense that it uses momentum to carry information from previous steps. However, it extends the method by evaluating the gradient at a look-ahead point y_k. Note that this is note the y_k used in the analysis of Heavy Ball. The update formula may be represented as

$$x_{k+1} = y_k - \alpha \nabla f(y_k) \tag{2.162}$$

$$y_{k+1} = x_{k+1} + \beta(x_{k+1} - x_k). \tag{2.163}$$

For strongly convex quadratic functions, a similar analysis as the one done on the Heavy Ball can be performed. Using the gradient of a quadratic function we may rewrite (2.162) as

$$x_{k+1} = y_k - \alpha(Ay_k - b). \tag{2.164}$$

Let $\tilde{r}_k$ denote the residual between the look-ahead point y_k and the solution x^* so that $\tilde{r}_k = y_k - x^*$. Then we may simplify (2.164) using the definition $x^* = A^{-1}b$ and $\tilde{r}_k$

$$x_{k+1} = y_k - \alpha(Ay_k - b) = y_k - \alpha A(y_k - x^*) = y_k - \alpha A\tilde{r}_k \tag{2.165}$$

Similarly to the analysis of the Heavy Ball we may denote the residual between the current iterate and x^* by $r_k = x_k - x^*$. Note that it is not the same residual as the one used in (2.165). By subtracting x^* from both sides of (2.165) we may rewrite the relation as

$$r_{k+1} = \tilde{r}_k - \alpha A\tilde{r}_k = (I - \alpha A)\tilde{r}_k \tag{2.166}$$

From the other equation (2.163) we may, using a similar approach as for the Heavy Ball, use the residuals and the identity $x_k - x_{k-1} = r_k - r_{k-1}$ and write

$$y_k = x_k + \beta(x_k - x_{k-1}) \tag{2.167}$$
$$y_k - x^* = x_k - x^* + \beta(x_k - x_{k-1}) \tag{2.168}$$
$$\tilde{r}_k = r_k + \beta(r_k - r_{k-1}). \tag{2.169}$$

Together with (2.166) a full update formula of the residual when using Nesterov's method on quadratic functions may thus be written

$$r_{k+1} = \tilde{r}_k - \alpha A\tilde{r}_k = (I - \alpha A)(r_k + \beta(r_k - r_{k-1}) = (I - \alpha A)(1 + \beta)r_k - (I - \alpha A)\beta r_{k-1}. \tag{2.170}$$

This may also be described in matrix form, let $z_k = \begin{bmatrix} r_k \\ r_{k-1} \end{bmatrix}$, then (2.170) may be written as a linear system

$$z_{k+1} = \begin{pmatrix} (I - \alpha A)(1 + \beta) & -(I - \alpha A)\beta \\ I & 0 \end{pmatrix} z_k. \tag{2.171}$$

In a similar way as described before, the optimal values for α and β is chosen to minimize the largest eigenvalue of the matrix in (2.171). The optimal choice implies the fastest guaranteed reduction of residual, hence best convergence properties. The optimal values are found to be [12]

$$\alpha = \frac{1}{\lambda_{\max}} \quad \text{and} \quad \beta = \frac{1 - \sqrt{\frac{\lambda_{\min}}{\lambda_{\max}}}}{1 + \sqrt{\frac{\lambda_{\min}}{\lambda_{\max}}}}. \tag{2.172}$$

Similarly as for the Heavy Ball, Lipschitz smooth gradient implies that $\lambda_{\max}$ is known, see Lemma 1.8.

3 Numerical experiments

In order to complement the theoretical results of section 2 and 1 a number of numerical experiments was performed, the aim of the experiments was to see if the analysis remains valid in practice. Firstly, in 3.1 the proposed relation between the Conjugate Gradient and L-BFGS under exact line search is tested for a number of matrices with the properties that was assumed in proposition 2.1. Section 3.2 examines the mL-BFGS and the effect of using momentum β on quadratic functions, as well as the Rosenbrock function. A comparison of quasi Newton and first order methods on a constructed function is presented in section 3.3, demonstrating one simple case where quasi Newton methods are more stable. Some numerical experiments are performed in 3.4 to see how the choice of the momentum parameter β in mL-BFGS (Algorithm 3) affects the convergence of a quadratic function with noisy gradients.

3.1 Relation between Conjugate Gradient and L-BFGS

In subsection (2.3) a proposition was made that relates the Conjugate Gradient and L-BFGS when exact line search is used, for quadratic functions with specific properties. In order to test proposition (2.1) numerically, some simulations were performed to see if the conclusion is correct for a few cases. It is not possible to test all possible combinations of symmetric positive definite (SPD) matrices with repeated eigenvalues numerically, yet it provides some level of empirical backup. For full details about the setup and the proposed relation between the methods and their iterates see proposition (2.1), note that L-BFGS was implemented to initiate each iteration with $H_k^0 = I$.

The numerical experiment consisted of generating a number of SPD matrices with defined eigenvalues, see algorithm (4) for details. In order to test different samples of eigenvalues with varying multiplicity, the first n digits of π was used as eigenvalues. Note that the first digit equal to zero is after approximately 30 digits, all zeroes were replaced with a random integer between 1 and 9. This was done to ensure positive definiteness of the Hessian matrix A. In table (1) the experiment and results are shown, the left column shows the number of eigenvalues used in the simulation. The eigenvalues used in each experiment can be seen in figure 2, with multiplicity visualized on the y-axle. As seen in the right column, all iterates were identical for the L-BFGS and CG, supporting the conclusion in proposition (2.1).

Table 1: Empirical tests on the CG - L-BFGS algorithm equivalence (2.1)

Dimension (n)	CG iterations	L-BFGS iterations	Identical steps (Y/N)
5	4	4	Y
10	7	7	Y
15	9	9	Y
30	9	9	Y
50	9	9	Y
100	9	9	Y

Algorithm 4 Symmetric Positive Definite matrix generator

1: **Input:** $\lambda \in \mathbb{R}^n$ containing nonzero eigenvalues.
2: **Returns:** SPD matrix $A \in \mathbb{R}^{n\times n}$ with eigenvalues defined by λ.
3: $M \leftarrow$ random $M \in \mathbb{R}^{n\times n}$
4: Perform QR decomposition of M and save $Q \in \mathbb{R}^{n\times n}$ ▷ Q is orthogonal
5: Let $\Lambda = diag(\lambda)$ ▷ $\Lambda \in \mathbb{R}^{n\times n}$
6: $A \leftarrow Q\Lambda Q^T$ ▷ One might want to return $\frac{1}{2}(A + A^T)$ to enforce symmetry.
7: **Return** A

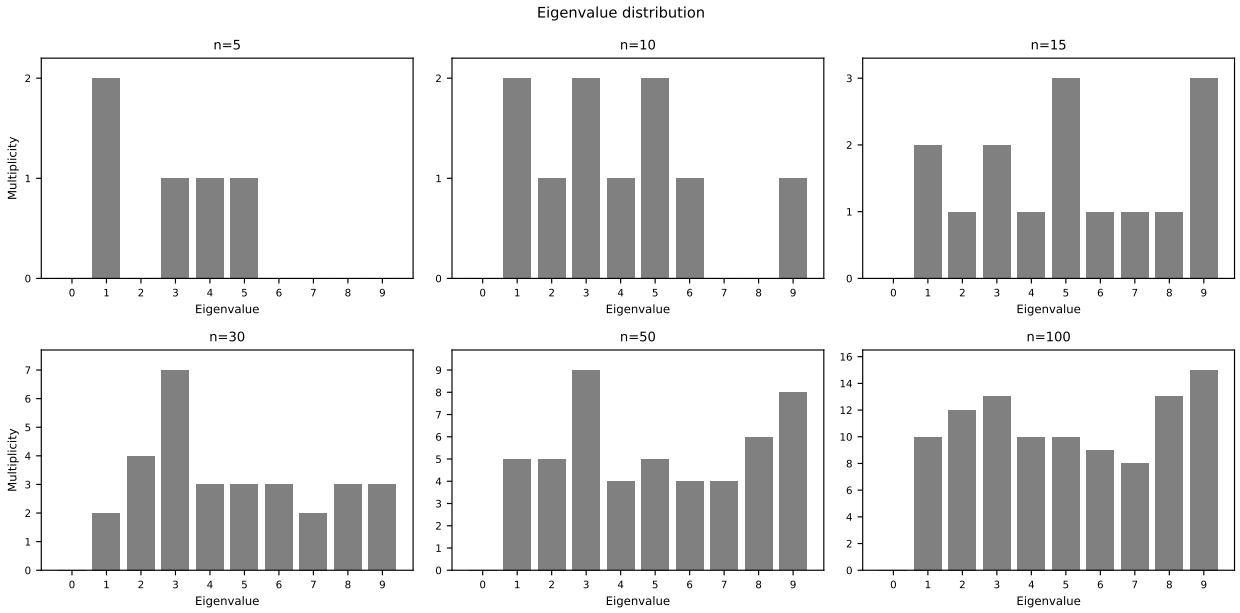


Figure 2: Distribution of eigenvalues for the different n in table (1).

3.2 Momentums effect on convergence for mL-BFGS

In order to examine the effect of momentum on convergence, a number of simulations was performed to compare the performance of mL-BFGS with the standard L-BFGS. A quadratic function $f : \mathbb{R}^{10} \to \mathbb{R}$ was generated using algorithm (4). To simulate the performance in a stochastic setting where the the gradients are inexact, artificial noise was added to the gradient as

$$\tilde{\nabla} f(x) = \nabla f(x) + e(x), \tag{3.1}$$

where $e(x)$ was generated using the *numpy.random* library in Python [9]. The noise was scaled relative to the norm of $\nabla f(x)$ using a scaling factor $\psi = 0.5$ according to

$$e(x) = \psi \|\nabla f(x)\| \gamma, \qquad \gamma \sim \mathcal{N}(0, I). \tag{3.2}$$

Both algorithms were configured with memory $m = 4$, and inexact line search was used. An implementation of the Wolfe conditions (1.89) was used to select the step size α_k.

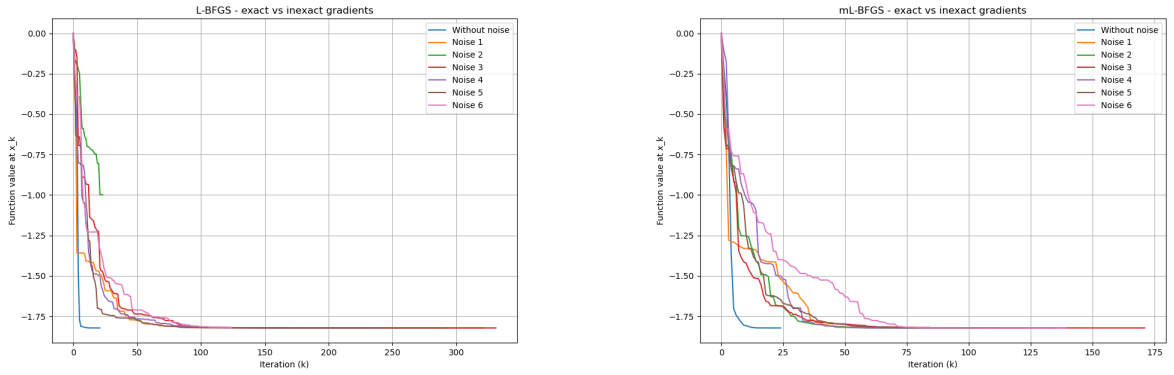
The experiments indicates that it is beneficial to smoothen out the vectors before updating the inverse Hessian approximation. One of the reasons for this is that without momentum the update may become numerically instable, see for example figure 3a where the "Noise 2" failed due to numerical issues caused by the noisy gradients. Comparing with the corresponding figure 3b in which momentum was used, all tests converged to the solution. A similar experiment was made using the Rosenbrock function, defined as

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \tag{3.3}$$

with gradient

$$\nabla f(x_1, x_2) = \begin{bmatrix} -400x_1(x_2 - x_1^2) + 2x_1 - 2 \\ 200(x_2 - x_1^2) \end{bmatrix}. \tag{3.4}$$

Both the mL-BFGS and L-BFGS was configured with memory $m = 4$, and the initial H_k^0 was set to the identity matrix I, which might not be optimal [14]. Both methods used inexact line search to find an α_k that satisfies the Wolfe condition, if none such α_k was found a default step α_0 was selected. In order to examine how the two algorithms would



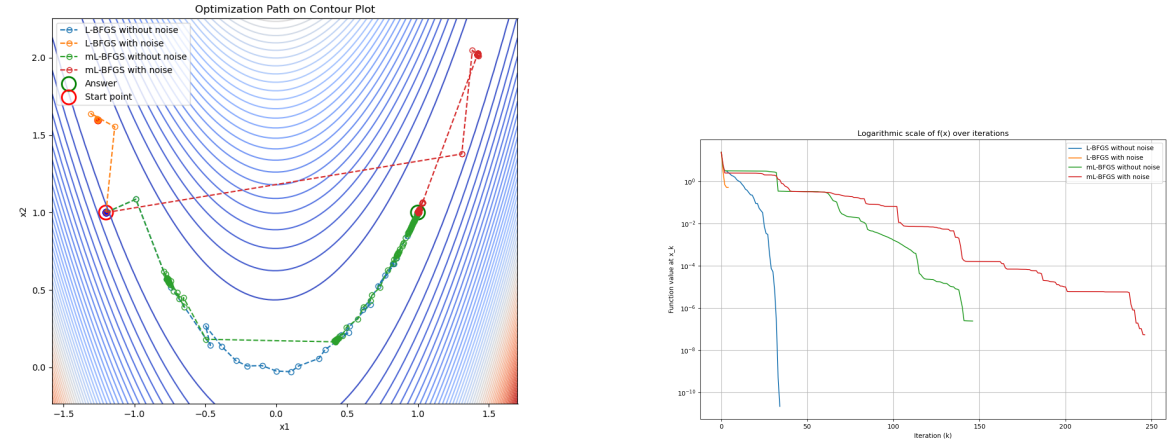
(a) L-BFGS convergence

(b) mL-BFGS convergence

Figure 3: Comparison of momentum's influence on convergence when applied to quadratic functions.

behave with the same noise factor, the seed of the random noise vector was equal for both algorithms during each run, see (3.2).

Since Rosenbrock is a function of two variables, it is possible to visualize the contour of the function along with the different paths taken by the algorithms. Four different paths are shown in figure 4a, where a the L-BFGS and mL-BFGS are tested with and without noisy gradients. The L-BFGS fails to converge to the solution when the gradient is noisy, but converges in fewer iterations than the mL-BFGS when the gradients are exact. Figure 4b shows a comparison of how the function value is reduced for the different setups (note that they are not the same exact sequence as in figure 4a due to stochastic noise). A benchmark test was performed to compare the stability of the two methods



(a) Contour plot of Rosenbrock function with paths taken by L-BFGS and mL-BFGS with and without noisy gradients.

(b) Function evaluated at each step for the different setups. Note that the scale is logarithmic.

Figure 4: Comparison of momentum's influence on convergence when applied to the Rosenbrock function in noisy and exact settings.

when tested with varying noise levels and memory. Experiments showed that the number of iterations used in the line search to find a α_k had impact on the convergence. In this experiment the number of allowed iterations was set to 100,

before using a default α_0. The momentum parameter β was set to 0.9 for the mL-BFGS.

Since the numerical experiments are inexact in an arithmetic sense, the algorithm was considered to have converged successfully if

$$|f(x_k) - f(x^*)| \leq 10^{-4}, \quad \text{where } x^* = [1, 1]. \tag{3.5}$$

As seen in table 2, L-BFGS did not converge in any simulation with noisy gradients. On the other hand, mL-BFGS was more stable stable with both memory $m = 4$ and $m = 8$. For the highest noise level $\psi = 0.5$, mL-BFGS shows a higher success rate when configured with memory $m = 8$. Another interesting observation is that when the gradients are exact, L-BFGS outperforms the mL-BFGS. In the case where $m = 8$ both algorithms converges, but the number of iterations required are significantly higher for mL-BFGS compared with L-BFGS.

In summary, mL-BFGS shows higher success rate when the gradients are inexact, while L-BFGS is was robust and efficient in noise free setting.

Table 2: Comparison of stability and efficiency between L-BFGS and mL-BFGS

Algorithm	Memory (m)	Noise level (ψ)	No. Tests	Successful runs (%)	Average iterations (k) (Successful runs)
L-BFGS	4	0.5	1000	0%	-
mL-BFGS	4	0.5	1000	95.2%	187.5
L-BFGS	4	0.1	1000	0%	-
mL-BFGS	4	0.1	1000	100%	174.2
L-BFGS	4	0	1	100%	36
mL-BFGS	4	0	1	0%	-
L-BFGS	8	0.5	100	0%	-
mL-BFGS	8	0.5	100	99%	219.2
L-BFGS	8	0.1	100	0%	-
mL-BFGS	8	0.1	100	100%	178.2
L-BFGS	8	0	100	100%	35
mL-BFGS	8	0	100	100%	439

3.3 Quasi Newton vs. NAGD and Heavy Ball on constructed function

In [10], the authors presents a strongly convex function that the Heavy Ball algorithm fails to converge on, when using the corresponding optimal parameter values for a quadratic function. The construct function is a piecewise defined, which has been smoothed out to ensure it is twice continuously differentiable. The function is defined as

$$f(x) = \begin{cases} \frac{25}{2}x^2 - 36.0 & x \leq 0.9 \\ -20x^3 + 66.5x^2 - 48.6x - 21.42 & 0.9 < x \leq 1.1 \\ \frac{1}{2}x^2 + 24x - 48.04 & 1.1 < x \leq 1.9 \\ 20x^3 - 113.5x^2 + 240.6x - 185.22 & 1.9 < x \leq 2.1 \\ \frac{25}{2}x^2 - 24x & x > 2.1 \end{cases} \tag{3.6}$$

with the corresponding gradient

$$\nabla f(x) = \begin{cases} 25x & x \leq 0.9 \\ -48.6 + 133x - 60x^2 & 0.9 < x \leq 1.1 \\ x + 24 & 1.1 < x \leq 1.9 \\ 240.6 - 227x + 60x^2 & 1.9 < x \leq 2.1 \\ 25x - 24 & x > 2.1 \end{cases} \tag{3.7}$$

The one dimensional function is constructed to be strongly convex 1.3, which is seen by evaluating the second derivative. Since $f''(x)$ is bounded as $1 \leq f''(x) \leq 25$, it also implies that the function has a Lipschitz continuous gradient with constant $L = 25$ (Lemma 1.8). As shown in the analysis of the method in section 2.6.1, the smoothness and strong convexity implies that there exist optimal choices of step size α and β. In the paper [10], the authors show that the Heavy Ball method does not converge for all initial points $x_0 \in \mathbb{R}$ when using the optimal values of α and β 2.161.

In this thesis quasi Newton methods are compared with the Heavy Ball and NAGD for the constructed function f. The initial x_0 was set to 3.2 for all the tests in the comparison, shown in (5). Both the NAGD and Heavy Ball was configured to use their respective optimal values.

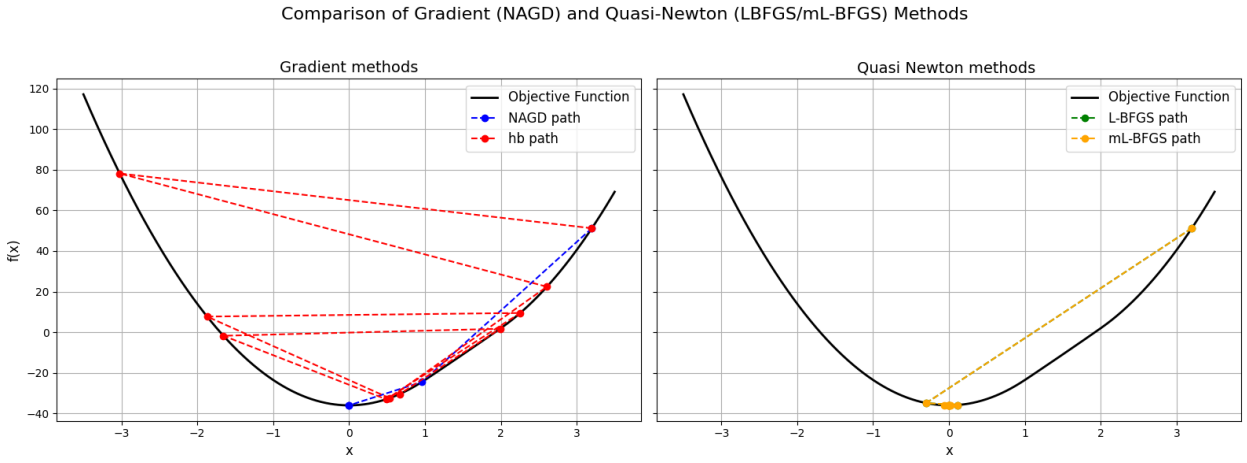


Figure 5: Comparison of algorithm's convergence when applied to the constructed function.

As can be seen in the figure 5 the Heavy Ball does indeed behave unstable, as previously observed by [10]. The other methods converge toward the minimum of the function, showing significant improvement compared to the Heavy Ball. Figure 6 shows the same simulation, but with the sequences x_k plotted vs iteration, clearly showing the difference in behavior.

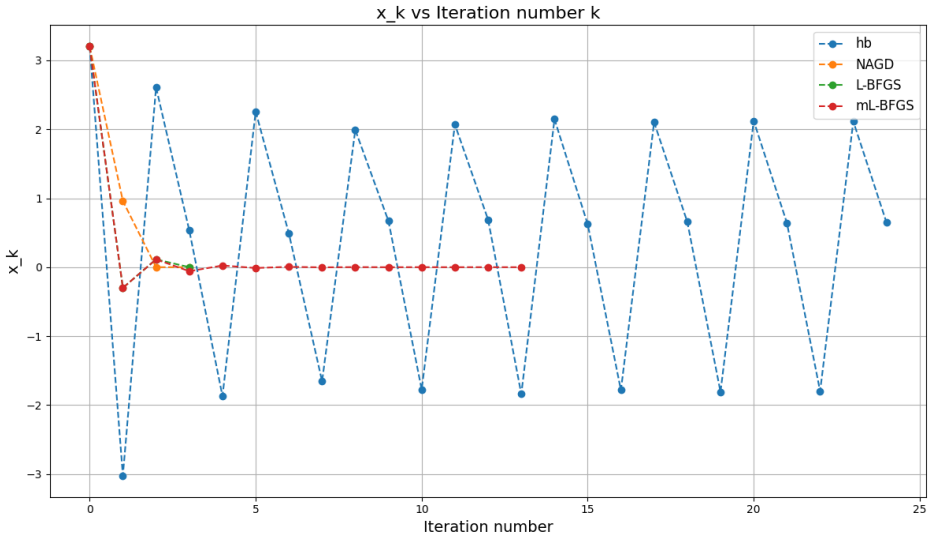


Figure 6: Comparison of points visited by each algorithm.

As seen in figure 6 all algorithms except for the Heavy Ball (blue) converges to $x_k = 0$. The NAGD and L-BFGS had the fastest convergence in the experiment, but as been seen previously the mL-BFGS seems to be more stable when the gradients are noisy. In order to compare how the methods behave with noisy gradients, the function (3.6) was used but a random noise was added to the exact gradient (3.7) in the same way as described in (3.2). In figure 7 one such simulation is shown, where the paths of x_k are plotted for the different algorithms over a number of iterations.

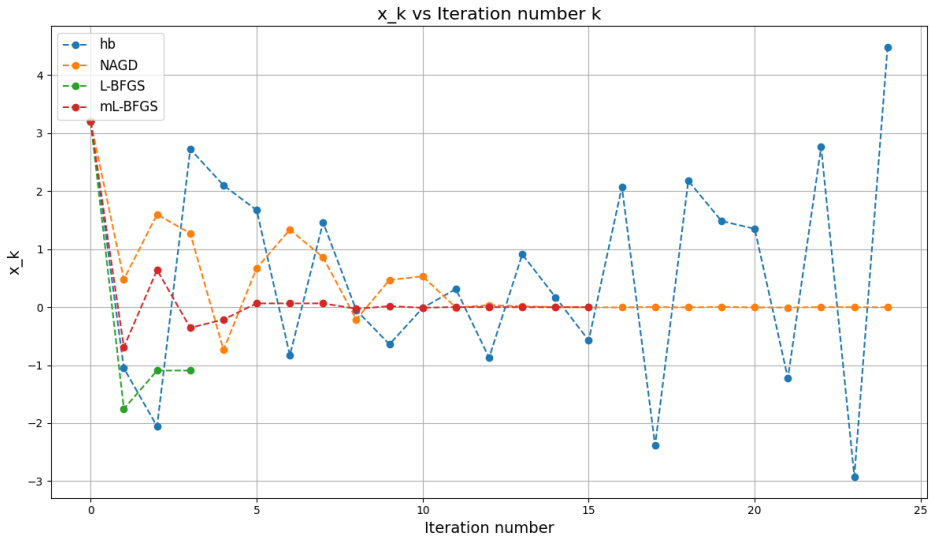


Figure 7: Comparison of points visited by each algorithm with randomly added noise, scaled with the norm of the gradient times 0.5.

In order to examine the stability of each method a number of tests with varying noise levels were performed. This was done to see how inexact gradient affected convergence of the different methods. In the test, the momentum used in mL-BFGS was held fixed at $\beta = 0.9$ as in previous tests.

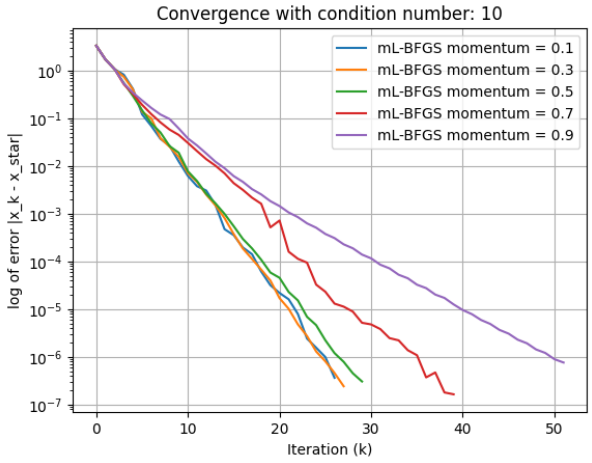
Table 3: Comparison of convergence between optimization methods under varying gradient noise levels (ψ). Memory for L-BFGS and mL-BFGS was fixed at $m = 5$. Momentum parameter β for mL-BFGS was set to 0.9.

Algorithm	Noise level (ψ)	No. Tests	Successful runs (%)	Average iterations (k) (Successful runs)
L-BFGS	0	500	100%	4.0
mL-BFGS	0	500	100%	21.0
NAGD	0	500	100%	4.0
Heavy Ball	0	500	0%	-
L-BFGS	0.1	500	100%	9.27
mL-BFGS	0.1	500	100%	20.78
NAGD	0.1	500	100%	11.97
Heavy Ball	0.1	500	100%	65.19
L-BFGS	0.2	500	99.8%	12.11
mL-BFGS	0.2	500	100%	15.27
NAGD	0.2	500	100%	15.20
Heavy Ball	0.2	500	99.6%	106.01
L-BFGS	0.3	500	93.7%	26.04
mL-BFGS	0.3	500	100%	14.75
NAGD	0.3	500	100%	18.30
Heavy Ball	0.3	500	87.8%	255.33
L-BFGS	0.4	500	72.4%	59.93
mL-BFGS	0.4	500	99.6%	15.42
NAGD	0.4	500	100%	22.90
Heavy Ball	0.4	500	7.6%	263.97
L-BFGS	0.5	500	48.8%	84.79
mL-BFGS	0.5	500	98.2%	16.48
NAGD	0.5	500	100%	27.84
Heavy Ball	0.5	500	0%	-
L-BFGS	1.0	500	5.8%	130.86
mL-BFGS	1.0	500	84.6%	27.22
NAGD	1.0	500	78.6%	203.43
Heavy Ball	1.0	500	0%	-

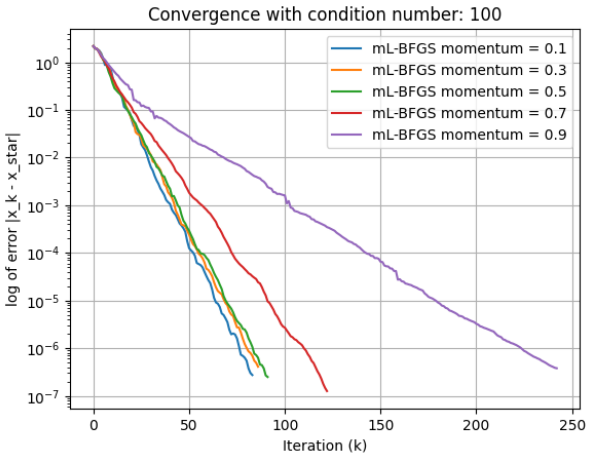
As seen in table 3, the Heavy Ball converged most of the times when a small noise is added to the gradient, but becomes unstable for larger noises. It should be noted that the initial point for all iterations was set to $x_0 = 3.2$, which is known from figure 6 to yield non convergent behavior for Heavy Ball in noise free setting. The mL-BFGS and NAGD showed best convergence rates, only for the largest noise with $\psi = 1$ the mL-BFGS performed better than NAGD. Another interesting takeaway is that the Heavy Ball did converge when the gradient was inexact, however further investigations regarding this was not done.

3.4 β's effect on mL-BFGS across varying condition numbers and ψ

This section evaluates how the choice of momentum β in mL-BFGS affects convergence for quadratic functions with varying condition number of $\nabla^2 f(x)$. The experiments aim to investigate how to choose the momentum β depending on the conditioning $\kappa(\nabla^2 f(x))$. In the simulations, a strongly convex quadratic function $f : \mathbb{R}^{40} \to \mathbb{R}$ was used.



(a) Convergence plot with different values of momentum β, condition number set to 10.

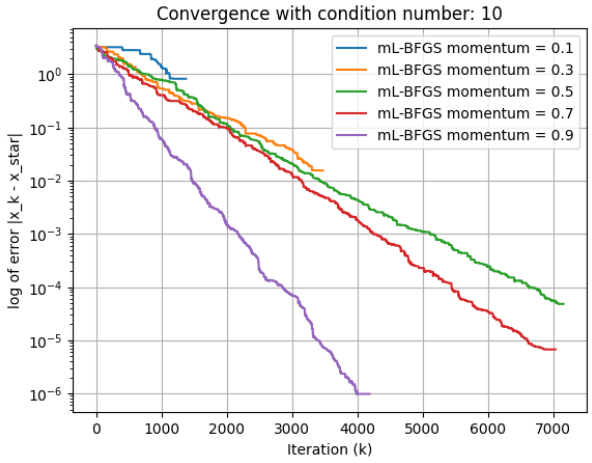


(b) Convergence plot with different values of momentum β, condition number set to 100.

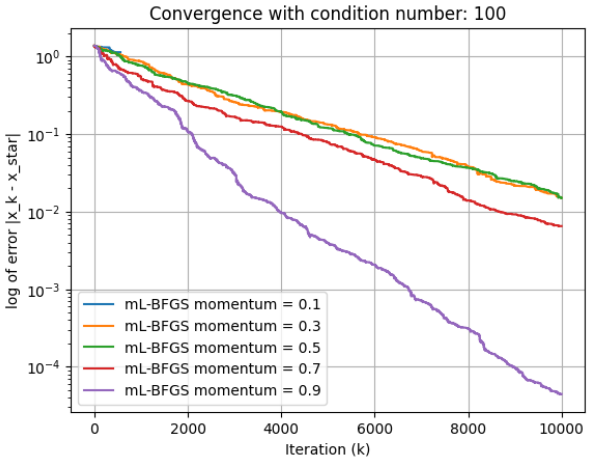
Figure 8: Numerical experiment on the influence of β when applied to strongly convex quadratic functions with varying condition number. The gradients did not contain any noise.

The results in Figure 8 captures the observed behavior from the simulations, smaller choice of β results in fewer iterations required to obtain a solution $\|x_k - x_*\| < 10^{-6}$. The experiment indicate that when the gradients are exact, momentum has slows down the convergence.

As discussed in section 2.5.2, the conditioning of the matrix A affects the sensitivity in the sense that a small change in a vector $v \in \mathbb{R}^n$ might have relatively large impact on the multiplication Ax. In order to see this, the corresponding test with noisy gradients was performed. The same method was used but where the noise level was set to $\psi = 0.5$. See figure 9



(a) Convergence plot with different values of momentum β, condition number set to 10.



(b) Convergence plot with different values of momentum β, condition number set to 100.

Figure 9: Numerical experiment on the influence of β when applied to strongly convex quadratic functions with varying condition number. The noise level was set to $\psi = 0.5$.

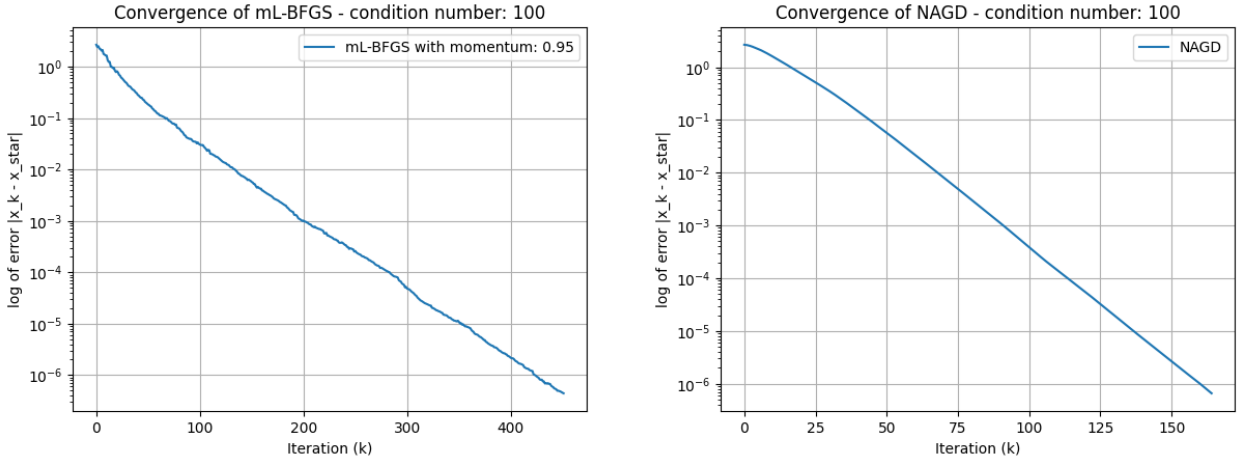
As seen in the left figure 9a, the result is almost the opposite to what was observed in the noise free setting 8a. The convergence speed increases with higher β, in contrast from what previously observed. It is also clear that even for a relatively small condition number $\kappa(A) = 10$, the mL-BFGS fails to approach x_* with a tolerance of 10^{-2} for the two smallest β values. When the condition number was increased to 100, the behavior looks similar on a log scale when comparing the different β. It is however clear that the higher condition number yields a slower convergence, as well as early stopping of the algorithms, likely because the secant condition fails to hold and the update formula becomes undefined.

Remark: As mentioned in [13] the use of damping is to be preferred, since it ensures the algorithm can handle almost orthogonal vectors $\{\tilde{s}, \tilde{y}\}$.

3.5 Comparison of mL-BFGS and NAGD on quadratic functions

In order to make a direct comparison on quadratic functions between the mL-BFGS and the NAGD method, some experiments was performed to see how they compare with and without noisy gradients. The same setup as before was used, a strongly convex quadratic function $f : \mathbb{R}^{40} \to \mathbb{R}$ was defined with controlled eigenvalues to enable testing different $\kappa(A)$.

From the previous simulations it was concluded that mL-BFGS perform best with larger momentum when gradients are noisy, so $\beta = 0.95$ was selected for this experiment. For Nesterovs method, the optimal choices of α and β was chosen according to equation (2.172).



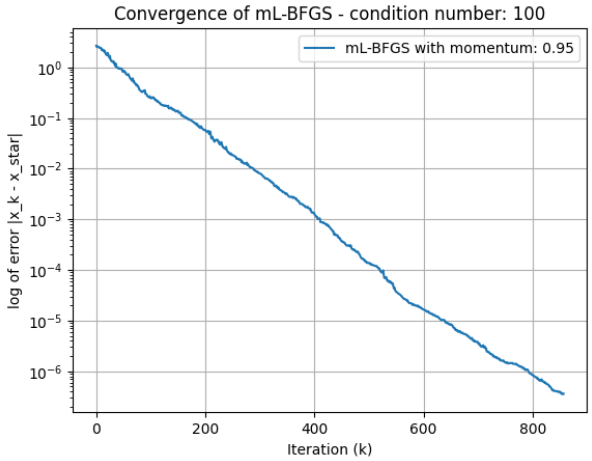
(a) Convergence plot for mL-BFGS with $\beta = 0.95$ and memory was set to $m = 5$.

(b) Convergence plot for NAGD with α and β according to equation (2.172)

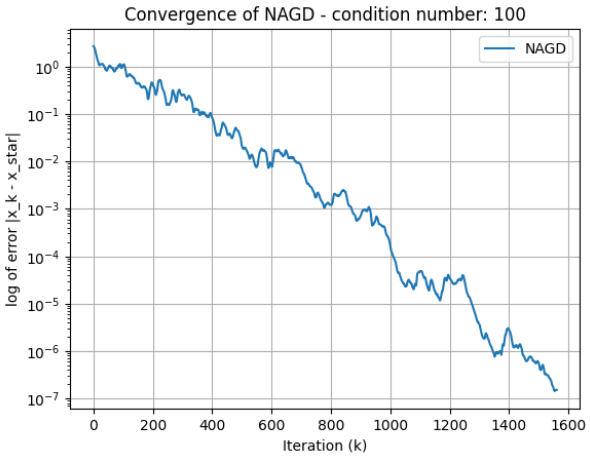
Figure 10: Comparison between mL-BFGS for a quadratic function $f(x) : \mathbb{R}^{40} \to \mathbb{R}$ with noise level set to $\psi = 0.1$.

With a noise level $\psi = 0.1$ it can be seen that both algorithms converged to the solution, see figure 10. Nesterov's method converges faster than the mL-BFGS for this noise level.

To see how the methods compare for higher values of ψ, some further analysis was made with $\psi = 0.21$.



(a) Convergence plot for mL-BFGS with $\beta = 0.95$ and memory was set to $m = 5$.

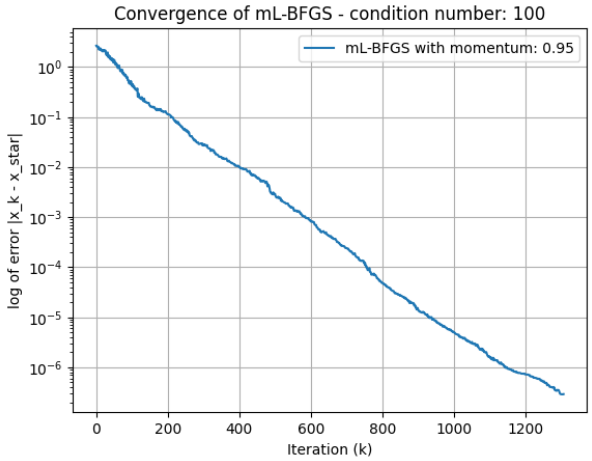


(b) Convergence plot for NAGD with α and β according to equation (2.172)

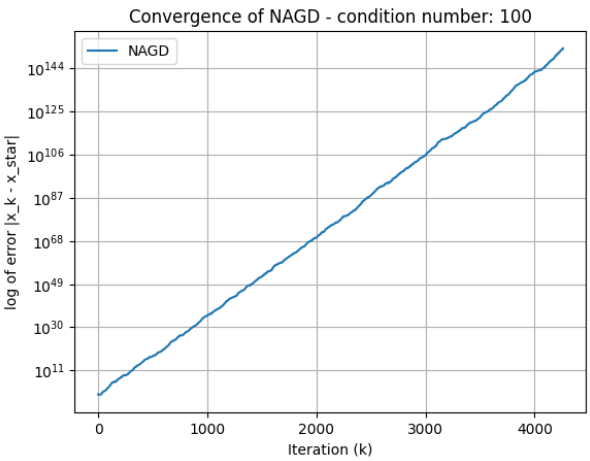
Figure 11: Comparison between mL-BFGS for a quadratic function $f(x) : \mathbb{R}^{40} \to \mathbb{R}$ with noise level set to $\psi = 0.21$.

With a noise level of $\psi = 0.21$, the number of iterations required for mL-BFGS (11a) to converge approximately doubled compared to the simulation with $\psi = 0.1$. As can be seen, the error norm still reduced steadily, without much oscillation toward the solution. The number of iterations required for NAGD (11b) to converge increased almost 10 times compared to the previous simulation, and the error is not reducing as smoothly as for the smaller noise level in (10b).

It was noted that the NAGD started diverging for larger noise level values ψ, with the approximate limit of what converges slightly larger than $\psi = 0.21$. In figure 12 ψ was set to 0.25, obtaining a divergent behavior of the NAGD method, note the magnitude on the y-axis. The mL-BFGS still converged toward the solution, even if the number of iterations increased.



(a) Convergence plot for mL-BFGS with $\beta = 0.95$ and memory was set to $m = 5$.



(b) Convergence plot for NAGD with α and β according to equation (2.172)

Figure 12: Comparison between mL-BFGS for a quadratic function $f(x) : \mathbb{R}^{40} \to \mathbb{R}$ with noise level set to $\psi = 0.25$.

To summarize, the experiments indicates that mL-BFGS is more stable as the gradient becomes more noisy.

4 Discussion and future work

In the numerical experiments a number of different objective functions and algorithms were tested. The main focus was to see how the methods react to noisy gradients, which was controlled by a noise level that scaled a random factor proportional to the gradient.

One of the interesting observations is the fact that the mL-BFGS scales well with the dimension of the Hessian compared to the other methods. NAGD showed promising results for the constructed function (3.6), performing at least as good as mL-BFGS for noise levels up to $\psi = 0.5$ as seen in table 3. However, as shown in the convergence plots in subsection (3.5) the method does not show the same stability in higher dimensions and with larger condition number in the Hessian.

One interesting observation of the mL-BFGS is the effect the choice of β had on the convergence properties shown in subsection (3.4). The simulations showed that with exact gradients a smaller momentum β is to be preferred, both with smaller and larger condition numbers, see figure 8. However, with noisy gradients the simulations shows that larger choices of β yields more stable and faster convergence. This is something that could be studied further, understanding why larger momentum values seem to be preferred with larger noise. Proposition (2.3) was an attempt to understand why smaller values of β might be yield better convergence properties without momentum, however it is not a good way to test the proposition. It does however, suggest (assuming c is constant between comparisons) that the interval of valid β reduces with higher condition numbers. It would be interesting to study this further. It would also be interesting to examine if a modified version of the L-BFGS or BFGS could be designed in a similar way to the Heavy Ball, where there exist an optimal value of the momentum β for certain functions.

No tests were performed with mL-BFGS in a stochastic setup, which was done in the original paper [13]. It would be interesting for the author to better understand how noise level correlates with batch sizes in a stochastic setup, to improve a simulated stochastic environment through noise levels.

References

[1] Roger A.Horn and Charles R.Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.

[2] S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] Sanae Lotfi, Dominique Orban, Andrea Lodi, and Tiphaine Bonniot De Ruisselet. Stochastic damped l-bfgs with controlled norm of the hessian approximation. 2020.

[4] Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *Foundations and Trends® in Optimization*, 5(1–2):1–245, 2021.

[5] Jennifer B. Erway and Roummel F. Marcia. On efficiently computing the eigenvalues of limited-memory quasi-newton matrices. *SIAM Journal on Matrix Analysis and Applications*, 36(3):1338–1359, 2015.

[6] Anders Forsgren and Tove Odland. On the connection between the conjugate gradient method and quasi-newton methods on quadratic problems. *Computational Optimization and Applications*, 60(2):377–392, July 2014.

[7] Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks, 2021.

[8] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press.

[9] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[10] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, January 2016.

[11] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45:503–528, 1989.

[12] Yurii Nesterov. *Lectures on Convex Optimization*. Springer Nature Switzerland AG, 2 edition, 2018.

[13] Yue Niu, Zalan Fabian, Sunwoo Lee, Mahdi Soltanolkotabi, and Salman Avestimehr. ml-bfgs: A momentum-based l-bfgs for distributed large-scale neural network optimization, 2023.

[14] Jorge Nocedal and Stephen J.Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

[15] Arne Persson and Lars-Christer Böiers. *Analys i en variabel*. Studentlitteratur, 3 edition, 2010.

[16] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, fourth edition, 2017.

[17] Gilbert Strang. *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press, 2019.

[18] Martin Tamm. Serier och generaliserade integraler. Stockholm University course material.

[19] Brian R. Weber. Young's, minkowski's, and hölder's inequalities. Course notes for Math 361, University of Pennsylvania, September 2011.

Errata for thesis submitted by Axel Olsson

August 19, 2025

Page (8/57)

Mixed upp Σ and Ω in the first sentence of section 1.1. Should be *Let* $f : \Omega \to \mathbb{R}$ *be a given...* instead of $f : \Sigma \to \mathbb{R}$.

Page (9/57)

Equation (1.13) is missing an exponent, should be $\langle \nabla^2 f(x^*)s, s\rangle$ i.e. Hessian and not gradient. Also poor explanation/writing following equation 1.3.

Page (10/57)

In the proof of Lemma 1.3, equation (1.26) should be rewritten. It should only state that

$$f(z) \geq f(x^*) + \frac{1}{4}\lambda_{\min}(\nabla^2 f(x^*))\langle z - x^*, z - x^*\rangle, \tag{1}$$

which then by $\nabla^2 f(x^*) \succ 0$ implies that for any $z \neq x^*$ the conclusion that

$$f(z) - f(x^*) \geq \frac{1}{4}\lambda_{\min}(\nabla^2 f(x^*))\langle z - x^*, z - x^*\rangle > 0 \tag{2}$$

holds. Hence x^* is a strict local minimizer.

Page (13/57)

Inconsequent usage of the domain of g. I write both $g : \mathbb{D} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$. In the generalization along all line segments I can use simplify to $\mathbb{R} \to \mathbb{R}$.

Page (14/57)

The sentence between equation (1.59) and (1.60) could be clarified by using a different vector than y. It could instead be written "*Since for all* $z \in \mathbb{D}$ *it is known that* $g''(z) \geq m$, *equation (1.59) can be rewritten ...*". This would make it clearer that $\xi \in \mathbb{D}$ and make the proof easier to ofllow.

Page (15/57)

In the definition of the Epigraph of a function (Def. 1.4) I use Ω instead of $\mathbb{R}^n$, here I should be consequent and use $\mathbb{R}^n$, or define Ω as the domain of f.

Page (18/57)

Equation (1.84) should be $mI \preceq \nabla^2 f(x) \preceq MI \quad \forall x \in \Omega$.

Page (20/57)

- Equation (1.91) have a poorly written denominator, should say $\|\nabla f_k\|\|p_k\|$, missed one "||". **Also** missed a parenthasis in Lemma 1.10 where I wrote $\|\nabla f(x_k\|$ instead of $\|\nabla f(x_k)\|$.
- Equation (1.96), (1.100), (1.101) and (1.102) all contains a (propagated copying error...) term $f(x_k)^T$ that should instead be $\nabla f(x_k)^T$.

Page (21/57)

Theorem 1.12 is assuming that f has Lipschitz continous gradient without making it clear. Since it uses Lemma 1.11 and that requires that for any x, y in the level set, f satisfies $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, that should have been more clearly stated in the assumptions. Furthermore, a small slipp is that I define $\mathbb{S} \subset \mathbb{R}$, it should of course be $\mathbb{S} \subset \mathbb{R}^n$.

Page (25/57)

The sentence before equation (2.16) says that "*a unique* $B(x_{k+1})$ *is chosen by solving the following optimization problem* ...". It should say $H(x_{k+1})$ instead of $B(x_{k+1})$ here as the optimization problem is defined for the inverse H.

Page (27/57)

Equation (2.22) contains one missplaced transpose. The update formula should be of the form $V^T HV + X$ and not $V^T HV^T + X$.

Page (30/57)

Equation (2.41) is missing one x term. Should be $-b^T x$ and not $-b^T$.

Page (37/57)

I wrongly refer to H_k as the approximated Hessian, it is the approximated inverse Hessian.