Tskip0pt

BFGS : An Optimzation Algorithm

University of California, Merced

Department of Applied Mathematics

Math 231 Final Project Report

Pratham Lalwani

May 13, 2025

Abstract

Optimizing functions is a very important problem which has applications across disciplines from Mathematics and Science to Finance. In cases where analytical solution doesn't exist to a problem we have to rely on optimization algorithms to find a minimum or maximum of the given function. First order methods like gradient descent although cheap are very slow at converging to a minima whereas Newton's method is quite expensive. Quasi-Newton like BFGS provide middle ground and allow a "Newton like" update without having to find the Hessian.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Optimization problems are ubiquitous. From solving linear problems to large-scale physical simulations, minimizing a function is required. Due to the wide range of optimization problems available, a good optimization algorithm is general, makes minimal assumptions, and is fast. One such algorithm is BFGS (named after Broyden, Fletcher, Goldfarb, and Shanno), which provides superlinear convergence.

BFGS is one of the most widely used algorithms for optimization. It works by approximating the Hessian of the objective function, which we want to minimize and applying symmetric updates such that its inverse of the Hessian can be computed in $\mathcal{O}(n^2)$ operations. Compared to Newton's method, which usually has a dense linear system to solve, requiring $\mathcal{O}(n^3)$ operations to solve.

Even though BFGS is more efficient than Newton's method, the $\mathcal{O}(n^2)$ cost per iteration is high. To maximize each iteration, we perform a line search along a proposed search direction to minimize the function along the search direction.

In this report we will discuss the Wolfe conditions to terminate a line search, the BFGS update, the convergence of Quasi Newton methods and their rate of convergence.

Chapter 2

Main Results

2.1 Background

We consider the unconstrained optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and bounded below. A line-search method generates iterates

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k, \tag{2.1}$$

where p_k is a search direction (e.g. a quasi-Newton direction) and $\alpha_k > 0$ is called the step size chosen to satisfy the *Wolfe conditions*:

$$f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k, \tag{2.2}$$

$$\nabla f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)^T \boldsymbol{p}_k \geq c_2 \nabla f(x_k)^T \boldsymbol{p}_k, \tag{2.3}$$

with constants $0 < c_1 < c_2 < 1$. 2.2 is called the sufficient decrease condition, which checks for decrease of the objective function along the descent direction and 2.3 only allows for a smaller directional derivative along the descent direction is called the curvature condition.

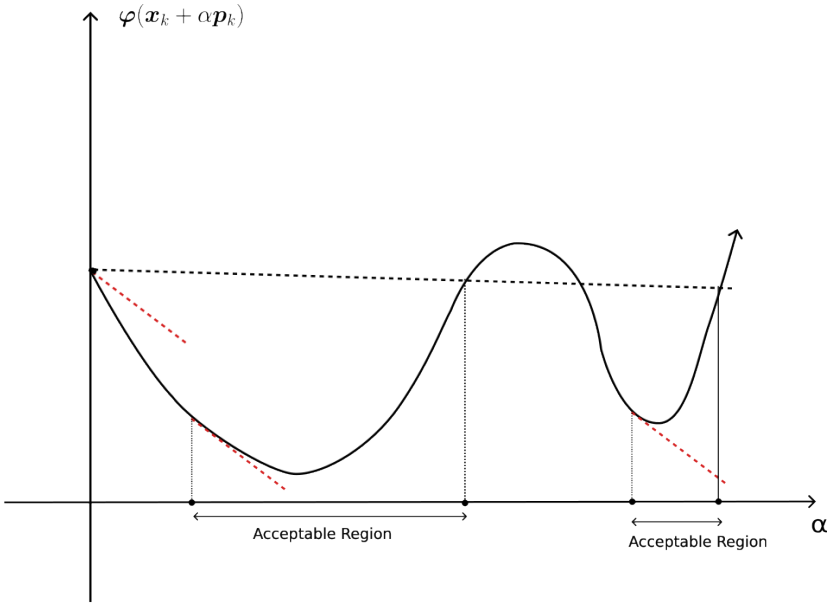


Figure 2.1: Acceptable regions for the Wolfe conditions

The existence of such and α is guaranteed by the theorem below:

Lemma 2.1.1. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. Let $\boldsymbol{p}_k$ be a descent direction at $\boldsymbol{x}_k$, and assume that f is bounded below along the ray $\{\boldsymbol{x}_k + \alpha \boldsymbol{p}_k \mid \alpha > 0\}$. Then if $0 < c_1 < c_2 < 1$, there exist intervals of step size the Wolfe conditions (2.2) and (2.3).*

A good value of $c_1 = 10^{-4}$ and $c_2 = 0.9$.[1]
Now we shall find a good $\boldsymbol{p}_k$. The best search direction is given by Newton's update, which is $\boldsymbol{p}_k = -(\nabla^2 f)^{-1}(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)$. Quasi-Newton methods usually construct an approximation $B_k \approx \nabla^2 f(\boldsymbol{x}_k)^{-1}$ to provide a search direction,

$$\boldsymbol{p}_k = -B_k^{-1}\nabla f(\boldsymbol{x}_k) \tag{2.4}$$

One such Quasi Newton method is the BFGS (Broyden, Fletcher, Goldfarb and Shanno) algorithm. BFGS directly constructs the inverse Hessian as follows:

$$H_{k+1} = H_k + \frac{\left(\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k + \boldsymbol{y}_k^{\mathrm{T}} H_k \boldsymbol{y}_k\right)\left(\boldsymbol{s}_k \boldsymbol{s}_k^{\mathrm{T}}\right)}{\left(\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k\right)^2} - \frac{H_k \boldsymbol{y}_k \boldsymbol{s}_k^{\mathrm{T}} + \boldsymbol{s}_k \boldsymbol{y}_k^{\mathrm{T}} H_k}{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k}.$$

Where $H_k \approx (\nabla^2 f)^{-1}(\boldsymbol{x}_k)$. For convenience, we define $f(\boldsymbol{x}_k) := f_k$. We shall now prove the convergence and rate of convergence of BFGS. To prove the result, we need the following lemmas:

Lemma 2.1.2 (Taylor's Theorem)**.** *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $p \in \mathbb{R}^n$. Then we have,*

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x} + t\boldsymbol{p})^T \boldsymbol{p}$$

for some $t \in (0, 1)$. Moreover, if f is twice continuously differentiable, we have that

$$\nabla f(\boldsymbol{x} + \boldsymbol{p}) = \nabla f(\boldsymbol{x}) + \int_0^1 \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p} dt$$

and that

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p}$$

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

Lemma 2.1.3. *Suppose that $f : \mathbf{R}^n \to \mathbf{R}$ is twice continuously differentiable. Consider the iteration $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$, where $\boldsymbol{p}_k$ is a descent direction and α_k satisfies the Wolfe conditions with $c_1 \leq 1/2$. If the sequence $\{\boldsymbol{x}_k\}$ converges to a point $\boldsymbol{x}^*$ such that $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite, and if the search direction satisfies*

$$\lim_{k\to\infty} \frac{\|\nabla f_k + \nabla^2 f_k \boldsymbol{p}_k\|}{\|\boldsymbol{p}_k\|} = 0 \tag{2.5}$$

then

- *the step length $\alpha_k = 1$ is admissible for all k greater than a certain index k_0; and*
- *if $\alpha_k = 1$ for all $k > k_0$, $\{x_k\}$ converges to x^* superlinearly.*

We also define the following quantity,

$$\cos(\theta_k) = -\frac{p_k^T \nabla f(\boldsymbol{x}_k)}{\|\boldsymbol{p}_k\|\|\nabla f_k\|} \tag{2.6}$$

which measures the deviation between the gradient and the descent direction.

Theorem 1. *Consider any iteration of the form* (2.1)*, where* $\boldsymbol{p}_k$ *is a descent direction and* α_k *satisfies the Wolfe conditions. Suppose that* f *is bounded below in* $\mathbb{R}^n$ *and that* f *is continuously differentiable in an open set* $\mathcal{M}$ *containing the level set* $\mathcal{L} \stackrel{def}{=} \{\boldsymbol{x} : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$*, where* $\boldsymbol{x}_0$ *is the starting point of the iteration. Assume also that the gradient* ∇f *is Lipschitz continuous on* $\mathcal{N}$*, that is, there exists a constant* $L > 0$ *such that*

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\tilde{\boldsymbol{x}})\| \leq L\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|, \quad \text{for all } \boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{N}. \tag{2.7}$$

Then

$$\sum_{k\geq 0} \cos^2\theta_k \, \|\nabla f_k\|^2 < \infty \tag{2.8}$$

it can be shown that the condition (2.5) is equivalent to the following,

$$\lim_{k\to\infty} \frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} = 0 \tag{2.9}$$

Theorem 2. *Suppose that* $f : \mathbb{R}^n \to \mathbb{R}$ *is twice continuously differentiable. Consider the iteration* $x_{k+1} = x_k + p_k$ *(that is, the step length* α_k *is uniformly 1) and that* p_k *is given by* (2.4) *Let us assume also that* $\{x_k\}$ *converges to a point* x^* *such that* $\nabla f(x^*) = 0$ *and* $\nabla^2 f(x^*)$ *is positive definite. Then* $\{x_k\}$ *converges superlinearly if and only if* (2.9) *holds.*

2.2 Theoretical Results

2.2.1 Convergence of Quasi Newton

Proof of Theorem 1. From the curvature condition (2.3) and the update (2.1) we have

$$(\nabla f_{k+1} - \nabla f_k)^T p_k \;\geq\; (c_2 - 1)\, \nabla f_k^T p_k,$$

while the Lipschitz condition (2.2) implies

$$(\nabla f_{k+1} - \nabla f_k)^T p_k \leq L\|\boldsymbol{x}_k - \alpha_k \boldsymbol{p}_k - \boldsymbol{x}_k\|\|p_k\| \leq \alpha_k \, L \, \|p_k\|^2.$$

Combining these two inequalities gives

$$\alpha_k \;\geq\; \frac{c_2 - 1}{L} \, \frac{\nabla f_k^T p_k}{\|p_k\|^2}.$$

Substituting this bound into the sufficient-decrease condition (2.2) yields

$$f_{k+1} \;\leq\; f_k - c_1 \, \alpha_k \, \nabla f_k^T p_k \;\leq\; f_k - c_1 \frac{1 - c_2}{L} \, \frac{(\nabla f_k^T p_k)^2}{\|p_k\|^2}.$$

From (2.8), this becomes

$$f_{k+1} \;\leq\; f_k \;-\; c \cos^2\theta_k \, \|\nabla f_k\|^2,$$

where $c = c_1(1 - c_2)/L$. Using the recursive relation gives

$$f_{k+1} \;\leq\; f_0 \;-\; c \sum_{j=0}^{k} \cos^2\theta_j \, \|\nabla f_j\|^2. \tag{2.10}$$

Since f is bounded below, the left-hand side of (2.10) remains finite as $k \to \infty$. Hence

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

which is the desired result. □

If we can show that $\cos^2(\theta_k) > m > 0$ then we shall have $\|\nabla f_k\| \to 0$ Suppose we have a Quasi Newton update matrix B_k which is positive definite and has a bounded condition number. It is easy to show that for any non-singular matrix A, we have the relation,

$$\frac{\|\boldsymbol{x}\|}{\|A^{-1}\|} = \|A\boldsymbol{x}\|.$$

So we have,

$$\frac{\|\boldsymbol{x}\|}{\|B^{-1}\|} = \|B\boldsymbol{x}\|.$$

Consider,

$$\begin{aligned}
\cos(\theta_k) &= -\frac{\boldsymbol{p}_k \nabla f(\boldsymbol{x}_k)}{\|\boldsymbol{p}_k\|\|\nabla f(\boldsymbol{x}_k)\|} \\
&= \frac{\boldsymbol{p}_k B_k \boldsymbol{p}_k}{\|\boldsymbol{p}_k\|\|B_k \boldsymbol{p}_k\|} \\
&\geq \frac{\|\boldsymbol{p}_k\|^2}{\|B_k^{-1}\|\|B_k\|\|\boldsymbol{p}_k\|^2} \\
&\geq \frac{1}{\kappa(B_k)}.
\end{aligned}$$

Therefore, for a Quasi Newton we have the result,

$$\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\| = 0.$$

Which means the BFGS iteration converges given a bounded condition number.

Proof of Theorem 2. We define the Newton Descent direction as follows,

$$\boldsymbol{p}_k^N = (\nabla^2 f)^{-1} \nabla f(\boldsymbol{x}_k).$$

We first show that it is equivalent to (2.9)

$$\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{N}} = o\left(\|\boldsymbol{p}_k\|\right). \tag{2.11}$$

Suppose (2.9) holds for B_k,

$$\begin{aligned}
\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{N}} &= \nabla^2 f_k^{-1} \left(\nabla^2 f_k \boldsymbol{p}_k + \nabla f_k\right) \\
&= \nabla^2 f_k^{-1} \left(\nabla^2 f_k - B_k\right) \boldsymbol{p}_k \\
&= \mathcal{O}\left(\left\|\left(\nabla^2 f_k - B_k\right) \boldsymbol{p}_k\right\|\right) \\
&= o\left(\|\boldsymbol{p}_k\|\right).
\end{aligned}$$

Now we suppose (2.11) is true,

$$\begin{aligned}
\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{N}} &= o(\|\boldsymbol{p}_k\|) \\
\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{N}}) &= o(\|\nabla f^2(\boldsymbol{x}_k)\|\|\boldsymbol{p}_k\|).
\end{aligned}$$

As we have a bounded hessian near x^*, therefore, for sufficiently large x_k we get,

$$\begin{aligned}
\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{N}}) &= o(\|\boldsymbol{p}_k\|) \\
\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{p}_k - \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{p}_k^{\mathrm{N}} &= o(\|\boldsymbol{p}_k\|) \\
\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{p}_k - \nabla f(\boldsymbol{x}_k) &= o(\|\boldsymbol{p}_k\|) \\
\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{p}_k - B_k^{-1}\boldsymbol{p}_k &= o(\|\boldsymbol{p}_k\|).
\end{aligned}$$

Which is the same as (2.9).
By combining (3.33) and (3.37), we obtain that

$$\|\boldsymbol{x}_k + \boldsymbol{p}_k - \boldsymbol{x}^*\| \leq \left\|\boldsymbol{x}_k + \boldsymbol{p}_k^{\mathrm{N}} - \boldsymbol{x}^*\right\| + \left\|\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{N}}\right\| = O\left(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2\right) + o\left(\|\boldsymbol{p}_k\|\right).$$

Now we bound $\|\boldsymbol{p}_k\|$,

$$\begin{aligned}\|\boldsymbol{p}_k\| &= \|\boldsymbol{x}_k + \boldsymbol{p}_k - \boldsymbol{x}^* - (\boldsymbol{x}_k - \boldsymbol{x}^*)\| \\ &\leq \|\boldsymbol{x}_k + \boldsymbol{p}_k - \boldsymbol{x}^*\| + \|\boldsymbol{x}_k - \boldsymbol{x}^*\| \\ &\leq \|\boldsymbol{x}_k - \boldsymbol{x}^*\| + O\left(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2\right) + o\left(\|\boldsymbol{p}_k\|\right) \\ &= O\left(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|\right).\end{aligned}$$

Therefore,

$$\|\boldsymbol{x}_k + \boldsymbol{p}_k - \boldsymbol{x}^*\| \leq o\left(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|\right),$$

which gives,

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq o\left(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|\right)$$

giving the superlinear convergence result. □

We shall now derive the Inverse Hessian update of BFGS.

BFGS Update Derivation

We would like to update the Hessian such that we have to make small updates to the matrix.

$$\begin{gathered}\min \|H - H_k\|_W^2 \quad H \in \mathbb{R}^{n\times n} \quad \text{subject to constraint,} \\ H\boldsymbol{y}_k = \boldsymbol{s}_k \quad \text{and } H^T = H.\end{gathered}$$

The norm is the weighted Frobenius norm, and W is a symmetric Positive Definite matrix. As W is SPD we have a matrix $W^{1/2}$ such that:

$$W^{1/2}W^{1/2} = W$$

Therefore, the weighted Frobenius norm is,

$$\|H - H_k\|_W^2 = \left\|W^{1/2}\left(H - H_k\right)W^{1/2}\right\|_F^2$$

We define the Lagrange Multiplier Problem as follows:

$$\mathcal{L}(H, \lambda, \theta) = \|H - H_k\|_W^2 + \operatorname{tr}\left(\lambda^T\left(Hy_k - \boldsymbol{s}_k\right)\right) + \left(H^T - H\right)\Theta \tag{2.12}$$

Taking the respective partials,

$$\begin{aligned}&\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = H\boldsymbol{y}_k - \boldsymbol{s}_k = 0 \\ &\frac{\partial \mathcal{L}}{\partial \Theta} = O \implies H^T = H \\ &\frac{\partial \mathcal{L}}{\partial H} = W\left(H - H_k\right)W + \boldsymbol{\lambda}\boldsymbol{y}_k^T + \Theta - \Theta^T = 0 \\ &\therefore W H_W W = W H_k W + \boldsymbol{\lambda}\boldsymbol{y}_k^T + \Theta - \Theta^T.\end{aligned} \tag{2.13}$$

Applying the transpose on both sides

$$WH_kW = WH_kW + \boldsymbol{y}_k^T\boldsymbol{\lambda}^T - \left(\Theta - \Theta^T\right) \tag{2.14}$$

Subtracting (2.14) from (2.13) :

$$0 = \left(\boldsymbol{\lambda}\boldsymbol{y}_k^T - \boldsymbol{y}_k\boldsymbol{\lambda}^T\right) + 2\left(\Theta - \Theta^T\right)$$

$$\frac{1}{2}\left(\boldsymbol{y}_k\boldsymbol{\lambda}^T - \boldsymbol{\lambda}\boldsymbol{y}_k^T\right) = \Theta - \Theta^T \tag{2.15}$$

Using (2.15) in (2.13)

$$WH_kW = WH_kW + \frac{1}{2}\left(\boldsymbol{\lambda}y_k^T + y_k\boldsymbol{\lambda}^T\right)$$

.

$$H_k = H_k + \frac{1}{2}W^{-1}\left(\boldsymbol{\lambda}_{\boldsymbol{\lambda}}\boldsymbol{y}_k^T + \boldsymbol{y}_k\boldsymbol{\lambda}^T\right)W^{-1}$$

We also have

$$H\boldsymbol{y}_k = \boldsymbol{s}_k$$

We also didn't choose a specific matrix W, one good choice is $W\boldsymbol{s}_k = \boldsymbol{y}_k$. Which results in the BFGS update formula.

$$H_k\boldsymbol{y}_k + \frac{1}{2}W^{-1}\left(\boldsymbol{\lambda}\boldsymbol{y}_k^T + \boldsymbol{y}_k\boldsymbol{\lambda}^T\right)W^{-1}\boldsymbol{y}_k = \boldsymbol{s}_k$$

$$H_k\boldsymbol{y}_k + \frac{1}{2}W^{-1}\boldsymbol{\lambda}\boldsymbol{y}_k^TW^{-1}\boldsymbol{y}_k + \frac{1}{2}W^{-1}\boldsymbol{y}_k\boldsymbol{\lambda}^TW^{-1}\boldsymbol{y}_k = \boldsymbol{s}_k$$

$$WH_k\boldsymbol{s}_k + \frac{1}{2}\left(\boldsymbol{\lambda}\boldsymbol{y}_k^T + \boldsymbol{y}_k^T\boldsymbol{\lambda}^T\right)W^{-1}\boldsymbol{y}_k = W\boldsymbol{s}_k$$

$$\frac{1}{2}\left(\boldsymbol{\lambda}\boldsymbol{y}_kW^{-1}\boldsymbol{y}_k + \boldsymbol{y}_k\boldsymbol{\lambda}^TW^{-1}\boldsymbol{y}_k\right) = W\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_{\boldsymbol{k}}\right)$$

$$\boldsymbol{\lambda} = \frac{2W\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_k\right) - \boldsymbol{y}_k\boldsymbol{\lambda}^TW^{-1}\boldsymbol{y}_k}{\boldsymbol{y}_kW^{-1}\boldsymbol{y}_k}$$

Multiplying both sides by $\boldsymbol{y}_k^TW^{-1}$ and applying transpose.

$$\boldsymbol{\lambda}^TW^{-1}\boldsymbol{y}_k = \left(\frac{2(\boldsymbol{y}_k^T\boldsymbol{s}_k - \boldsymbol{y}_k^TH_k\boldsymbol{y}_k) - \left(\boldsymbol{y}_k^TW^{-1}\boldsymbol{y}_k\right)\left(\boldsymbol{\lambda}^TW^{-1}\boldsymbol{y}_k\right)}{\boldsymbol{y}_k^TW^{-1}y_k}\right)$$

$$\boldsymbol{\lambda}^TW^{-1}\boldsymbol{y}_k = \frac{\boldsymbol{y}_k^T\boldsymbol{s}_k - \boldsymbol{y}_k^TH_k\boldsymbol{y}_k}{\boldsymbol{y}_k^TW^{-1}y_k}$$

$$\therefore \boldsymbol{\lambda} = \frac{2W\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_k\right)}{\boldsymbol{y}_k^TW^{-1}\boldsymbol{y}_k} - \frac{\boldsymbol{y}_k\left(\boldsymbol{y}_k^T\boldsymbol{s}_k - \boldsymbol{y}_k^TH_k\boldsymbol{y}_k\right)}{\left(\boldsymbol{y}_k^TW^{-1}\boldsymbol{y}_k\right)^2}$$

with $W\boldsymbol{s}_k = \boldsymbol{y}_k \implies \boldsymbol{s}_k = W^{-1}\boldsymbol{y}_k$

$$\boldsymbol{\lambda} = \frac{2W\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_k\right)}{\boldsymbol{y}_k^T\boldsymbol{s}_k} - \frac{\boldsymbol{y}_k\left(\boldsymbol{y}_k^T\boldsymbol{s}_k - \boldsymbol{y}_k^TH_k\boldsymbol{y}_k\right)}{\left(\boldsymbol{y}_k^T\boldsymbol{s}_k\right)^2}$$

$$H = H_k + \frac{1}{2}W^{-1}\left(\left(\frac{2W\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_k\right)}{\boldsymbol{y}_k^T\boldsymbol{s}_k}\right)\boldsymbol{y}_k^T - \boldsymbol{y}_k\boldsymbol{y}_k^T\frac{\left(\boldsymbol{y}_k^T\boldsymbol{s}_k - \boldsymbol{y}_k^TH_k\boldsymbol{y}_k\right)}{\left(\boldsymbol{y}_k^T\boldsymbol{s}_k\right)^2}\right.$$

$$\left. + \frac{2\boldsymbol{y}_k\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_k\right)^TW}{\boldsymbol{y}_k^T\boldsymbol{s}_k} - \boldsymbol{y}_k\boldsymbol{y}_k^T\left(\frac{\boldsymbol{y}_k^T\boldsymbol{s}_k - \boldsymbol{y}_k^TH_k\boldsymbol{y}_k}{\left(\boldsymbol{y}_k^T\boldsymbol{s}_k\right)^2}\right)\right)W^{-1}$$

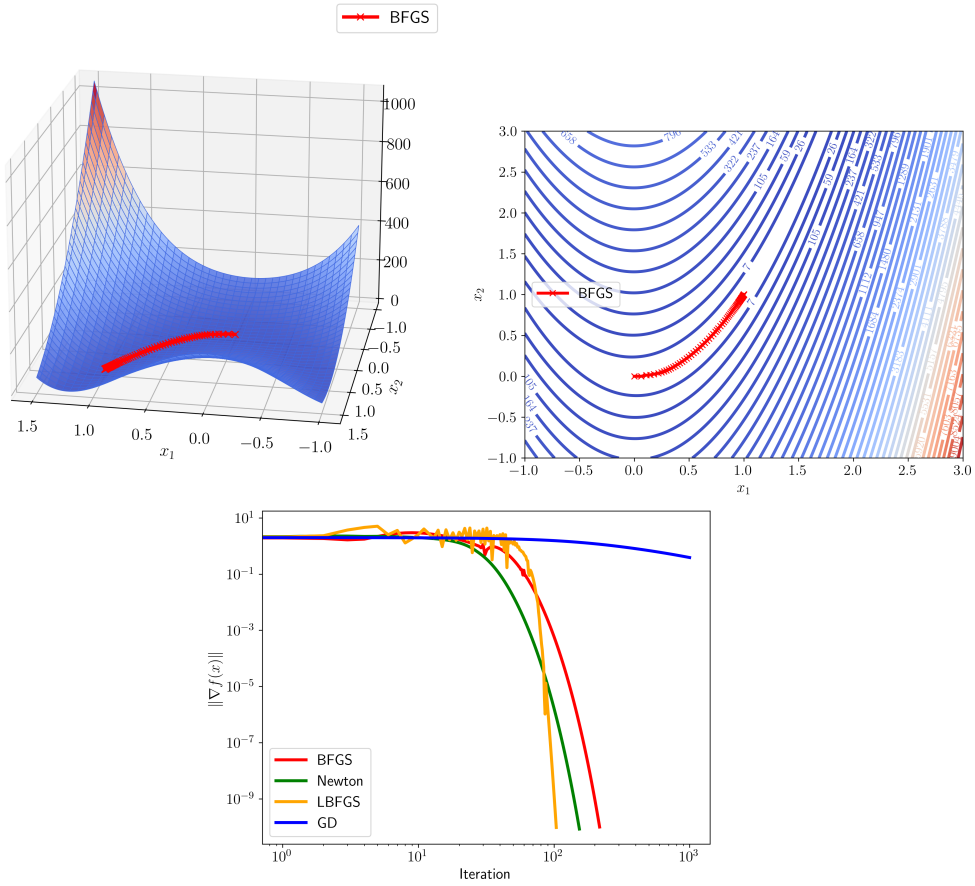
$$H = H_k + \frac{\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_k\right)\boldsymbol{s}_k^T}{\boldsymbol{y}_k^T\boldsymbol{s}_k} - \frac{\boldsymbol{s}_k\boldsymbol{s}_k^T\left(\boldsymbol{y}_k^T\boldsymbol{s}_k - \boldsymbol{y}_k^TH_k\boldsymbol{y}_k\right)}{\left(\boldsymbol{y}_k^T\boldsymbol{s}_k\right)^2} + \frac{\boldsymbol{s}_k\left(\boldsymbol{s}_k - H_k\boldsymbol{y}_k\right)^T}{\boldsymbol{y}_k^T\boldsymbol{s}_k}$$

We leave it here because this is the efficient way to compute it.

2.3 Numerical Results

2.3.1 Examples

Rosenbrock Function



(a) Convergence of various methods on the Rosenbrock function

Figure 2.2: BFGS and Newton applied on the Rosenbrock function

2.3.2 Comparitive study of BFGS

Every algorithm has been given 1000 max iterations and tolerance of norm of gradient is less than 1e-6.
We compare four algorithms BFGS, Newton, L-BFGS (A low memory variant of BFGS) and Gradient Descent (GD).
From the tables we can see that BFGS and L-BFGS converge fairly well at the cost of more function evaluations than Newton. We can se the cost per iteration is the lowest for Gradient Descent which is expected but it also didn't converge for a lot of problem whereas BFGS and L-BFGS provide a middle ground between accuracy and cost per iteration. This is especially apparent for the 100-D problem. The non convergence of BFGS in certain places is the fault of my implementation as line search might fail and there was no simple way of handling it.

Table 2.1: Final gradient norm of each method

Function	BFGS	Newton	LBFGS	GD
Adjiman Function(2 - D)	nan	$3.72e-16$	$5.50e-16$	$7.70e-01$
Rosenbrock N-D(100-D)	$9.28e-11$	$8.52e-11$	$9.24e-11$	$2.92e+00$
Paviani Function(10 - D)	nan	nan	nan	$8.72e-02$
Csendes Function(10 - D)	$9.57e-11$	$1.21e-03$	$9.16e-11$	$7.48e-02$
Griewank Function(2 - D)	nan	$1.26e-15$	6.65e-11	$8.31e-09$
Hosaki Function(2 - D)	nan	$8.83e-11$	nan	$8.66e-02$
Brent Function(2-D)	$9.00e-11$	$9.46e-11$	$9.90e-11$	$3.67e+00$
Giunta Function(2 - D)	$2.22e-15$	$1.57e-16$	$2.67e-15$	$1.29e-08$
Styblinski-Tang Function(2-D)	$2.93e-14$	$0.00e+00$	$6.39e-14$	$9.96e-11$
Trid 6 Function(6-D)	nan	$7.60e-07$	nan	$4.08e+00$

Table 2.2: Time per iteration (in seconds)

Function	BFGS	Newton	LBFGS	GD
Adjiman Function(2-D)	0.01741	0.02919	0.01431	0.004024
Rosenbrock N-D(100-D)	0.01578	0.02691	0.01229	0.003553
Paviani Function(10 - D)	0.4714	0.4163	0.01651	0.003982
Csendes Function(10 - D)	0.007706	0.06964	0.008144	0.00223
Griewank Function(2 - D)	0.01202	0.01461	0.01285	0.004034
Hosaki Function(2 - D)	0.02751	0.04954	0.02716	0.007989
Brent Function(2-D)	0.01732	0.03552	0.01729	0.005377
Giunta Function(2 - D)	0.01624	0.02645	0.01503	0.005281
Styblinski-Tang Function(2-D)	0.01263	0.0217	0.0127	0.004353
Trid 6 Function(6-D)	0.01533	0.05652	0.01025	0.002685

Table 2.3: Number of Function Evaluations

Function	BFGS	Newton	LBFGS	GD
Adjiman Function(2 - D)	1426	956	1124	1001
Rosenbrock N-D(100 - D)	4744	2465	5642	1001
Paviani Function(10 - D)	1012	7	5	1001
Csendes Function(10 - D)	7826	169638	1350	1001
Griewank Function(2 - D)	2233	31	987	1001
Hosaki Function(2 - D)	2352	2060	1256	1001
Brent Function(2 - D)	2789	2021	1395	1001
Giunta Function(2 - D)	2217	1348	980	1001
Styblinski-Tang Function(2-D)	2118	1220	879	730
Trid 6 Function(6-D)	2020	89229	883	1001

Table 2.4: Number of Iterations

Function	BFGS	Newton	LBFGS	GD
Adjiman Function(2-D)	Did not converge	72	148	Did not converge
Rosenbrock N-D(100-D)	333	177	775	Did not converge
Paviani Function(10 - D)	Did not converge	Did not converge	Did not converge	Did not converge
Csendes Function(10 - D)	708	Did not converge	191	Did not converge
Griewank Function(2 - D)	Did not converge	6	138	1001
Hosaki Function(2 - D)	Did not converge	148	Did not converge	Did not converge
Brent Function(2-D)	199	145	200	Did not converge
Giunta Function(2 - D)	135	97	138	1001
Styblinski-Tang Function(2 - D)	124	90	127	730
Trid 6 Function(6-D)	Did not converge	1001	Did not converge	Did not converge

Chapter 3

Conclusion

The report provides introductory look at BFGS algorithm and Quasi Newton algorithms in general, including line-search theory, convergence proofs, and the inverse Hessian update derivation. We were able to prove convergence under very restricted set of conditions, more can be said about BFGS under relaxed condition as outlined in [1]. We proved under the restricted conditions that BFGS is globally convergent and achieves superlinear convergence near a non-degenerate minimizer. Numerical experiments verify our claims, it is much better than gradient descent and on par with Newton without paying for the linear system solve.
The biggest problem I encountered during this project was line search. Making a line search algorithm is quite difficult as they need to be fairly quick at finding an appropriate step size. I also had some difficulty in making my BFGS algorithm robust because there are so many ways it can go wrong from numerical instabilities to very small curvatures which make change in approximate Hessian quite big. But overall, the project was a very good experience and has definitely increased my appreciation for libraries.

Appendix

A Additional calculations or proof of lemmas

B Pseudocode of your algorithm

Algorithm 1 BFGS with Wolfe Line Search

Input: x_0, $B_0 = I$, tolerances ε
for $k = 0, 1, 2, \ldots$ **do**
    $p_k \leftarrow -B_k \nabla f(x_k)$
    Find α_k satisfying Wolfe conditions
    $s_k \leftarrow \alpha_k p_k, \quad y_k \leftarrow \nabla f(x_k + s_k) - \nabla f(x_k)$
    Update B_{k+1} via BFGS formula
    if $\|\nabla f(x_{k+1})\| < \varepsilon$ **then break**
    end if
end for
Output: Approximate minimizer x_{k+1}

Bibliography

[1] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2nd edition, 2006.