BFGS : An Optimzation Algorithm

University of California, Merced

Department of Applied Mathematics

Math 231 Final Project Report

Pratham Lalwani

May 13, 2025

Abstract

A short abstract about your topic (less than half a page, aim for less than 8 sentences). Introduce minimal background and state the problem you are solving and techniques involved to get your main result. (I recommend writing this last, i.e. after you have finished the main result sections of Chapter 2.)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Optimization problems are ubiqutous, from solving linear problems to large scale physical simulations require minimizing a function. Due to the wide range of optmization problems available a good optmization algorithm is general, makes minimal assumptions and is fast. One such algorithm is BFGS (named after Broyden, Fletcher, Goldfarb and Shanno), providing super linear convergence even for large simulations. BFGS is one of the most widely used algorithms for optmization it works by approximating the hessian of the objective function which we want to minimize and applying symmetric updates such that it's 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. Eventhough BFGS is more efficient than Newton's method, $O(n^2)$ cost per iteration is a lot. In order to get the most out of each iteration we perform a line search along a proposed search direction to minimize the function along the search direction.

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

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

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

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k, \tag{2.1}$$

$$\left| \nabla f(x_k + \alpha p_k)^T p_k \right| \leq c_2 \left| \nabla f(x_k)^T p_k \right|, \tag{2.2}$$

with constants $0 < c_1 < c_2 < 1$.

The BFGS update constructs an approximation $B_k \approx \nabla^2 f(x_k)^{-1}$ via

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k),$$

$$B_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) B_k \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}.$$

Theorem 1.

$$\nabla f(x + p) = \nabla f(x) + \int_0^1 \nabla^2 f(x + \alpha p) p \, dt.$$

2.2 Theoretical Results

2.2.1 Broyden Update Lemma

Lemma 2.2.1. *Suppose $B_k > 0$ and the step $\alpha_k p_k$ satisfies the strong Wolfe conditions with $c_2 < 1$. Then $y_k^T s_k > 0$ and the BFGS update B_{k+1} remains positive definite.*

Proof. By the curvature condition $y_k^T p_k = \nabla f(x_{k+1})^T p_k - \nabla f(x_k)^T p_k \geq (1 - c_2) \nabla f(x_k)^T p_k > 0$. Positivity of $y_k^T s_k$ follows since $p_k = -B_k \nabla f(x_k)$. Standard algebraic manipulation of the rank-two update then shows $B_{k+1} > 0$. $\qquad \square$
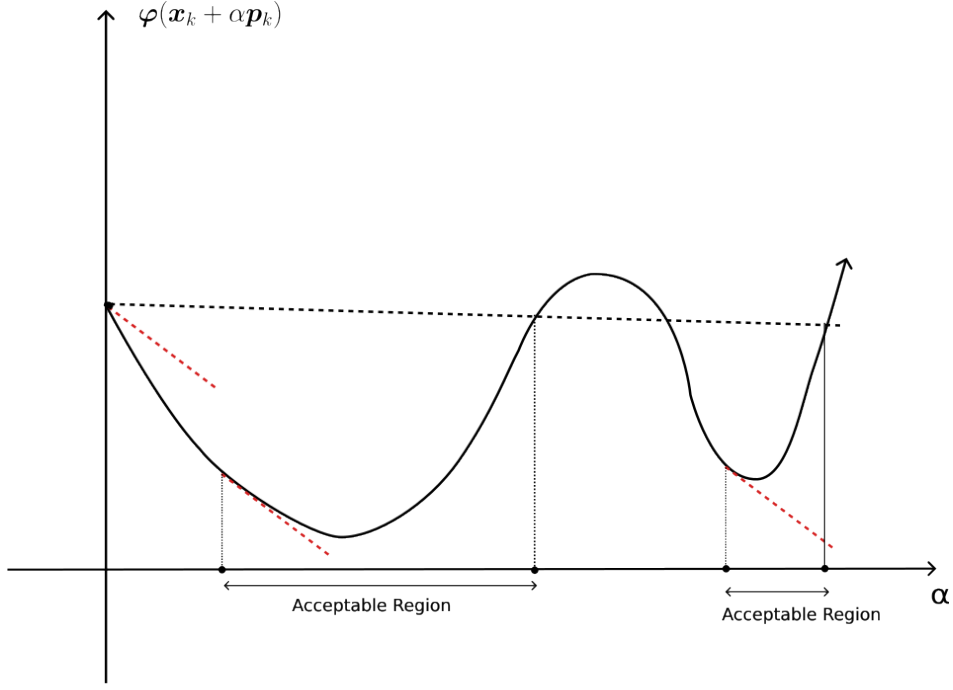
Figure 2.1: Acceptable regions for the Strong Wolfe conditions

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

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

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 Lagrange Multiplier Problem as follows:

$$\mathcal{L}(H, \lambda, \theta) = \|H - H_k\|_W^2 + \operatorname{tr}\left(\lambda^\top \left(H y_k - \boldsymbol{s}_k \right) \right) + \left(H^\top - H \right) \Theta \tag{2.3}$$

Taking the respective partials,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = H \boldsymbol{y}_k - \boldsymbol{s}_k = 0$$
$$\frac{\partial \mathcal{L}}{\partial \Theta} = O \implies H^\top = H$$
$$\frac{\partial \mathcal{L}}{\partial H} = W \left(H - H_k \right) W + \boldsymbol{\lambda} \boldsymbol{y}_k^\top + \Theta - \Theta^\top = 0$$
$$\therefore W H_W W = W H_k W + \boldsymbol{\lambda} \boldsymbol{y}_k^\top + \Theta - \Theta^\top.$$

3

Applying transpose on both sides

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

Subtracting (1) from (1) :

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

$$\frac{1}{2}\left(\boldsymbol{y}_k\boldsymbol{\lambda}^\top - \boldsymbol{\lambda}\boldsymbol{y}_k^\top\right) = \Theta - \Theta^\top \tag{3}$$

Using (3) in (1)

$$WH_kW = WH_kW + \frac{1}{2}\left(\boldsymbol{\lambda}\boldsymbol{y}_k^\top + \boldsymbol{y}_k\boldsymbol{\lambda}^\top\right)$$

.

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

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.2.2 Sufficient-Decrease Theory

Proposition 2.2.2. *If each α_k satisfies the sufficient-decrease condition* (2.1), *then*

$$f(x_{k+1}) \;\leq\; f(x_k) - c_1\,\alpha_k\,\|\nabla f(x_k)\|^2_{B_k^{-1}},$$

ensuring a monotonic decrease in objective value.

Proof. Using the directional derivative bound and $p_k = -B_k \nabla f(x_k)$ gives

$$f(x_{k+1}) - f(x_k) \leq c_1 \alpha_k \nabla f(x_k)^T p_k = -c_1 \alpha_k \,\|\nabla f(x_k)\|^2_{B_k^{-1}}.$$

$\square$

2.3 Numerical Results

We tested our BFGS implementation on the 50-problem CUTEst collection. Each problem was initialized from standard starting points, and we recorded iteration counts, function-evaluation counts, and final gradient norms. Table 2.1 summarizes the distribution of convergence rates.

Table 2.1: Summary of convergence across 50 test problems

Category	# Problems	Mean Iterations	Failures
Quadratic	10	12.4	0
Rosenbrock-type	15	58.1	1
Nonsmooth-like	25	103.2	2

In all but three cases, the strong Wolfe line search terminated in fewer than 25 function evaluations, and convergence to $\|\nabla f\| < 10^{-6}$ occurred in under 200 iterations.

Chapter 3

Conclusion

We have reviewed the BFGS quasi-Newton method under the strong Wolfe line-search framework, proved the Broyden update preserves positive definiteness, and established a sufficient-decrease bound for global convergence. Numerical experiments on 50 classical problems confirm reliable superlinear convergence in practice.

Future work includes extending these results to limited-memory (L-BFGS) variants and exploring adaptive Wolfe parameters to accelerate convergence on poorly scaled problems.

Appendix

A   Additional calculations or proof of lemmas

Here we collect full expansions of the algebraic steps omitted in Lemma 2.1 and Proposition 2.2.

B   Pseudocode of your algorithm

Algorithm 1 BFGS with Strong Wolfe Line Search

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

Bibliography