UNIVERSITY OF PADUA
Department of Mathematics

Bachelor Degree in Mathematics

A Variational Derivation of a Class of BFGS-like Methods

Supervisor:
Professor Michele Pavon

Candidate:
Aurora Milani
Student Number:
1121701

December 11th, 2020 - A. Y. 2019/2020

Contents

Introduction

People optimize. Airline companies schedule crews and aircraft to minimize cost. Investors seek to create portfolios that avoid excessive risks while achieving a high rate of return. Manifacturers aim for maximum efficiency in the design and operation of their production processes.

Nature optimize. Physical systems tend to a state of minimum energy. The molecules in an isolated chemical system react with each other until the total potential energy of their electrons is minimized. Rays of light follow paths that minimize their travel time.

Optimization is an important tool in decision science and in the analysis of physical systems. To use it, we must first identify some *objective function* (e.g. cost to be minimized), its *variables* and its *constraints.* Once the model has been formulated, an optimization algorithm can be used to find its solution (very few interesting optimization problems admite a *closed form* solution). There are numerous algorithms. In this work, we focus on quasi-Newton methods and, in particular, the most popular one, the BFGS method. We first present the classical derivation of the BFGS algorithm [4], [2]. Then, we provide a variational derivation of the BFGS-like iteration, based on the work of M. Pavon [1]. In [3], Fletcher, one of the discoverers of the BFGS algorithm, had already provided a variational characterization for the BFGS iteration. However, we take a different approach, leading to a new family of BFGS-like methods and an independent proof of a result of Fletcher.

First, in Chapter 1, we recall some useful knowledge of positive definite matrices, normed spaces and the Kullback-Leibler divergence.

Then, in Chapter 2, we give the mathematical background of optimization, recalling the definitions and the main results on local and global minima of a function, and convex sets and functions.

In Chapter 3, we present line search methods: they are strategies used by an optimization algorithm for moving from the current point to a new iterate. We analyse some of these strategies: the steepest descent method, Newton's method and the quasi-Newton method.

In Chapter 4, we study the BFGS method, focusing on its classical derivation.

In Chapter 5, we provide the variational derivation of a class of BFGS-like methods mentioned above.

Finally, in Chapter 6, we present some numerical experiments comparing the classical BFGS method and the BFGS-like one.

Chapter 1

Mathematical background

In this Chapter, we want to briefly recall some results that shall be useful for this work. First, we discuss positive definiteness of matrices, in particular we illustrate its characterizations. Then, we talk about metrics and norms, with particular attention on matricial norms and Frobenius norm. Finally, we give the definition and the explanation of the Kullback-Leibler divergence, focusing on multivariate Gaussian distributions.

1.1 Positive definite matrices

In this section, we shall consider only real matrices, but most of the definitions and the results can be extended to complex matrices.

First, we give the definition of a *symmetric* matrix.

Definition 1.1 (Symmetric matrix)**.** A matrix $A \in \mathbb{R}^{n\times n}$ is *symmetric* if $A^T = A$, where A^T means the transpose of A.

The main result concerning real symmetric matrices is the *real spectral theorem*: before stating it, we recall some definitions.

Definition 1.2 (Eigenvalue)**.** Let A be a real $n \times n$ matrix. $\lambda \in \mathbb{C}$ is an *eigenvalue* of A if there exists some $x \in \mathbb{R}^n$, $x \neq 0$, such that $Ax = \lambda x$.

Definition 1.3 (Diagonalizable matrix)**.** Let A be a real $n \times n$ matrix. A is called *diagonalizable* if it is similar to a diagonal matrix, i.e. if there exists an invertible matrix P and a diagonal matrix D such that $P^{-1}AP = D$.

Definition 1.4 (Orthogonal matrix)**.** Let A be a real $n \times n$ matrix. A is called *orthogonal* if $A^T = A^{-1}$.

Definition 1.5 (Orthogonally diagonalizable matrix)**.** Let A be a real $n \times n$ matrix. A is *orthogonally diagonalizable* if it is orthogonally similar to a diagonal matrix, i.e. if there exists an orthogonal matrix P and a diagonal matrix D such that $P^TAP = D$.

We are ready to state the real spectral theorem.

Theorem 1.1 (Real spectral theorem)**.** *Let A be a real $n \times n$ matrix. Then, A is orthogonally diagonalizable if, and only if, A is symmetric.*

We now talk about definite and semi-definite matrices.

Definition 1.6 (Semi-definite matrix)**.** A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is *positive semi-definite* if

$$x^T Ax \geq 0 \quad \forall x \in \mathbb{R}^n.$$

Moreover, a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is negative semi-definite if

$$x^T Ax \leq 0 \quad \forall x \in \mathbb{R}^n.$$

We shall use the notation $A \geq 0$ for a positive semi-definite matrix and $A \leq 0$ for a negative semi-definite one.

Definition 1.7 (Definite matrix)**.** A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive definite if

$$x^T Ax > 0 \quad \forall x \in \mathbb{R}^n \setminus \{0\}.$$

Moreover, a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is negative definite if

$$x^T Ax < 0 \quad \forall x \in \mathbb{R}^n \setminus \{0\}.$$

We shall use the notation $A > 0$ for a positive definite matrix and $A < 0$ for a negative definite one.

Before giving the first characterization of a positive definite matrix, we define the *spectrum* of a matrix.

Definition 1.8 (Spectrum of a matrix)**.** The *spectrum* of a matrix is the set of its eigenvalues. If the matrix is A, we denote its spectrum by $\sigma(A)$.

Proposition 1.2 (First characterization of positive definiteness)**.** Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then

$$A > 0 \quad \Leftrightarrow \quad \sigma(A) \subset \mathbb{R}^+,$$

that is all eigenvalues of A are real and positive.

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. We have just discovered that we can determine the positive definiteness of A by computing its eigenvalues. Another method is to use the principal minors.

Definition 1.9 (Minor)**.** A *minor* of order k of a $n \times n$ matrix A is a submatrix of A obtained by deleting $n - k$ rows and $n - k$ columns.

Definition 1.10 (Principal and leading principal minor)**.** A minor of order k of a $n \times n$ matrix A is *principal* if it is obtained by deleting $n-k$ rows and the $n-k$ columns with the same numbers.
The *leading principal* minor of A of order k is the minor of order k obtained by deleting the last $n-k$ rows and columns. We write A_k for the leading principal minor of order k.

Proposition 1.3 (Second characterization of positive definiteness)**.** Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then

$$A > 0 \quad \Leftrightarrow \quad det(A_k) > 0 \quad \forall k = 1, \dots, n.$$

We give the last characterization of a positive definite matrix.

Proposition 1.4 (Third characterization of positive definiteness)**.** Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then the following statements are equivalent:

- $A > 0$;
- $\exists m > 0$ constant : $x^T Ax \geq m\|x\|^2 \quad \forall x \in \mathbb{R}^n$.

Here $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$.

1.2 Metrics and norms

Let X be an arbitrary nonempty set.

Definition 1.11 (Metric or distance)**.** A *metric*, or *distance*, on X is a function $d\colon X \times X \to \mathbb{R}$ with the following properties:

(i) $d(x, y) \geq 0$ for all $x, y \in X$, and $d(x, y) = 0$ if and only if $x = y$;

(ii) (symmetry) $d(x, y) = d(y, x)$, for all $x, y \in X$;

(iii) (triangle inequality) $d(x, y) \leq d(x, z) + d(z, y)$, for all $x, y, z \in X$.

Definition 1.12 (Metric space)**.** A *metric space* (X, d) is a nonempty set X with a metric d.

We give some examples of metric spaces:

- the set of real numbers $\mathbb{R}$ with the distance $d(x, y) = |x - y|$, where $|\cdot|$ denotes absolute value;
- the set of complex numbers $\mathbb{C}$ with the distance $d(x, y) = |x - y|$, where $|\cdot|$ is the complex modulus;

- (Euclidean metric on $\mathbb{R}^k$) the set $\mathbb{R}^k$ with the distance

$$d(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}.$$

Definition 1.13 (norm)**.** A *norm* on a vector space X over $\mathbb{R}$ (or $\mathbb{C}$) is a function $\|\cdot\|\colon X \to \mathbb{R}$ with the following properties:

(i) (nonegative) $\|x\| \geq 0$, for all $x \in X$;

(ii) $\|x\| = 0$ implies that $x = 0$;

(iii) (homogeneous) $\|\lambda x\| = |\lambda|\|x\|$, for all $x \in X$ and $\lambda \in \mathbb{R}$ (or $\mathbb{C}$);

(iv) (triangle inequality) $\|x + y\| \leq \|x\| + \|y\|$, for all $x, y \in X$.

Definition 1.14 (Normed vector space)**.** A *normed vector space* $(X, \|\cdot\|)$ is a vector space X equipped with a norm $\|\cdot\|$.

Some examples of normed vector spaces are:

- $\mathbb{R}^k$ with the norm $\|x\|_1 = \sum_{i=1}^{k} |x_i|$ for all $x \in \mathbb{R}^k$;
- $\mathbb{R}^k$ with the Euclidean norm, defined for all $x \in \mathbb{R}^k$ by

$$\|x\|_2 = \sqrt{\sum_{i=1}^{k} x_i^2};$$

- $\mathbb{R}^k$ with the norm $\|x\|_\infty = \max_{i=1,\cdots,k} |x_i|$.

A norm $\|\cdot\|$ on a vector space X induces naturally a metric d defined by

$$d(x,y) = \|x - y\|,$$

for all $x, y \in X$.

Definition 1.15 (Operator norm)**.** Let $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ two normed vector spaces. Let $T\colon X \to Y$ a linear operator. We define

$$\|T\| := \sup_{\|x\| \leq 1} \|Tx\|_Y. \tag{1.1}$$

If $\|T\| < \infty$ we say that T is a *bounded* operator and we call (1.1) the *(operator) norm* of T.

It can be easily proved that $\|\cdot\|$ is a norm.

Notation 1.5. Let X, Y be normed vector spaces. We denote by $\mathcal{L}(X, Y)$ the following set:

$$\mathcal{L}(X, Y) := \{T \colon X \to Y \ : \ T \text{ linear and bounded}\}.$$

With the natural operations of sum between functions and multiplication of a function by a scalar, $\mathcal{L}(X, Y)$ is a vector space.
We observe that from (1.1) we have the following inequality:

$$\|Tx\|_Y \leq \|T\| \, \|x\|_X,$$

for all $x \in X$.

Proposition 1.6. Let X, Y be normed vector spaces and $T \colon X \to Y$ a linear operator. TFAE:

(i) T is bounded;

(ii) T is continuous, that is

$$\forall \epsilon > 0 \quad \exists \delta > 0 : \quad \forall x \in X \text{ such that } \|x\|_X \leq \delta \implies \|Tx\|_Y \leq \epsilon;$$

(iii) T is continuous at 0.

We would like to introduce *matricial norm*, in particular Frobenius norm.

Definition 1.16 (Matricial norm)**.** A *matricial norm* is a function

$$\| \cdot \| \colon \mathbb{R}^{n \times n} \to \mathbb{R}$$

that satisfies the following properties:

(i) $\|A\| \geq 0 \quad \forall A \in \mathbb{R}^{n \times n}$ and $\|A\| = 0 \Leftrightarrow A = \mathbb{O}$;

(ii) $\|\lambda A\| = |\lambda| \, \|A\| \quad \forall A \in \mathbb{R}^{n \times n}$;

(iii) $\|A + B\| \leq \|A\| + \|B\| \quad \forall A, B \in \mathbb{R}^{n \times n}$;

(iv) (submultiplicative property) $\|A\,B\| \leq \|A\| \, \|B\| \quad \forall A, B \in \mathbb{R}^{n \times n}$.

In the definition above we could substitute $\mathbb{R}^{n \times n}$ with $\mathbb{C}^{n \times n}$.
Here some examples of matricial norms:

- $\|A\|_1 := \max_{j=1,\cdots,n} \sum_{i=1}^n |a_{ij}|$;
- $\|A\|_\infty := \max_{i=1,\cdots,n} \sum_{j=1}^n |a_{ij}|$;
- $\|A\|_2 := \sqrt{\rho(A^T A)}$, where $\rho(B)$ is the *spectral radius* of the matrix B, that is the largest absolute value of its eigenvalues.

Frobenius norm is another example of matricial norm.

Definition 1.17 (Frobenius norm)**.** The *Frobenius norm* is a matricial norm defined by

$$\|A\|_F := \left(\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2\right)^{\frac{1}{2}} = \left[trace(A^TA)\right]^{\frac{1}{2}},$$

for all $(a_{ij})_{i,j=1,\cdots,n} = A \in \mathbb{R}^{n\times n}$.
Frobenius norm is defined also for complex matrices: in this case, $|\cdot|$ must be intended as the complex modulus and A^T must be substituted with the Hermitian transpose of A, A^H.

A variant of Frobenius norm is the *weighted Frobenius norm.* We first recall the following:

Definition 1.18 (Square root of a matrix)**.** Let $W \in \mathbb{R}^{n\times n}$ a symmetric matrix with nonnegative eigenvalues. A *square root* of W is a matrix B such that $B\cdot B = B^2 = W$. We denote B by $W^{1/2}$.

Such matrices exist because a symmetric matrix with nonnegative eigenvalues is diagonalizable, that is there exists an invertible matrix P and a diagonal matrix $D = diag(\lambda_1,\cdots,\lambda_n)$ such that $W = U^{-1}DU$. Observe that

$$\begin{aligned}W &= U^{-1}DU\\ &= U^{-1}D^{1/2}D^{1/2}U\\ &= U^{-1}D^{1/2}UU^{-1}D^{1/2}U\\ &= \left(U^{-1}D^{1/2}U\right)\left(U^{-1}D^{1/2}U\right),\end{aligned}$$

where $D^{1/2} = diag(\mu_1,\cdots,\mu_n)$ and $\mu_i = \sqrt{\lambda_i}$.
Hence, a square root of W is $W^{1/2} = \left(U^{-1}D^{1/2}U\right)$.

Definition 1.19 (Weighted Frobenius norm)**.** Let $\|\cdot\|_F$ be Frobenius norm and $W \in \mathbb{R}^{n\times n}$ a symmetric matrix with nonnegative eigenvalues. The *weighted Frobenius norm* is defined for all $A \in \mathbb{R}^{n\times n}$ by

$$\|A\|_W := \|W^{1/2}AW^{1/2}\|_F,$$

where $W^{1/2}$ is taken as explained above. The matrix W is call the *weight* of Frobenius norm.

We can associate a vector norm to a matricial norm as follows:

Definition 1.20 (Induced matricial norm)**.** Let X be a vector space over $\mathbb{R}$. Suppose that $\|\cdot\|_X$ is a norm on X. The norm defined for all $A \in \mathbb{R}^{n\times n}$ by

$$\|A\| := \sup_{x\neq 0}\frac{\|Ax\|_X}{\|x\|_X} = \max_{\|x\|_X=1}\|Ax\|_X$$

is called *induced or operator matricial norm.*

In particular, induced matricial norms satisfy the properties:

- (i)-(iv) of the previous definition;
- $\|Ax\|_X \leq \|A\| \, \|x\|_X$ for all $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$;
- $\|I\| = 1$, where I is the identity matrix.

The matricial norms $\|\cdot\|_1$, $\|\cdot\|_\infty$ and $\|\cdot\|_2$ illustrated previously are examples of induced matricial norms, whereas Frobenius norm in not induced (for example, it does not satisfy the last property: $\|I_n\|_F = \sqrt{n} \neq 1$).

1.3 Kullback-Leibler Divergence

To measure the difference between two probability distributions over the same variable x, a "measure" called the *Kullback-Leibler divergence* (or *relative entropy* or *Kullback-Leibler index*) has been popularly used in the data mining literature. The concept originated in statistical mechanics, probability theory and information theory.

The Kullback-Leibler divergence is a "non-symmetric measure" of the difference between two probability distributions $p(x)$ and $q(x)$. Specifically, the Kullback-Leibler divergence of $q(x)$ from $p(x)$ is a measure of the information lost when $q(x)$ is used to approximate $p(x)$.

Definition 1.21 (Kullback-Leibler divergence, discrete case)**.** Let $p(x)$ and $q(x)$ be two probability distributions on a discrete set X. The Kullback-Leibler divergence of q from p is defined by

$$\mathbb{D}(p\|q) = \sum_{x\in X} p(x) \ln \frac{p(x)}{q(x)},$$

if *supp*$(p) \subset$ *supp*(q), $+\infty$ otherwise (here, by definition, $0 \cdot \log 0 = 0$).

The Kullback-Leibler divergence measures the expected number of extra bits required to code samples from $p(x)$ when using a code based on $q(x)$, rather than using a code based on $p(x)$. Typically $p(x)$ represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution. The measure $q(x)$ typically represents a theory, model, description, or approximation of $p(x)$.

We now discuss the continuous version of the Kullback-Leibler divergence.

Definition 1.22 (Kullback-Leibler Divergence, continuous case)**.** Let p, q be two probability distributions of a continuous random variable $x \in \mathbb{R}^m$. The Kullback-Leibler divergence of q from p is the quantity

$$\mathbb{D}(p\|q) = \int_{\mathbb{R}^m} p(x) \ln \frac{p(x)}{q(x)} dx.$$

Notice that attention should be paid in the definition above. We know that $\lim_{p\to 0} p\log p = 0$. However, when $p \neq 0$ but $q = 0$, $\mathbb{D}(p\|q)$ is defined as ∞. This means that if one event e is possible (i.e., $p(e) > 0$), and the other predicts it is absolutely impossible (i.e., $q(e) = 0$), then the two distributions are absolutely different.

Although the KL divergence measures the "distance" between two distributions, it is not a distance measure:

- it is not symmetric: in general, $\mathbb{D}(p\|q) \neq \mathbb{D}(q\|p)$;
- it does not satisfy the triangular inequality.

Nevertheless, $\mathbb{D}(p\|q) \geq 0$ and $\mathbb{D}(p\|q) = 0$ iff $p = q$, so the KL divergence is a pseudo-metric. We now prove that $\mathbb{D}(p\|q) \geq 0$.

$$\mathbb{D}(p\|q) = \int_{\mathbb{R}^m} p(x)\ln\frac{p(x)}{q(x)}dx = \int_{\mathbb{R}^m} \frac{p(x)}{q(x)}q(x)\ln\frac{p(x)}{q(x)}dx. \tag{1.2}$$

Calling $g(\zeta) := \zeta\log\zeta$, (1.2) becomes

$$\int_{\mathbb{R}^m} q(x)g\left(\frac{p(x)}{q(x)}\right)dx.$$

We observe that $g'(\zeta) = \log\zeta + 1$ and $g''(\zeta) = \frac{1}{\zeta}$, which is strictly positive for all $\zeta > 0$ so g is stricty convex for all $\zeta > 0$. By Jensen inequality, we have

$$\int_{\mathbb{R}^m} q(x)g\left(\frac{p(x)}{q(x)}\right)dx \geq g\left(\int_{\mathbb{R}^m}\frac{p(x)}{q(x)}q(x)dx\right) = g\left(\int_{\mathbb{R}^m} p(x)dx\right) = g(1) = 0.$$

Therefore, we proved that $\mathbb{D}(p\|q) \geq 0$ for all p, q distributions on $\mathbb{R}^m$.

We would like to derive a specific formula for the KL divergence when p, q are zero-mean multivariate Gaussian distributions. We first recall the definition of a multivariate Gaussian distribution.

Definition 1.23 (Multivariate Gaussian distribution)**.** The multivariate Gaussian distribution is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. The multivariate Gaussian distribution is said to be "non-degenerate" when the symmetric covariance matrix Σ is positive definite. In this case the distribution has density

$$f(x) = \frac{\exp\left(-\frac{1}{2}\left(x-\mu\right)^T\Sigma^{-1}(x-\mu)\right)}{\sqrt{(2\pi)^k\det(\Sigma)}},$$

where x is a real m-dimensional column vector and μ is the mean ($\mu \in \mathbb{R}^m, \Sigma \in \mathbb{R}^m \times \mathbb{R}^m$).

Let p, q be two zero-mean multivariate Gaussian distributions with nonsingular $n \times n$ covariance matrices P, Q, respectively. The KL divergence can be derived in closed form

$$\mathbb{D}(p\|q) = \frac{1}{2}\left[\log\det\left(P^{-1}Q\right) + tr(Q^{-1}P) - n\right].$$

Since P^{-1} and Q^{-1} are the natural parameters of the Gaussian distributions, we write

$$\mathbb{D}(P^{-1}\|Q^{-1}) = \frac{1}{2}\left[\log\det\left(P^{-1}Q\right) + tr(Q^{-1}P) - n\right].$$

Note that in the right side of the last equality there is the term $\log\det\left(P^{-1}Q\right)$. It is useful for this work to denote by $J(\cdot)$ the following map defined on nonsingular $n \times n$ matrices M:

$$J(M) = \log|\det(M)|. \tag{1.3}$$

Let $\delta J(M; \delta M)$ denote the directional derivative of J in direction $\delta M \in \mathbb{R}^{n\times n}$, namely

$$\delta J(M; \delta M) = \lim_{\epsilon\to 0} \frac{J(M + \epsilon\delta M) - J(M)}{\epsilon}.$$

Then we have the following result [5, Lemma 2].

Lemma 1.7. If M is nonsingular then, for any $\delta M \in \mathbb{R}^{n\times n}$,

$$\delta J(M; \delta M) = trace[M^{-1}\delta M]. \tag{1.4}$$

Proof. We have

$$\begin{aligned}
\delta J(M; \delta M) &= \lim_{\epsilon\to 0} \frac{\log|\det(M + \epsilon\delta M)| - \log|\det(M)|}{\epsilon} \\
&= \lim_{\epsilon\to 0} \frac{\log|\det[(M + \epsilon\delta M)M^{-1}]|}{\epsilon} \\
&= \lim_{\epsilon\to 0} \frac{\log|\det(I + \epsilon M^{-1}\delta M)|}{\epsilon} \\
&= \lim_{\epsilon\to 0} \frac{\log|\prod_i(1 + \epsilon\lambda_i)|}{\epsilon} \\
&= \lim_{\epsilon\to 0} \frac{\sum_i \log|1 + \epsilon\lambda_i|}{\epsilon} \\
&= \sum_i \lambda_i = trace[M^{-1}\delta M],
\end{aligned}$$

where the λ_i's are the eigenvalues (counted with multiplicity) of $M^{-1}\delta M$. □

Chapter 2

Optimization

Outside the realm of pure mathematics, most practicing scientists and engineers are not concerned with finding exact answers to problems. Indeed, living a finite universe, we have no way of exactly measuring physical quantities and even if we did, the exact answer would not be of much use.

In mathematics, there are many problems and equations (algebraic, differential and partial differential) whose exact solutions are known to exist but are difficult, very time consuming or impossible to solve exactly. But for many practical purposes, an estimate to the exact answer will do just fine, provided that we have a guarantee that the error is not too large. In this sense, we talk about *numerical analysis.* In particular, when the problem is that of minimizing or maximizing a function, possibly subject to constraints on its variables, we speak of *optimization.*

The most important distinction of optimization problems is between problems that have constraints on the variables and those that do not. According to this classification, we have:

- *unconstrained optimization*;
- *constrained optimization.*

In this Chapter, we give the mathematical background of both unconstrained and constrained optimization. After some definitions, we state two of the main results in Calculus, that is Taylor's theorem and Lagrange multiplier theorem. Moreover, we recall the concept of convexity, concerning both sets and functions.

Troughout this Chapter, we use $\|\cdot\|$ to denote the Euclidean norm on $\mathbb{R}^n$.

2.1 Minimum of a function

Definition 2.1 (Local minimum of a function)**.** Let A be an open subset of $\mathbb{R}^n$. We say that the function $f\colon A \to \mathbb{R}$, $f \in \mathcal{C}^2$ (that is, f is twice continuously differentiable in A), has a local minimum at $x_0 \in A$ if

$$\exists r > 0 : \quad \forall x \in B_r(x_0) \cap A \quad f(x) \geq f(x_0),$$

where $B_r(x_0) = \{x \in \mathbb{R}^n : \|x - x_0\| < r\}$.
Moreover, we say that f has a strict local minimum at $x_0 \in A$ if

$$\exists r > 0 : \quad \forall x \in B_r(x_0) \cap A \setminus \{x_0\} \quad f(x) > f(x_0).$$

Definition 2.2 (Global minimum of a function)**.** Let A be an open subset of $\mathbb{R}^n$. We say that the function $f\colon A \to \mathbb{R}$, $f \in \mathcal{C}^2$, has a global minimum at $x_0 \in A$ if

$$f(x) \geq f(x_0), \quad \forall x \in A.$$

Definition 2.3 (Critical point)**.** Let A be an open subset of $\mathbb{R}^n$ and $f\colon A \to \mathbb{R}$ a differentiable function. We call $x_0 \in A$ a critical point of f if $\nabla f(x_0) = 0$.

The mathematical tool used to study minimizers of smooth functions -by which we generally mean functions whose second derivatives exist and are continuous- is Taylor's theorem. Since this theorem is central troughout this work, we state it now.

Theorem 2.1 (Taylor's theorem)**.** *Suppose that $f\colon \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $p \in \mathbb{R}^n$. Then we have that*

$$f(x + p) = f(x) + \nabla f(x + tp)^T p, \tag{2.1}$$

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

$$\nabla f(x + p) = \nabla f(x) + \int_0^1 \nabla^2 f(x + tp)p \, dt, \tag{2.2}$$

and that

$$f(x + p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x + tp)p, \tag{2.3}$$

for some $t \in (0, 1)$. Here, $\nabla^2 f(x + tp)$ denotes the Hessian matrix of f at $x + tp$.

Now we give necessary conditions for x_0 to be a local minimum for f.

Theorem 2.2. *Let A be an open subset of $\mathbb{R}^n$, $f\colon A \to \mathbb{R}$ a function, $f \in \mathcal{C}^2$ and $xo \in A$ a local minimum for f. Then x_0 is a critical point of f and $\nabla^2 f(x_0) \geq 0$.*

The following theorem gives sufficient conditions for the same problem.

Theorem 2.3. *Let A be an open subset of $\mathbb{R}^n$ and $f\colon A \to \mathbb{R}$ a function, $f \in \mathcal{C}^2$. Suppose that $x_0 \in A$ is a critical point of f and $\nabla^2 f(x_0) > 0$. Then x_0 is a strict local minimum for f.*

Finally, we briefly talk about constrained optimization, recalling the Lagrange multiplier theorem.

Suppose we have two functions:

- $f\colon \mathbb{R}^n \to \mathbb{R}$;
- $g\colon \mathbb{R}^n \to \mathbb{R}^m$.

We would like to find a (global/local) minimum x^* of f subject to $g = 0$, that is to solve the following problem:

$$\min_x \{f(x) : g(x) = 0\}. \tag{2.4}$$

Let $Jg(x)$ denote the Jacobian matrix of g at a certain point $x \in \mathbb{R}^n$. Notice that $Jg(x)$ is a $m \times n$ matrix. We are now ready to state the following theorem.

Theorem 2.4 (Lagrange multiplier theorem)**.** *Let f, g be two functions as above, x^* a local minimizer for problem (2.4) and f, g continuously differentiable at x^*. If $Jg(x^*)$ has full row rank, then there exists a (unique) $\lambda \in \mathbb{R}^m$, $\lambda = (\lambda_1, ..., \lambda_m)$, satysfying*

$$\nabla f(x^*) \;=\; \lambda^T Jg(x^*).$$

The numbers $\lambda_1, ..., \lambda_m$ are called *Lagrange multiplier.*
Let $\mathcal{L}\colon \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ be the function defined, for all $(\lambda, x) \in \mathbb{R}^m \times \mathbb{R}^n$, by:

$$\mathcal{L}(\lambda, x) \;=\; f(x) + \lambda^T g(x).$$

This function is called *Lagrange function*, or *Lagrangian.*

From the previous theorem, we have the following:

Theorem 2.5. *Let f, g be two functions as above, x^* a local minimizer for problem (2.4) and f, g continuously differentiable at x^*. If $Jg(x^*)$ has full row rank, then there exists a (unique) $\lambda \in \mathbb{R}^m$, $\lambda = (\lambda_1, ..., \lambda_m)$, such that*

$$\frac{\partial \mathcal{L}}{\lambda_i}(\lambda, x^*) = 0,\; i = 1, \cdots, m \tag{2.5}$$

and

$$\frac{\partial \mathcal{L}}{x_j}(\lambda, x^*) = 0,\; j = 1, \cdots, n. \tag{2.6}$$

Observe that the conditions (2.5)-(2.6) are equivalent to require (λ, x^*) to be a critical point of the Lagrangian.

2.2 Convex sets and functions

We now recall the definitions and the main properties of convex sets and functions.

Definition 2.4 (Convex set)**.** A set $D \in \mathbb{R}^n$ is *convex* if

$$\forall x, y \in D \implies \lambda x + (1-\lambda)y \in D \quad \forall \lambda \in [0,1].$$

Definition 2.5 (Convex function)**.** Let $f\colon D \subset \mathbb{R}^n \to \mathbb{R}$ a function, where D is a convex set. f is *convex* if

$$\forall x, y \in D, \forall \lambda \in [0,1] \implies f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$$

Moreover, f is *strictly* convex when the previous inequality is strict for all $\lambda \in (0,1)$.

The following theorems show the connection between convexity and differentiability.

Theorem 2.6. *Let $D \subset \mathbb{R}^n$ be an open convex set and $f\colon D \to \mathbb{R}$ a function, $f \in \mathcal{C}^1(D)$. TFAE:*

(i) *f is convex;*

(ii) $\forall x, y \in D \implies f(x) \geq f(y) + \nabla f(y)^T (x-y)$*;*

(iii) $\forall x, y \in D \implies (\nabla f(x) - \nabla f(y))^T (x-y) \geq 0$.

Corollary 2.7. *Let $D \subset \mathbb{R}^n$ be an open convex set and $f\colon D \to \mathbb{R}$ a convex function, $f \in \mathcal{C}^1(D)$. If $x_0 \in D$ is a critical point of f, then x_0 is a global minimum of the function.*

Corollary 2.8. *Let $D \subset \mathbb{R}^n$ be an open convex set and $f\colon D \to \mathbb{R}$ a strictly convex function, $f \in \mathcal{C}^1(D)$. If $x_0 \in D$ is a minimum of f, then x_0 is the only minimum of the function.*

Theorem 2.9. *Let $D \subset \mathbb{R}^n$ be an open convex set and $f\colon D \to \mathbb{R}$ a function, $f \in \mathcal{C}^1(D)$. TFAE:*

(i) *f is convex;*

(ii) $\nabla^2 f(x) \geq 0 \quad \forall x \in D$.

Under the hypoteses of the previous theorem, if $\nabla^2 f(x) > 0 \quad \forall x \in D$, then f is strictly convex. In general, it is not true that the strictly convexity of f implies $\nabla^2 f(x) > 0 \quad \forall x \in D$ (e.g. $f(x) = x^4$).

Definition 2.6 (Strongly convex function)**.** Let $f\colon D \subset \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function, where D is a convex set. f is *strongly convex* if

$$\exists\, \alpha > 0 \text{ such that } \quad \nabla^2 f(x) \geq \alpha I_n, \quad \forall x \in D,$$

namely the Hessian is "bounded away" from 0. Here I_n denotes the identity matrix in $\mathbb{R}^n$.

Obviously, strongly convexity implies strictly convexity. In general, the vice versa does not hold.
Observe that if f is a strongly convex function, its Hessian is a positive definite matrix; moreover, if f is a $\mathcal{C}^2$ strongly convex function, its Hessian is a symmetric positive definite matrix.

Chapter 3

Line search methods

Suppose we have a function f to minimize. We would like to find a/the minimum (or its approximation) of f using a numerical method. We have already seen that the problem of minimizing a function belongs to the so-called (unconstrained) minimization or, more generally, (unconstrained) optimization; we use the terms optimization methods and optimization algorithms to indicate numerical methods and algorithms used for this kind of problems.

Now we introduce the notations that are used in this Chapter:

- $x \in \mathbb{R}^n$ is the vector of variables;
- $f\colon \mathbb{R}^n \to \mathbb{R}$ is the objective function, a function of x that we want to minimize;
- x_0 is the starting point of the algorithm;
- $\{x_k\}_k$ is the sequence of iterates generated by the algorithm;
- $f_k = f(x_k)$ is the value of f at the point x_k;
- $\nabla f_k = \nabla f(x_k)$ is the value of the gradient of f at the point x_k;
- $\nabla^2 f_k = \nabla^2 f(x_k)$ is the value of the Hessian of f at the point x_k;
- $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$.

All algorithms for unconstrained minimization require a starting point x_0. The user with knowledge about the application and the data set may be in a good position to choose x_0 to be a reasonable estimate of the solution. Otherwise, the starting point must be chosen in some arbitrary manner.
Beginning at x_0, optimization algorithms generate a sequence of iterates $\{x_k\}_{k\in\mathbb{N}}$ that terminate when either no more progress can be made or when it seems that a solution point has been approximated with sufficient accuracy. The strategy used to move from one iterate to the next distinguishes one algoritm from another. Most strategies make use of the objective function

f and possibly its first and second derivatives (or their approximations). Some algorithms accumulate information gathered at previous iterations, while others use only local information from the current point. In order to find a minimum of f, all algorithms should find a new iterate x_{k+1} with a lower function value than x_k.
All good algorithms should possess the following properties:

- robustness: they should perform well on a wide variety of problems in their class, for all reasonable choices of the initial variable;
- efficiency: they shoul not require too much computer time or storage;
- accuracy: they should be able to identify a solution with precision, without being overly sensitive to errors in the data or the arithmetic rounding errors that occur when the algorithm is implemented on a computer.

There are two fundamental strategies for moving from the current point x_k to a new iterate x_{k+1}: line search strategies and trust-region methods. We shall discuss only the first ones.

In the line search strategy, the algorithm chooses a direction p_k and searches along this direction from the current iterate x_k for a new iterate with a lower function value. The distance to move along p_k can be found by approximately solving the following one-dimensional minimization problem to find a step length α:

$$\min_{\alpha>0} f(x_k + \alpha p_k). \tag{3.1}$$

By solving (3.1) exactly, we would derive the maximum benefit from the direction p_k, but an exact minimization is expensive and unnecessary. Instead, the line search algorithm generates a limited number of trial step lengths until it finds one that loosely approximates the minimum of (3.1). At the new point a new search direction and step length are computed, and the process is repeated.

3.1 Search directions

In this section, we shall discuss three types of search directions:

- the steepest descent direction;
- the Newton direction;
- the quasi-Newton direction.

Definition 3.1 (Steepest descent direction)**.** If f is our objective function, the *stepest descent direction* is the quantity $-\nabla f_k = -\nabla f(x_k)$.

The following proposition shows that $-\nabla f_k$ is the most obvious choice for search direction for a line search method.

Proposition 3.1. Among all the directions we could move from x_k, $-\nabla f_k$ is the one along which f dicreases most rapidly.

Proof. By theorem 2.1 (Taylor's theorem), for any search direction p and step length parameter α, we have

$$f(x_k + \alpha p) = f(x_k) + \alpha p^T \nabla f_k + \frac{1}{2}\alpha^2 p^T \nabla^2 f(x_k + tp), \quad \text{for some t} \in (0, \alpha)$$

(see (2.3)). The rate of change in f along the direction p at x_k is simply the coefficient of α, namely, $p^T \nabla f_k$. Hence, the unit direction p of most rapid decrease is the solution to the problem

$$\min_{\{p:\|p\|=1\}} p^T \nabla f_k. \tag{3.2}$$

Since $p^T \nabla f_k = \|p\| \|\nabla f_k\| \cos\theta$, where θ is the angle between p and ∇f_k, we have from $\|p\| = 1$ that $p^T \nabla f_k = \|\nabla f_k\| \cos\theta$, so the objective in (3.2) is minimized when $\cos\theta$ takes on its minimum value of -1 at $\theta = \pi$ radiants. In other words, the solution to (3.2) is

$$p = -\frac{\nabla f_k}{\|\nabla f_k\|},$$

as claimed. □

Definition 3.2 (Steepest descent method)**.** The *steepest descent method* is a line search method that moves along $p_k = -\nabla f_k$ at every step.

It can choose the step length a_k in a variety of ways. One advantage of the steepest descent direction is that it requires calculation of the gradient ∇f_k but not of the second derivatives. However, it can be excruciatingly slow on difficult problems.

Line search methods may use search directions other than the steepest descent direction.

Definition 3.3 (Discent direction)**.** A *descent* direction is a direction that makes an angle of strictly less than $\pi/2$ radians with $-\nabla f_k$.

Proposition 3.2. In general, any descent direction is guaranteed to produce a decrease in f, provided that the step length is sufficiently small.

Proof. By (2.3) from Taylor's theorem 2.1, we have that

$$f(x_k + \epsilon p_k) = f(x_k) + \epsilon p_k^T \nabla f_k + O(\epsilon^2).$$

When p_k is a downhill direction, the angle θ_k between p_k and ∇f_k has $\cos\theta_k < 0$, so that

$$p^T \nabla f_k = \|p\| \, \|\nabla f_k\| \cos\theta < 0.$$

It follows that $f(x_k + \epsilon p_k) < f(x_k)$ for all positive but sufficiently small values of ϵ.

□

Another important search direction is the *Newton direction.* This direction is derived from the second-order Taylor series approximation to $f(x_k + p)$, which is

$$f(x_k + p) \approx f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p \overset{def}{=} m_k(p). \tag{3.3}$$

Assuming for the moment that $\nabla^2 f_k$ is positive definite, we obtain the Newton direction by finding the vector p that minimizes $m_k(p)$. By simply setting the derivative of $m_k(p)$ to zero, we obtain $p_k = -\nabla^2 f_k^{-1} \nabla f_k$.

Definition 3.4 (Newton direction)**.** The *Newton direction* is the vector

$$p_k^N = -\left(\nabla^2 f_k\right)^{-1} \nabla f_k. \tag{3.4}$$

The Newton direction is reliable when the difference between the true function $f(x_k + p)$ and its quadratic model $m_k(p)$ is not too large. By comparing (3.3) with (2.3), we see that the only difference between these functions is that the matrix $\nabla^2 f(x_k + tp)$ in the third term of the expansion has been replaced by $\nabla^2 f_k = \nabla^2 f(x_k)$. If $\nabla^2 f(\cdot)$ is sufficiently smooth, this difference introduces a perturbation of only $O(\|p\|^3)$ into the expansion, so that the approximation $f(x_k + p) \approx m_k(p)$ is very accurate indeed.

Proposition 3.3. If $\nabla^2 f_k$ is positive definite and $\nabla f_k \neq 0$, the Newton direction is a descent direction.

Proof. We have

$$\nabla f_k^T p_k^N = -{p_k^N}^T \nabla^2 f_k p_k^N \leq -\sigma_k \|p_k^N\|^2$$

for some $\sigma_k > 0$ (the last inequality follows from the theorem 1.4). Since the gradient ∇f_k (and therefore the step p_k^N) is zero, we have that $\nabla f_k^T p_k^N < 0$, so that the Newton direction is a descent direction. □

Unlike the steepest descent direction, there is a "natural" step length of 1 associated with the Newton direction. Most line search implementations of Newton's method use the unit step $\alpha = 1$ where possible and adjust this step length only when it does not produce a satisfactory reduction in the value of f.

Methods that use the Newton direction have a fast rate of local convergence, typically quadratic. When a neighborhood of the solution is reached, convergence to high accuracy often occurs in just a few iterations. The main drawback of the Newton direction is the need for the Hessian $\nabla^2 f(x)$. Explicit computation of this matrix, its inversion and storage are sometimes, though not always, a cumbersome, error-prone, and expensive process. In some large dimensional problems, this becomes simply unfeasible.

Quasi-Newton search directions provide an attractive alternative in that they do not require computation of the Hessian and yet still attain a superlinear rate of convergence. In place of the true Hessian $\nabla^2 f_k$, they use an approximation B_k, which is updated after each step to take account of the additional knowledge gained during the step. The updates make use of the fact that changes in the gradient provide information about the second derivative of f along the search direction. In fact, it might be possible, in some quasi-Newton iterations, to update directly B_k^{-1}.
By using the expression (2.2) from our statement of Taylor's theorem

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

we have, by adding and substracting the term $\nabla^2 f(x)p$, that

$$\nabla f(x+p) = \nabla f(x) + \nabla^2 f(x)p + \int_0^1 [\nabla^2 f(x+tp) - \nabla^2 f(x)]\, p\, dt.$$

Since $\nabla^2 f$ is continuous, the size of the final integral term is $o(\|p\|)$. By setting $x = x_k$ and $p = x_{k+1} - x_k$, we obtain

$$\nabla f_{k+1} = \nabla f_k + \nabla^2 f_{k+1}(x_{k+1} - x_k) + o(\|x_{k+1} - x_k\|).$$

When x_k and x_{k+1} lie in a region near the solution x^*, within which $\nabla^2 f$ is positive definite, the final term in this expansion is eventually dominated by the $\nabla^2 f_{k+1}(x_{k+1} - x_k)$ term, and we can write

$$\nabla^2 f_{k+1}(x_{k+1} - x_k) \approx \nabla f_{k+1} - \nabla f_k. \tag{3.5}$$

We choose the new Hessian approximation B_{k+1} so that it mimics this property (3.5) of the true Hessian, that is, we require it to satify the following condition, known as the *secant equation.*

Definition 3.5 (Secant equation)**.** The *secant condition* is defined by

$$B_{k+1}s_k = y_k, \tag{3.6}$$

where

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f_{k+1} - \nabla f_k.$$

Typically we impose additional requirements on B_{k+1}, such as:

- simmetry, motivated by simmetry of the exact Hessian;
- the difference between successive approximation B_k to B_{k+1} must have low rank.

The initial approximation B_0 must be chosen by the user. Two of the most popular formulas for updating the Hessian approximation B_k are the *symmetric-rank-one* (SR1) formula and the *BFGS formula* (we shall discuss the BFGS formula in Chapter 4).

The quasi-Newton search direction is given by using B_k in place of the exact Hessian in the formula (3.4), as illustrated in the following definition.

Definition 3.6 (Quasi-newton direction)**.** The *quasi-newton direction* is given by

$$p_k^Q = -B_k^{-1} \nabla f_k. \tag{3.7}$$

Some practical implementations of quasi-Newton methods avoid the need to factorize B_k at each iteration by updating the *inverse* of B_k, instead of B_k itself. Calculation of p_k can then be performed by using the formula $p_k = -D_k \nabla f_k$, where $D_k \overset{def}{=} B_k^{-1}$ is the inverse approximation. This can be implemented as a matrix-vector multiplication, which is typically simpler than the factorization/back-substitution procedure that is needed to implement the formula (3.7).

3.2 Scaling

The performance of an algorithm may depend crucially on how the problem is formulated. One important issue in problem formulation is *scaling*. In unconstrained optimization, a problem is said to be *poorly scaled* if changes to x in a certain direction produce much larger variations in the value of f than do changes to x in another direction.

Scaling is performed (sometimes unintentionally) when the units used to represent variables are changed.

Some optimization algorithms, such as steepest descent, are sensitive to poor scaling, while others, such as Newton's method, are unaffected by it.

Algorithms that are not sensitive to scaling are preferable to those that are sensitive, because they can handle poor problem formulations in a more rubust manner. In designing complete algorithms, we try to incorporate *scale invariance* into all aspects of the algorithm, including the line search or trust-region strategies and convergence tests. Generally speaking, it is easier to preserve scale invariance for line search algorithm than for trust-region algorithms.

3.3 Rates of convergence

One of the key measures of performance of an algorithm is its rate of convergence. We now define the terminolgy used in the following pages.

Definition 3.7 (Q-linear convergence)**.** Let $\{x_k\}$ be a sequence in $\mathbb{R}^n$ that converges to x^*. We say that the convergence is *Q-linear* if there is a constant $r \in (0, 1)$ such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq r, \quad \text{for all } k \text{ sufficiently large.} \tag{3.8}$$

Definition 3.8 (Q-superlinear convergence)**.** The convergence is said to be *Q-superlinear* if

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

Definition 3.9 (Q-quadratic convergence)**.** *Q-quadratic* convergence, an even more rapid convergence rate, is obtained if

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} \leq M, \quad \text{for all } k \text{ suffciently large,}$$

where M is a positive constant, not necessarily less than 1.

The speed of convergence depends on r and (more weakly) on M, whose values depend not only on the algorithm but also on the properties of the particular problem. Regardless of these values, however, a quadratically convergent sequence shall always eventually converge faster than a linearly convergent sequence.

The following implications hold:

$$\textit{quadratic} \implies \textit{superlinearly} \implies \textit{linearly},$$

that is any sequence that converges Q-quadratically also converges Q-superlinearly, and any sequence that converges Q-superlinearly also converges Q-linearly. We can also define higher rates of convergence (cubic, quartic, and so on), but these are less interesting in practical terms. In general, we talk about Q-convergence *of order p*.

Definition 3.10 (Q-order of convergence)**.** We say that the *Q-order of convergence* is p (with $p > 1$) if there is a positive constant M such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} \leq M, \quad \text{for all } k \text{ suffciently large.}$$

We show the rates of convergence of the line search methods we have discussed above:

- steepest descent algorithms converge only at a Q-linear rate (when the problem is ill-condioned the convergence constant r in (3.8) is close to 1);
- Newton's method converges Q-quadratically;
- quasi-Newton methods typically converge Q-superlinearly.

Throughout this work we shall normally omit the letter Q and simply talk about superlinear convergence, quadratic convergence, etc.

3.4 Step Length

Each iteration of a line search method computes a search direction p_k and then decides how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{3.9}$$

where the positive scalar a_k is called *step length.* The success of a line search method depends on effective choices of both the direction p_k and the step length α_k. We have already discuss the difference choices for p_k previously, now we talk about the choice of α_k.

In computing the step length α_k, we face a tradeoff. We would like to choose α_k to give a substantial reduction of f, but at the same time, we do not want to spend too much time making the choice. The ideal choice would be the global minimizer of the univariate function $\Phi(\cdot)$ defined by

$$\Phi(\alpha) = f(x_k + \alpha p_k), \quad \alpha > 0, \tag{3.10}$$

but in general, it is too expensive to identify this value. To find even a local minimizer of Φ to moderate precision generally requires too many evaluations of the objective function f and possibly the gradient ∇f. More practical strategies perform an *inexact line search* to identify a step length that achieves adequate reductions in f at minimal cost.

Typical line search algorithms try out a sequence of candidate values for α, stopping to accept one of these values when certain conditions are satisfied. There are various termination conditions for the line search algorithm such as the *Wolfe conditions*, the *strong Wolfe conditions* and the *Goldstein conditions*: we shall discuss only the first ones.

A popular inexact line search condition stipulates that a_k should first of all give sufficient decrease in the objective function f, as measured by the following inequality:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k, \tag{3.11}$$

for some constant $c_1 \in (0, 1)$. The inequality (3.11) is called the *sufficient decrease condition* and, sometimes, the *Armijo condition.*

The sufficient decrease condition is not enough by itself to ensure that the algorithm makes reasonable progress. To rule out unacceptably short steps we introduce a second requirement, called the *curvature condition*, which requires a_k to satisfy

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, \tag{3.12}$$

for some constant $c_2 \in (c_1, 1)$, where c_1 is the constant from (3.11).

The sufficient decrease condition (3.11) and the curvature condition (3.12) are known collectively as the *Wolfe conditions*. We restate them here:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k, \tag{3.13a}$$
$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k. \tag{3.13b}$$

So we can write:

$$Wolfe \quad = \quad Sufficient\, decrease + Curvature.$$

It is not difficult to prove that there exist step lengths that satisy the Wolfe conditions for every function f that is smooth and bounded below.

3.5 Convergence of line search methods

Notation 3.4 (Global convergence)**.** Here we use the term *globally convergent* to refer to algorithms for which the property

$$\lim_{k \to +\infty} \|\nabla f_k\| = 0 \tag{3.14}$$

is satisfied.

Note that this term is sometimes used in other contexts to mean different things. For line search methods of the general form (3.9), the limit (3.14) is the strongest global convergence result that can be obtained: we cannot guarantee that the method converges to a minimizer, but only that is attracted by stationary points.

To obtain global convergence, we must not only have well-chosen step lengths but also well-chosen search directions p_k. A theorem due to Zoutendijk plays a fundamental role in global convergence: under the Wolfe conditions and the *regularity* of both f and ∇f, this theorem implies that

$$\cos^2 \theta_k \|\nabla f_k\|^2 \to 0, \tag{3.15}$$

where $\cos \theta_k$ is defined by:

$$\cos \theta_k = -\frac{\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|}. \tag{3.16}$$

The limit (3.15) can be used in turn to derive global convergence results for line search algorithms. If our method ensures that $\cos\theta_k \geq \delta > 0$ for all k, it follows immediatly from (3.15) that $\lim_{k\to+\infty} \|\nabla f_k\| = 0$. In particular, the method of steepest descent (remind that $\cos\theta_k = 1$) produces a gradient sequence that converges to zero, provided that it uses a line search satisfying the Wolfe (or Goldstein) conditions. Moreover, Newton and quasi-Newton methods are globally convergent if the matrices B_k are positive definite with a uniformly bounded condition number (that is, there is a constant M such that $\|B_k\|\,\|B_k^{-1}\| \leq M$ for all k), and if the step lengths satisfy the Wolfe conditions.

Algorithmic strategies that achieve rapid convergence can sometimes conflict with the requirements of global convergence, and vice versa. For example, the steepest descent method is the quintessential globally convergent algorithm, but it is quite slow in practice. On the other hand, the pure Newton iteration converges rapidly when started close enough to a solution, but its steps may not even be descent directions away from the solution. The challenge is to design algorithms that incorporate both properties:

- good global convergence guarantees;
- a rapid rate of convergence.

Chapter 4

The BFGS method

In the previous Chapter we discussed the quasi-Newton method. Now we recall its iteration:

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f_k, \quad \alpha_k > 0,$$

with α_k chosen by a line search and B_k an approximation of $\nabla^2 f_k$, imposing the secant equation

$$y_k = B_{k+1} s_k,$$

where

$$y_k := \nabla f(x_k + s_k) - \nabla f(x_k), \quad s_k := \Delta x_k = x_{k+1} - x_k. \tag{4.1}$$

The most popular quasi-Newton algorithm is the *BFGS method*, named for its discoverers Broyden, Fletcher, Goldfarb and Shanno. In this Chapter we provide a classical derivation of this algorithm (and of its close relative, the DFP algorithm) and we describe its theoretical properties and practical implementation. Moreover, we present the *Broyden class* and discuss the convergence of the BFGS method.

4.1 Classical derivation of the BFGS method

We consider the following approximation of f:

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p, \tag{4.2}$$

where $p \in \mathbb{R}^n$ and $B_k \in \mathbb{R}^{n \times n}$ is symmetric positive definite. B_k shall be revised or updated at each iteration. Note that $m_k(0) = f_k$ and $\nabla m_k(0) = \nabla f_k$. We know that the minimizer of $m_k(\cdot)$ is

$$p_k = B_k^{-1} \nabla f_k, \tag{4.3}$$

so the new iterate is

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

where the step length α_k is chosen to satisfy the Wolfe conditions (3.13).

Instead of computing B_k afresh at every iteration, we would like to take advantage of the knowledge we have gained during the last step. In particular, we look for requirements to impose on B_{k+1}. At the new iteration, we would like to construct a model like the previous one:

$$m_{k+1}(p) = f_{k+1} + \nabla f_{k+1}^T p + \frac{1}{2} p^T B_{k+1} p.$$

One reasonable requirement is that the gradient of m_{k+1} should match the gradient of f at x_k and x_{k+1}. Since $\nabla m_{k+1}(0) = \nabla f_{k+1}$, the second of these conditions is automatically satisfied. The first condition can be written mathematically as

$$\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1} - \alpha_k B_{k+1} p_k \equiv \nabla f_k,$$

from which we obtain

$$B_{k+1} \alpha_k p_k = \nabla f_{k+1} - \nabla f_k.$$

Using the notation (4.1), the last equation becomes

$$B_{k+1} s_k = y_k, \tag{4.4}$$

which we know as the *secant equation* (see (3.6)).

The secant equation (4.4) implies that s_k and y_k satisfy the *curvature condition*

$$s_k^T y_k > 0, \tag{4.5}$$

as is easily seen by premultiplying the secant equation (4.4) by s_k^T (remind that B_{k+1} is a symmetric positive definite matrix).
When f is strongly convex, the curvature condition (4.5) shall be satisfied for any two points x_k and x_{k+1} (see theorem (2.6)). However, this condition shall not always be satisfied for nonconvex functions, but it is guaranteed to hold if we impose the Wolfe (or strong Wolfe) conditions on the line search.

Proposition 4.1. The Wolfe conditions (3.13) imply the curvature condition (4.5).

Proof. We rewrite (3.13b):

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k.$$

Multiplying this inequality by α_k and using the notation (4.1) we have that

$$\nabla f_{k+1}^T s_k \geq c_2 \nabla f_k^T s_k.$$

From this inequality, by substracting $-\nabla f_k^T s_k$ and using again the notation (4.1), we obtain

$$y_k^T s_k \geq (c_2 - 1)\alpha_k \nabla f_k^T p_k. \tag{4.6}$$

Since $c_2 < 1$ and p_k is a descent direction (i.e. $\nabla f_k^T p_k < 0$), the term on the right shall be positive, and the curvature condition (4.5) holds:

$$y_k^T s_k \geq (c_2 - 1)\alpha_k \nabla f_k^T p_k > 0.$$

□

Proposition 4.2. When the curvature condition (4.5) is satisfied, the secant equation (4.4) always has a solution B_{k+1}.

Proof. In fact, (4.4) admits an infinite number of solutions, since there are $n(n+1)/2$ degrees of freedom in a symmetric matrix, and the secant equation (4.4) represents only n conditions. The requirement of positive definiteness imposes n additional inequalities -all principal minors must be positive- but these conditions do not absorb the remaining degrees of freedom (because the curvature condition (4.5) and the secant equation (4.4) together imply that B_{k+1} is positive definite). □

To determine B_{k+1} uniquely, we impose that *among all symmetric matrices satisfying the secant equation (4.4), B_{k+1} is, in some sense, closest to the current matrix B_k.* In other words, we would like to solve the problem

$$\min_B \|B - B_k\| \tag{4.7a}$$

$$\text{subject to} \quad B = B^T, \quad Bs_k = y_k, \tag{4.7b}$$

where s_k and y_k satisfy the curvature condition (4.5) and B_k is symmetric and positive definite.
Many matricial norms can be used in (4.7a) and each norm gives rise to a different quasi-Newton method. A norm that allows easy solution of the minimization problem (4.7), and gives rise to a scale-invariant optimization method, is the weighted Frobenius norm $\|\cdot\|_W$ (see Chapter 1), where W can be chosen as any matrix satisfying the relation $Wy_k = s_k$. For concreteness, we choose the matrix W illustrated in [2, Chapter 8, section 1]. With this choice, the norm $\|\cdot\|_W$ is adimensional, which is a desiderable property, since we do not wish the solution of (4.7) to depend on the units of the problem.
With this weighting matrix and this norm, the unique solution of (4.7) is the so-called *DFP formula.*

Definition 4.1 (DFP formula)**.** The *DFP formula* is defined by the following:

$$\text{(DFP)} \quad B_{k+1} = (I - \gamma_k y_k s_k^T) B_k (I - \gamma_k s_k y_k^T) + \gamma_k y_k y_k^T, \tag{4.8}$$

with

$$\gamma_k = \frac{1}{y_k^T s_k}.$$

This formula is called the *DFP* updating formula, since it is the one originally proposed by Davidon in 1959 and subsequently studied, implemented and popularized by Fletcher and Powell.

The inverse of B_k, which we denote by

$$D_k := B_k^{-1}$$

is useful in the implementation of the method, since it allows the search direction (4.3) to be calculated by means of a simple matrix-vector multiplication. We can derive the following expression for the update of the inverse Hessian approximation D_k that corresponds to the DFP update of B_k in (4.8):

$$\text{(DFP)} \quad D_{k+1} = D_k - \frac{D_k y_k y_k^T D_k}{y_k^T D_k y_k} + \frac{s_k s_k^T}{y_k^T s_k}.$$

The DFP updating formula is quite effective, but it was soon superseded by the BFGS formula, which is presently considered to be the most effective of all quasi-Newton updating formulas. BFGS updating can be derived by making a simple change in the argument that led to (4.8).
Instead of imposing conditions on the Hessian approximation B_k, we impose similar conditions on its inverse D_k. The updated approximation D_{k+1} must be symmetric and positive definite, and must satisfy the secant equation (4.4), now written as

$$D_{k+1} y_k = s_k.$$

The condition of closeness to D_k is now specified by the following analougue of (4.8):

$$\min_D \|D - D_k\| \tag{4.9a}$$

$$\text{subject to} \quad D = D^T, \quad D y_k = s_k, . \tag{4.9b}$$

The norm is again the weighted Frobenius norm, where the weight matrix W is again any matrix satisfying $W s_k = y_k$. For concreteness, we choose W as above. The unique solution D_{k+1} to (4.9) is given by the following formula, the so-called *BFGS formula.*

Definition 4.2 (BFGS formula)**.** The *BFGS formula* is defined by:

$$\text{(BFGS)} \quad D_{k+1} = (I - \rho_k s_k y_k^T) D_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \tag{4.10}$$

where

$$\rho_k = \frac{1}{y_k^T s_k}. \tag{4.11}$$

There is only one last issue to be solved: the choice of the initial approximation D_0. Unfortunately, there is no magic formula that works well in all cases. We can use specific information about the problem, for instance by setting it to the inverse of an approximate Hessian calculated by finite differences at x_0. Otherwise, we can simply set it to be the identity matrix or a multiple of the identity matrix, where the multiple is chosen to reflect the scaling of the variables.

The following is an algorithm prototype of the BFGS method:

```
**procedure** BFGS($f, x_0, D_0, tolerance, \epsilon > 0$)
    $k \leftarrow 0$;
    **while** $\|\nabla f_k\| > \epsilon$ **do**
        Compute search direction $p_k = -D_k \nabla f_k$;
        Compute step length $\alpha_k$ from a line search procedure to satisfy the
        Wolfe conditions (3.13);
        $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
        $s_k \leftarrow x_{k+1} - x_k$;
        $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$;
        Compute $D_{k+1}$ by means of (4.10);
        $k \leftarrow k + 1$;
    **end (while)**
```

Algorithm 1: BFGS algorithm

Each iteration can be performed at a cost of $O(n^2)$ arithmetic operations (plus the cost of function and gradient evaluations); there are no $O(n^3)$ operations such as linear system solves or matrix-matrix operations. The algorithm is robust and its rate of convergence is superlinear, which is fast enough for most practical purposes. Even though Newton's method converges more rapidly (that is, quadratically), its cost per iteration is higher beacuse it requires the solution of a linear system. A more important advantage for BFGS is, of course, that it does not require calculation of second derivatives.

4.2 Properties and implementation

A few points in the derivation of the BFGS and DFP methods merit further discussion.

Note that the minimization problem (4.9) that gives rise to the BFGS update formula does not explicitly require the updated Hessian approximation to be positive definite. However, the following statement holds:

Proposition 4.3. If D_k is positive definite, then D_{k+1} in (4.10) is positive definite too.

Proof. First, we note from (4.6) that $y_k^T s_k > 0$, so that the updating formula (4.10)-(4.11) is well-defined. For any nonzero vector z, we have

$$z^T D_{k+1} z = w^T D_k w + \rho_k (z^T s_k)^2 \geq 0,$$

where we have defined $w = z - \rho_k y_k (s_k^T z)$. The right end side can be zero only if $s_k^T z = 0$, but in this case $w = z \neq 0$, which implies that the first term is greater than zero. Therefore, we have

$$z^T D_{k+1} z = w^T D_k w + \rho_k (z^T s_k)^2 > 0 \quad \forall z \in \mathbb{R}^n \setminus \{0\},$$

that is D_{k+1} is positive definite. □

The choice of the weighting matrix W used to define the norms in (4.7a) and (4.9b) ensures that the updating formulas are invariant to changes.

If the matrix D_k incorrectly estimates the curvature in the objective function and if this bad estimate slows down the iteration, then the Hessian approximation shall tend to correct itself within a few steps. It is also known that the DFP method is less effective in correcting bad Hessian approximations. The self-correcting properties of BFGS hold only when an adequate line search is performed: in particular, the Wolfe line search conditions (3.13) ensure that the gradients are sampled at points that allow the model (4.2) to capture appropriate curvature information.

It is interesting to note that the DFP and BFGS updating formulas are *duals* of each other, in the sense that one can be obtained from the other by the interchanges $s \leftrightarrow y, B \leftrightarrow D$. This symmetry is not surprising, given the manner in which we derived these methods above.

A few details and enhancements need to be added to the BFGS algorithm to produce an efficient implementation:

- the line search should satisfy the Wolfe conditions (or the strong Wolfe conditions);
- the line search shoul always try the step length $\alpha_k = 1$ first;
- the values $c_1 = 10^{-4}$ and $c_2 = 0.9$ are commonly used in (3.11) and in (3.12);
- the initial matrix D_0 often is set to some multiple βI of the identity;
- a heuristic that is often quite effective is to scale the starting matrix after the first step has been computed but before the first BFGS update is performed: we change the provisional value $D_0 = I$ by setting

  $$D_0 \leftrightarrow \frac{y_k^T s_k}{y^T y_k} I$$

  before applying the update (4.10), (4.11) to obtain D_1.

4.3 The Broyden class

Definition 4.3 (Broyden class)**.** The *Broyden class* is a family of quasi-Newton updating formulas, specified by the following general formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \Phi_k (s_k^T B_k s_k) v_k v_k^T, \tag{4.12}$$

where Φ_k is a scalar and v_k is the vector

$$v_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}.$$

The BFGS and DFP methods are members of the Broyden class: we recover BFGS by setting $\Phi = 0$ and DFP by setting $\Phi = 1$ in (4.12). We can therefore rewrite (4.12) as a "linear combination" of these two methods, that is,

$$B_{k+1} = (1 - \Phi_k) B_{k+1}^{BFGS} + \Phi_k B_{k+1}^{DFP}.$$

This relationship indicates that all members of the Broyden class satisfy the following properties:

(i) the secant equation (4.4);

(ii) $\phi_k \geq 0$ and $H_k > 0 \implies H_{k+1} > 0$.

Note that (i) holds because the BFGS and DFP matrices themselves satisfy this equation. Also, since BFGS and DFP updating preserve positive definiteness of the Hessian approximations when $s_k^T y_k > 0$, the relation above implies that the same property shall hold for the Broyden family if $\phi_k \geq 0$.

4.4 Convergence

First, we study the global convergence. The main result for global convergence is the following theorem.

Theorem 4.4. *Let B_0 be any symmetric positive definite initial matrix and let x_0 be a starting point for which the following conditions are satisfied:*

(i) the objective function f is twice continuously differentiable;

(ii) the level $\Omega = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is convex and there exist positive constants m and M such that

$$m\|z\|^2 \leq z^T \nabla^2 f(x) z \leq M\|z\|^2$$

for all $z \in \mathbb{R}^n$ and $x \in \Omega$.

Then the sequence $\{x_k\}_k$ generated by the BFGS algorithm converges to the minimizer x^* of f.

Note that in the hypothesis (ii), the condition

$$m\|z\|^2 \leq z^T \nabla^2 f(x) z, \quad \forall z \in \mathbb{R}^n, \, \forall x \in \Omega$$

is equivalent to require that f is strongly convex in Ω (see Chapter 2).

The next theorem shows the superlinear convergence of the BFGS method.

Theorem 4.5. *Suppose that f is twice continuously differentiable and that the iterates generated by the Algorithm 1 (BFGS) converge to a minimizer x^* at which the following assumption holds:*
the Hessian matrix $\nabla^2 f(x)$ is Lipschitz continuous at x^*, that is,

$$\|\nabla^2 f(x) - \nabla^2 f(x^*)\| \leq L\|x - x^*\|,$$

for all x near x^*, where L is a positive constant.
Suppose also that

$$\sum_{k=1}^{\infty} \|x_k - x^*\| < \infty$$

holds. Then $\{x_k\}_k$ converges to x^* at a superlinear rate.

Chapter 5

A variational derivation of a class of BFGS-like methods

In this Chapter, we provide a maximum entropy derivation of a new family of BFGS-like methods.

Consider the quasi-Newton's iteration (see Chapter 3) and suppose f is strongly convex. We approximate the Hessian $\nabla^2 f_k$ using a symmetric, positive definite matrix that satisfies the secant equation and is closest in the Kullback-Leibler "metric" (see Chapter 1) to $\nabla^2 f_k$. This choice let us to consider a new variational problem. In [3], Fletcher indeed showed that the solution to this problem is provided by the BFGS iterate thereby providing a variational characterization for it alternative to Goldfarb's classical one [4], [2]. We take a different approach leading to a family of BFGS-like methods. This approach provides theoretical support for these methods and a new proof of Fletcher's classical derivation. Moreover, we shall see that some changes to our BFGS-like methods yields the standard BFGS iteration.

Note that Fletcher's results are extremely surprising: a priori, there is no connection between the variational problem and the standard BFGS methods.

5.1 Introduction

Suppose $f\colon \mathbb{R}^n \to \mathbb{R}$ is a strongly convex $\mathcal{C}^2$ function to be minimized. We recall the quasi-Newton iteration (see Chapter 3):

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f_k, \quad \alpha_k > 0,$$

with α_k chosen by a line search and B_k an approximation of $\nabla^2 f_k$, imposing the secant equation

$$y_k = B_{k+1} s_k, \tag{5.1}$$

where

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

For $n > 1$, B_{k+1} satisfying (5.1) is underdetermined. We would like to find a symmetric and positive definite matrix B_{k+1} that is closest in some metric to the current approximation B_k: we shall use the Kullback-Leibler divergence (see Chapter 1). From now on we assume that B_k is a symmetric and positive definite matrix.

Since for a strongly convex function the Hessian $\nabla^2 f_k$ is a symmetric positive definite matrix, we can think of its approximation B_k as a covariance of a zero-mean, multivariate Gaussian distribution. Recall that in the case of two zero-mean multivariate normal distributions p, q with nonsingular $n \times n$ covariance matrices P, Q, respectively, the relative Kullback-Leibler divergence can be written as

$$\mathbb{D}(P^{-1} \| Q^{-1}) = \frac{1}{2}\left[\log\det\left(P^{-1}Q\right) + trace(Q^{-1}P) - n\right]. \tag{5.2}$$

5.2 A maximum entropy problem

Consider the following minimizing problem:

$$\min_{\{B: B^T = B, B > 0\}} \quad \mathbb{D}(B^{-1} \| B_k^{-1}), \tag{5.3a}$$

$$\text{subject to} \quad B^{-1} y_k = s_k. \tag{5.3b}$$

We decide to choose B_{k+1} as the solution to this problem.
First of all, observe that $B^{-1}y_k$ must be the given vector s_k. Thus, it seems reasonable that B_{k+1}^{-1} should approximates B_k^{-1} only in directions different from y_k. We are then led to consider the following new problem

$$\min_{\{B: B^T = B, B > 0\}} \quad \mathbb{D}(B^{-1} \| P_k^T B_k^{-1} P_k), \tag{5.4a}$$

$$\text{subject to} \quad B^{-1} y_k = s_k, \tag{5.4b}$$

where P_k is a rank $n-1$ matrix satisfying $P_k y_k = 0$, subject to the secant equation (5.3b). One possible choice for P_k is the orthogonal projection

$$P_k = I_n - \frac{y_k y_k^T}{y_k^T y_k} = I_n - \Pi_{y_k}. \tag{5.5}$$

Since $P_k B_k^{-1} P_k$ is singular, however, (5.4) does not make sense. Thus, to regularize the problem, we replace P_k with the nonsingular, positive definite matrix $P_k^\epsilon = P_k + \epsilon I_n$:

$$P_k \longleftarrow P_k^\epsilon = P_k + \epsilon I_n.$$

We observe that $P_k^T = P_k$ and $(P_k^\epsilon)^T = P_k^\epsilon$, that is P_k and P_k^ϵ are symmetric.

The objective function of the problem (5.4) is:

$$\begin{aligned}
&\mathbb{D}(B^{-1}\|P_k^T B_k^{-1} P_k) = \\
&= \frac{1}{2}\left[\log\det\left(B^{-1}(P_k^\epsilon)^{-1}B_k(P_k^\epsilon)^{-T}\right) + trace\left((P_k^\epsilon)^T B_k^{-1} P_k^\epsilon B\right) - n\right] = \\
&= \frac{1}{2}\left[\log\det\left(B^{-1}B_k\right) + \log\det\left((P_k^\epsilon)^{-2}\right) + trace\left(P_k^\epsilon B_k^{-1} P_k^\epsilon B\right) - n\right].
\end{aligned}$$

The constraint of the problem (5.4) is $B^{-1}y_k = s_k$, which is equivalent to $B_k s_k = y_k$, and also to $Bs_k - y_k = 0$.

The Lagrangian for the problem (5.4) is

$$\begin{aligned}
&\mathcal{L}(B,\lambda) = \mathbb{D}(B^{-1}\|P_k^\epsilon B_k^{-1} P_k\epsilon) + \lambda_k^T[Bs_k - y_k] = \\
&= \frac{1}{2}\left[\log\det\left(B^{-1}B_k\right) + \log\det\left((P_k^\epsilon)^{-2}\right) + tr\left(P_k^\epsilon B_k^{-1} P_k^\epsilon B\right) - n\right] + \\
&\qquad\qquad + \lambda_k^T[Bs_k - y_k].
\end{aligned}$$

First, note that the terms $\log\det\left((P_k^\epsilon)^{-2}\right)$ and n do not depend on B and therefore they play no role in the variational analysis: from now on we shall omit them. It shall be useful using the following notation:

(i) $\mathcal{L}_1(B) = \log\det\left(B^{-1}B_k\right)$;

(ii) $\mathcal{L}_2(B) = tr\left(P_k^\epsilon B_k^{-1} P_k^\epsilon B\right)$;

(iii) $\mathcal{L}_3(B,\lambda) = \lambda_k^T[Bs_k - y_k]$.

With this notation, $\mathcal{L}(B,\lambda)$ becomes

$$\mathcal{L}(B,\lambda) = \frac{1}{2}\left[\mathcal{L}_1(B) + \mathcal{L}_2(B)\right] + \mathcal{L}_3(B,\lambda).$$

Observe also that any positive definite matrix B is an interior point in the cone $\mathcal{C}$ of positive semidefinite matrices in any symmetric direction $\delta B \in \mathbb{R}^{n\times n}$. Imposing $\delta\mathcal{L}(B,\lambda;\delta B) = 0$ for all such δB, we have:

(i) $\delta\mathcal{L}_1(B;\delta B) = tr[-B^{-1}\delta B]$, by the lemma (1.7);

(ii) $\delta\mathcal{L}_2(B;\delta B) = tr\left[P_k^\epsilon B_k^{-1} P_k^\epsilon \delta B\right]$;

(iii) $\delta\mathcal{L}_3(B,\delta;\delta B) = tr\left[s_k\lambda_k^T\delta B\right]$.

Therefore, we get

$$tr\left[\left(-B^{-1} + P_k^\epsilon B_k^{-1} P_k^\epsilon + 2s_k\lambda_k^T\right)\delta B\right] = 0 \quad \forall\delta B,$$

which gives

$$-B^{-1} + P_k^\epsilon B_k^{-1} P_k^\epsilon + 2s_k\lambda_k^T = 0,$$

and finally

$$B^{-1} = P_k^\epsilon B_k^{-1} P_k^\epsilon + 2s_k\lambda_k^T.$$

Remind that we are looking for the solution B_{k+1} of the problem (5.4) so we can write

$$(B_{k+1}^\epsilon)^{-1} = P_k^\epsilon B_k^{-1} P_k^\epsilon + 2s_k\lambda_k^T.$$

As $\epsilon \searrow 0$, we get the iteration

$$B_{k+1}^{-1} = P_k B_k^{-1} P_k + 2s_k\lambda_k^T.$$

Since $P_k y_k = 0$, we have

$$B_{k+1}^{-1} y_k = P_k B_k^{-1} P_k y_k + 2s_k\lambda_k^T y_k = 2s_k\lambda_k^T y_k.$$

In order to satisfy the secant equation $B_{k+1}^{-1} y_k = s_k$, it suffices to choose the multiplier λ_k so that

$$2s_k\lambda_k^T y_k = s_k, \text{ or equivalently, } 2\lambda_k^T y_k = 1.$$

We need, however, to also guarantee symmetry and positive definiteness of the solution. We are then led to choose λ_k as

$$\lambda_k = \frac{s_k}{2y_k^T s_k}.$$

We have finally obtained the iteration:

$$B_{k+1}^{-1} = \left(I_n - \frac{y_k y_k^T}{y_k^T y_k}\right) B_k^{-1} \left(I_n - \frac{y_k y_k^T}{y_k^T y_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}. \tag{5.6}$$

Notice that, under the curvature assumption $y_k^T s_k > 0$, if $B_k > 0$, indeed B_{k+1} in (5.6) is symmetric, positive definite, justifying the previous calculations. We have therefore established the following result.

Theorem 5.1. *Assume $B_k > 0$ and $y_k^T s_k > 0$. A solution B^* of the problem (5.4), in the regularized sense described above, is given by (5.6).*

5.3 BFGS-like methods

From Theorem 5.1, we get the following quasi-Newton iteration:

$$x_{k+1} = x_k - \alpha_k B_k^{-1}\nabla f(x_k), \quad x_0 = \bar{x}, \tag{5.7a}$$

$$B_{k+1}^{-1} = \left(I_n - \frac{y_k y_k^T}{y_k^T y_k}\right) B_k^{-1} \left(I_n - \frac{y_k y_k^T}{y_k^T y_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}, \quad B_0 = I_n. \tag{5.7b}$$

Now let $v_k \in \mathbb{R}^n$ be any vector not orthogonal to y_k. Then

$$P_k(v_k) := \frac{y_k v_k^T}{y_k^T v_k}$$

is an oblique projection onto y_k. Employing $P_k(v_k)$ and its transpose in place of Π_{y_k} in (5.5) and performing the variational analysis after regularisation, we get a BFGS-like iteration

$$B_{k+1}^{-1} = (I_n - P_k(v_k))^T B_k^{-1} (I_n - P_k(v_k)) + \frac{s_k s_k^T}{y_k^T s_k}. \tag{5.8}$$

In particular, if $v_k = s_k$, the corresponding oblique projection is

$$P_k(s_k) = \frac{y_k s_k^T}{y_k^T s_k}.$$

In such case, (5.8) is just the standard BFGS iteration for the inverse approximate Hessian (see (4.10))

$$B_{k+1}^{-1} = \left(I_n - \frac{y_k s_k^T}{y_k^T s_k}\right)^T B_k^{-1} \left(I_n - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}. \tag{5.9}$$

Here $T_k = I_n - P_k(s_k)$ is a rank $n-1$ matrix satisfying $T_k y_k = 0$ as is $I - \Pi_{y_k}$.

Corollary 5.2. *Assume $B_k > 0$ and $y_k^T s_k > 0$. A solution B^* of the problem (5.3) is given by the standard BFGS iteration (5.9).*

Proof. We show that in the limit, as $\epsilon \searrow 0$, $D_1 := \mathbb{D}(B^{-1} \| B_k^{-1})$ and $D_2^\epsilon := \mathbb{D}\left(B^{-1} \| \left(I_n - \frac{y_k s_k^T}{y_k^T s_k} + \epsilon I_n\right)^T B_k^{-1} \left(I_n - \frac{y_k s_k^T}{y_k^T s_k} + \epsilon I_n\right)\right)$ only differ by terms not depending on B.

$$\begin{aligned} D_2^\epsilon = \frac{1}{2}\Bigg\{ & \log\det\left(B^{-1} B_k\right) + \\ & + \log\det\left[\left(I_n - \frac{y_k s_k^T}{y_k^T s_k} + \epsilon I_n\right)^{-1} \left(I_n - \frac{y_k s_k^T}{y_k^T s_k} + \epsilon I_n\right)^{-T}\right] + \\ & + tr\left[\left((1+\epsilon) I_n - \frac{y_k s_k^T}{y_k^T s_k}\right)^T B_k^{-1} \left((1+\epsilon) I_n - \frac{y_k s_k^T}{y_k^T s_k}\right) B\right] - n \Bigg\}. \end{aligned}$$

Observe that the second term does not depend on B.
The third term $tr[\cdots]$ is equal to

$$\begin{aligned} & (1+\epsilon)^2 tr\left[B_k^{-1} B\right] - (1+\epsilon) tr\left[B_k^{-1} \frac{y_k s_k^T}{y_k^T s_k} B\right] + \\ & \qquad - (1+\epsilon) tr\left[\frac{s_k y_k^T}{y_k^T s_k} B_k^{-1} B\right] + tr\left[\frac{s_k y_k^T}{y_k^T s_k} B_k^{-1} \frac{y_k s_k^T}{y_k^T s_k} B\right]. \end{aligned}$$

For symmetric matrices B satisfying the secant equation $Bs_k = y_k$ we have $s_k^T B = y_k^T$. In particular:

$$tr\left[B_k^{-1}\frac{y_k s_k^T}{y_k^T s_k}B\right] = tr\left[B_k^{-1}\frac{y_k y_k^T}{y_k^T s_k}\right]$$

and

$$tr\left[\frac{s_k y_k^T}{y_k^T s_k}B_k^{-1}\frac{y_k s_k^T}{y_k^T s_k}B\right] = tr\left[\frac{s_k y_k^T}{y_k^T s_k}B_k^{-1}\frac{y_k y_k^T}{y_k^T s_k}\right].$$

Note that the last two expressions obtained do not depend on B.
Observe also that, by the circulant property of the trace and again by the secant equation,

$$tr\left[\frac{s_k y_k^T}{y_k^T s_k}B_k^{-1}B\right] = tr\left[B\frac{s_k y_k^T}{y_k^T s_k}B_k^{-1}\right] = tr\left[\frac{y_k y_k^T}{y_k^T s_k}B_k^{-1}\right],$$

which does not depend on B.
Omitting the terms that do non depend on B, we have

$$D_2^\epsilon = \frac{1}{2}\left[\log\det(B^{-1}B_k) + (1+\epsilon)^2 tr(B_k^{-1}B) - n\right].$$

Recalling that

$$D_1 = \frac{1}{2}\left[\log\det(B^{-1}B_k) + tr(B_k^{-1}B) - n\right]$$

and omitting the terms that do not depend on B, we have

$$D_2^\epsilon - D_1 = \frac{1}{2}\left[(\epsilon^2 + 2\epsilon)\, tr(B_k^{-1}B)\right] \longrightarrow 0, \quad \text{for } \epsilon \to 0.$$

$\square$

Chapter 6

Numerical experiments

Theorists working in nonlinear programming area, as well as practical optimizers, always need to evaluate nonlinear optimization algorithms. Due to the hypotheses introduced in order to prove convergence and complexity of algorithms, the theory is not enough to establish the efficiency and the reliability of a method. As a consequence the only way to see the "power" of an algorithm remains its implementation in computer codes and its testing on large classes of test problems of different structures and characteristics.

Generally, two types of (unconstrained) nonlinear programming problems can be identified: "artificial problems" and "real-life problems". The artificial nonlinear programming problems are used to see the behavior of the algorithms in different difficult situations like long narrow valleys, functions with significant null-space effects, essentially unimodal functions, functions with a huge number of significant local optima, etc. The main characteristic of artificial nonlinear programming problems is that they are relatively easy to manipulate and to use into the process of algorithmic invention. Besides, the optimizer may rapidly modify the problem in order to test the algorithm in different challenging conditions.
Real-life problems, on the other hand, are coming from different sources of applied optimization problems like physics, chemistry, engineering, biology, economy, oceanography, astronomy, meteorology, etc. Unlike artificial (unconstrained) nonlinear programming problems, real-life problems are not easily available and are difficult to manipulate. They may have complicated algebraic (or differential) expressions, may depend on a huge amount of data, and possibly are dependent on some parameters which must be estimated in a specific way.

The examples considered in this work are only of the first type, that is artificial problems. They are taken from some huge colletions of unconstrained optimization test functions [6], [7], [8].

6.1 BFGS and BFGS-like algorithms

Before providing some examples, we give the pseudocodes of BFGS and BFGS-like algorithms.

The standard BFGS algorithm has the form:

```
1: procedure BFGS($f, Gf, x_0, tol, \eta, Maxiter$)
2:     $B_0 \leftarrow I_d$
3:     $x \leftarrow x_0$
4:     $B \leftarrow B_0$
5:     for $n = 1, ..., Maxiter$ do
6:         $y \leftarrow Gf(x)$
7:         if $\|y\| < tol$ then
8:             break
9:         $SearchDirection \leftarrow -By$
10:        $\alpha \leftarrow LineSearch(f, Gf, x, SearchDirection)$
11:        if $\alpha$ exists then
12:            $\Delta x \leftarrow \alpha \cdot SearchDirection$
13:        else
14:            $\Delta x \leftarrow \eta \cdot SearchDirection$
15:        $T \leftarrow I_d - \frac{y \Delta x^T}{y^T \Delta x}$
16:        $B \leftarrow T^T B T + \frac{\Delta x \Delta x^T}{y^T dx}$
17:        $x \leftarrow x + \Delta x$
18:     return $x$
```

Algorithm 2: standard BFGS algorithm (5.7a)-(5.9)

The BFGS-like algorithm (5.7) has the form:

```
procedure BFGS-LIKE($f, Gf, x_0, tol, \eta, Maxiter$)
    $B_0 \leftarrow I_d$
    $x \leftarrow x_0$
    $B \leftarrow B_0$
    for $n = 1, ..., Maxiter$ do
        $y \leftarrow Gf(x)$
        if $\|y\| < tol$ then
            break
        $SearchDirection \leftarrow -By$
        $\alpha \leftarrow LineSearch(f, Gf, x, SearchDirection)$
        if $\alpha$ exists then
            $\Delta x \leftarrow \alpha \cdot SearchDirection$
        else
            $\Delta x \leftarrow \eta \cdot SearchDirection$
        $S \leftarrow I_d - \frac{yy^T}{y^T y}$
        $B \leftarrow S^T BS + \frac{\Delta x \Delta x^T}{y^T dx}$
        $x \leftarrow x + \Delta x$
    return $x$
```

Algorithm 3: BFGS-like algorithm (5.7)

Note that the inputs in both algorithms are:

- f, the function $f\colon \mathbb{R}^d \to \mathbb{R}$ to be minimized;
- Gf, the gradient of f;
- x_0, the initial point;
- tol (i.e. tolerance), which plays the role of stopping the algorithm if a certain condition is satisfied (we shall use $tol = 10^{-8}$);
- η, which is involved when the line search can not calculate the step length α (usually, η is equal to 10^{-2} or 10^{-4});
- $Maxiter$, the maximum number of iterations (we shall use $100 - 300$ iterations).

Observe that B and B_0 stand for B^{-1} and B_0^{-1}: we made this choice in order to simplify the notation. At the beginning, B_0 is set to I_d, which is the identity in $\mathbb{R}^d$. Note that the condition $\|y\| < tol$ -where $\|\cdot\|$ is the Euclidean norm on $\mathbb{R}^d$- is (almost) equivalent to $\nabla f(x) = 0$, which is a sufficient condition for x to be a minimizer of f when f is strongly convex (see theorem (2.3)). This is the reason why the algorithm stops if the condition $\|y\| < tol$ is satisfied. $LineSearch(\cdot)$ refers to a line search method, usually offered by the programming language used, and calculates the step length α. If the line search method does not converge, the quantity η is used in place of α.

6.2 Examples

While the effectiveness of the BFGS-like algorithms introduced in the previous Chapter needs to be tested on a significant number of large scale benchmark problems, we provide below some examples where the BFGS-like algorithm (Algorithm 2) appears to perform better than standard BFGS (Algorithm 3).

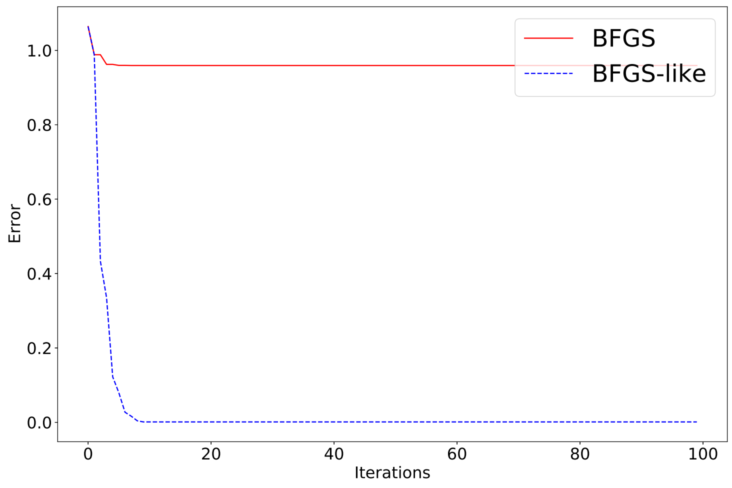
Each example has the following structure:

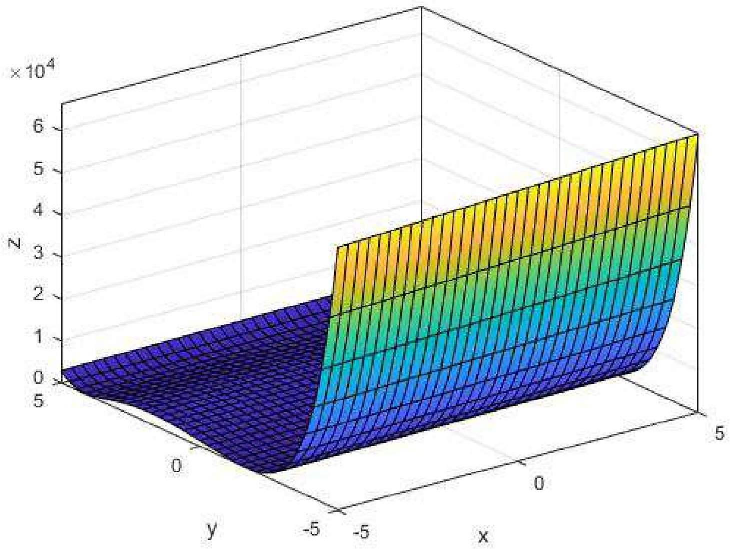
- name, dimensions, algebraic expression and main properties (convex, nonconvex etc.) of the function f;
- the absolute minimum x^* and the value of f at x^*;
- the initial point x_0;
- some comments about the performance of BFGS and BFGS-like methods;
- the plot of the decay of the error $\|x_n - x^*\|$ over a certain number of iterations for both algorithms (x_n is the current point at the iteration n and $\|\cdot\|$ is always the Euclidean norm on $\mathbb{R}^d$);
- the graphical representation of the function (we shall consider f always on 2 dimensions).

First, consider the (nonconvex) Freudenstein and Roth function f in 2 dimensions:

$$f(x_1, x_2) = (-13+x_1+((5-x_2)x_2-2)x_2)^2 + (-29+x_1+((x_2+1)x_2-14)x_2)^2$$

whose minimum point is $x^* = (5, 4)$ and $f(x^*) = 0$. Take as starting point $x_0 = (3, 2)$. As we can see in the figure below, BGFS-like converges after a few steps, whereas BFGS does not converge.

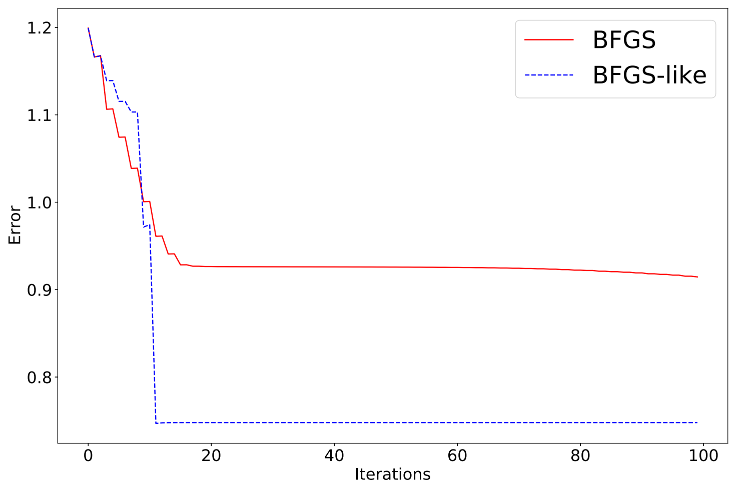


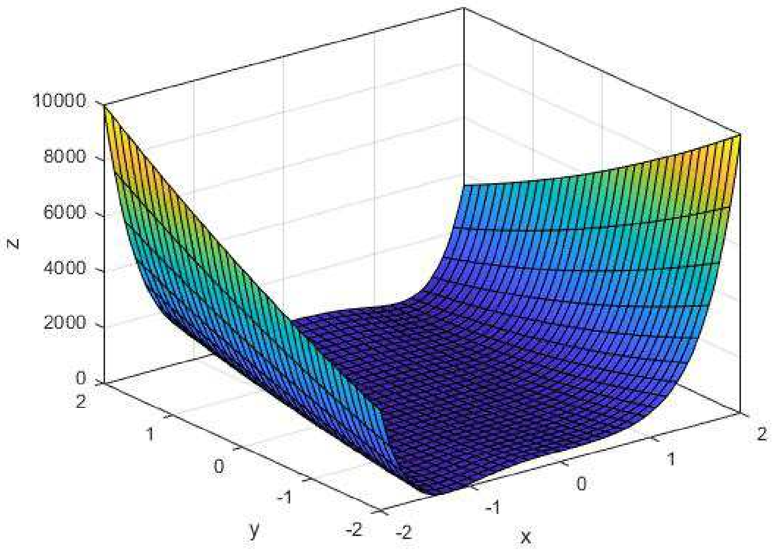


Consider now the (nonconvex) White and Holst function on $\mathbb{R}^2$:

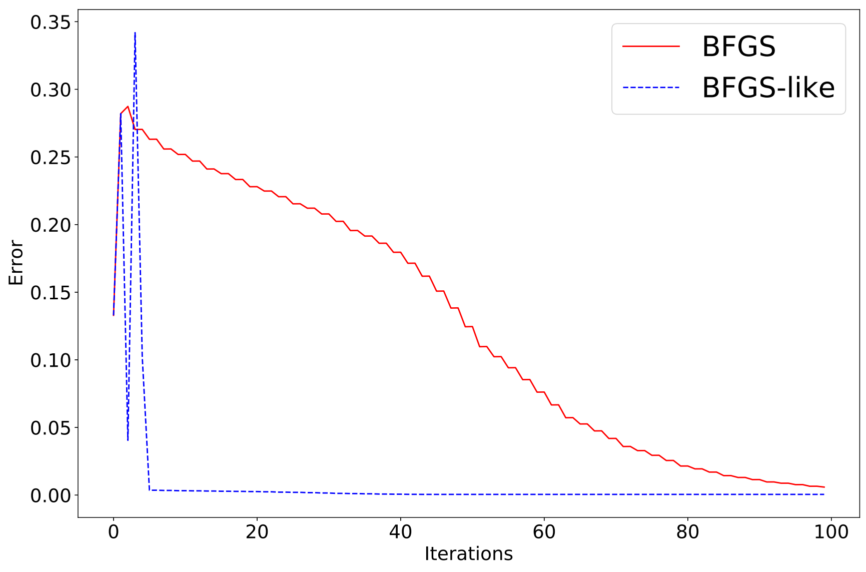
$$f(x_1, x_2) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2.$$

It has an absolute minimum at $x^* = (1, 1)$ and $f(x^*) = 0$. First, we take as initial point $x_0 = (0, 0)$ and we observe that both methods do not converge, see the plot below.





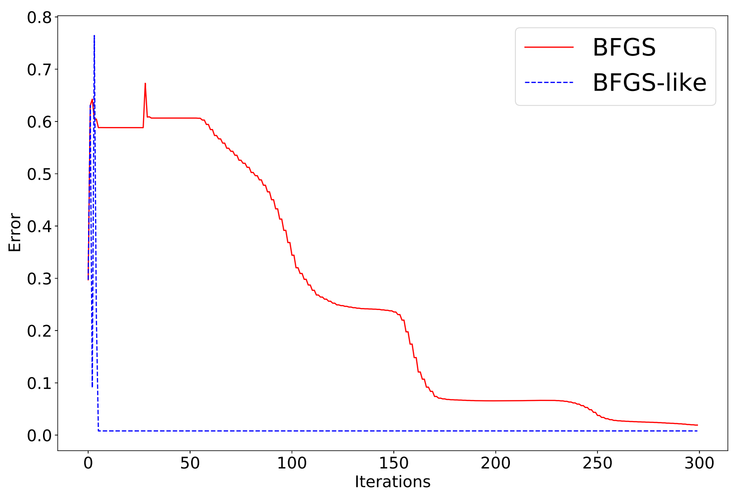
Instead, initiating the recursions at $x_0 = (0.9, 0.9)$, both algorithms converge to the absolute minimum. After a few initial steps, BFGS-like appears to perform better than BFGS.



The extended White and Holst function on 10 dimensions is given by the following:

$$f(x) = \sum_{i=1}^{5} 100(x_{2i} - x_{2i-1}^3)^2 + (1 - x_{2i-1})^2.$$

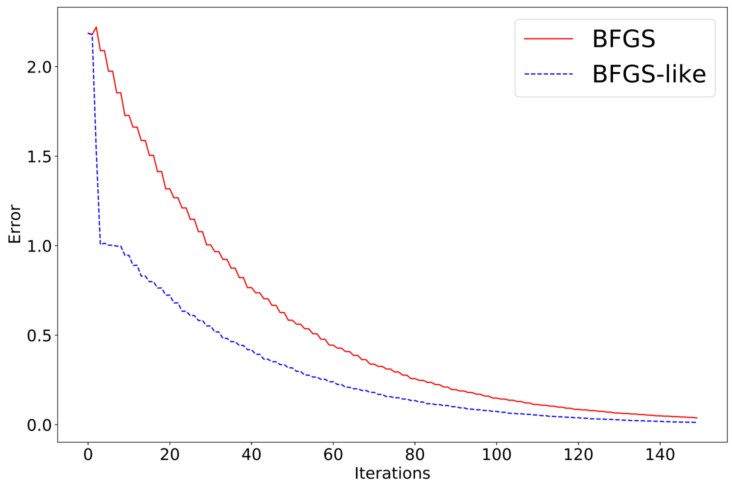
It has an absolute minimum at $x_i^* = 1$, $i = 1, ..., 10$ and $f(x^*) = 0$. Take as initial point: $x_0 = (0.9, \ldots, 0.9)$. After a few iterations, BFGS-like appears to perform better than BFGS.

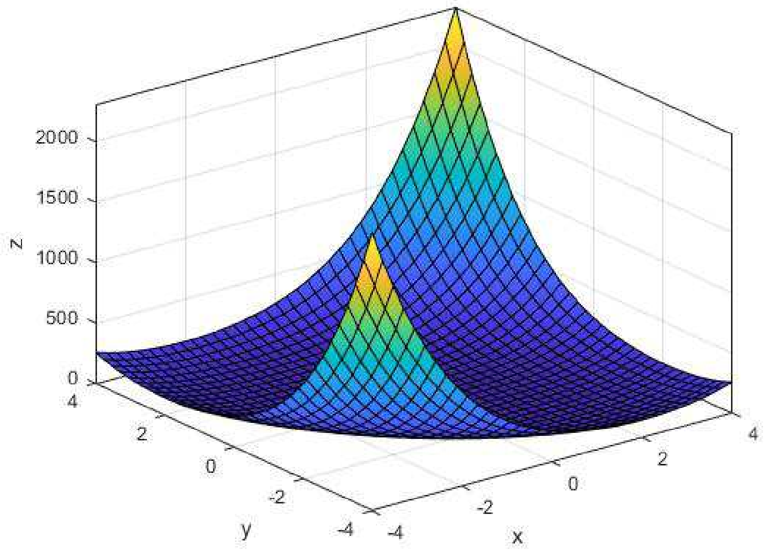


Consider now the (nonconvex) $PSC1$ function on $\mathbb{R}^2$, which is given by:

$$f(x_1, x_2) = (x_1^2 + x_2^2 + x_1 x_2)^2 \ + \ (\sin x_1)^2 \ + \ (\cos x_2)^2$$

whose minimum point is $x^* = (0, 0)$, $f(x^*) = 1$. Take as starting point: $x_0 = (3, 0.1)$. From the figure, it is clear that both methods perform great, but BFGS-like performs a little better than BFGS.

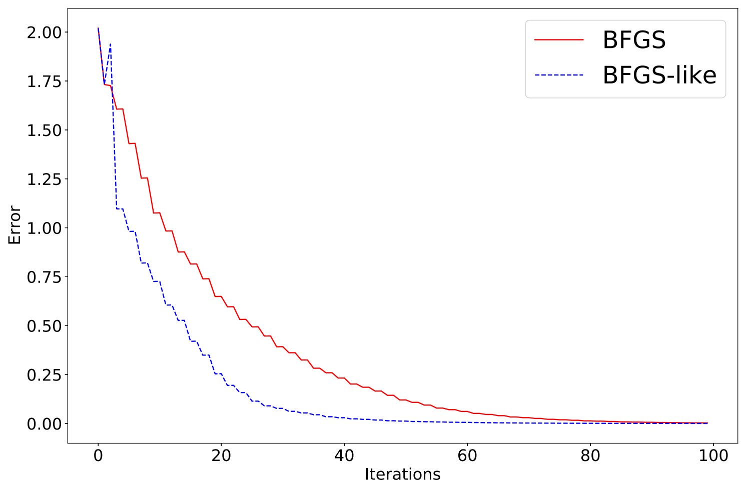


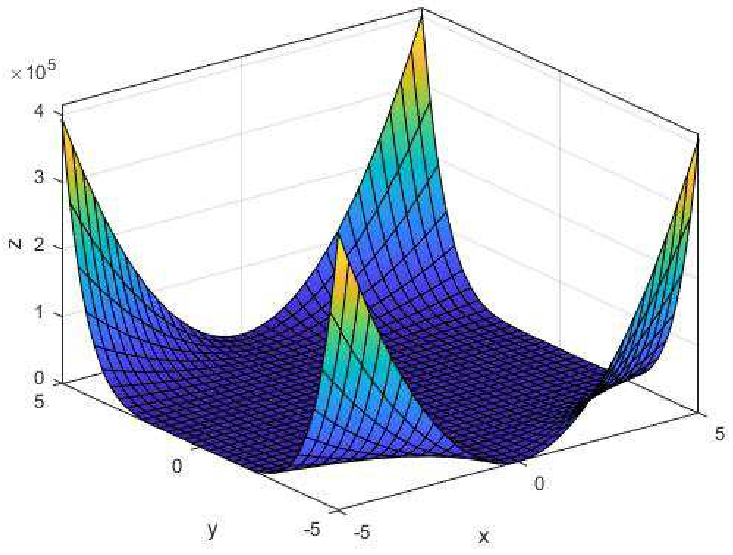


Consider now the (nonconvex) Beale function, defined by

$$f(x_1, x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2,$$

for all $(x_1, x_2) \in \mathbb{R}^2$. The global minimum is located at $x^* = (3, 0.5)$, $f(x^*) = 0$. Take as initial point: $x_0 = (1, 0.8)$. After a few steps, BFGS-like converges faster than BFGS. Anyway, both methods perform well.

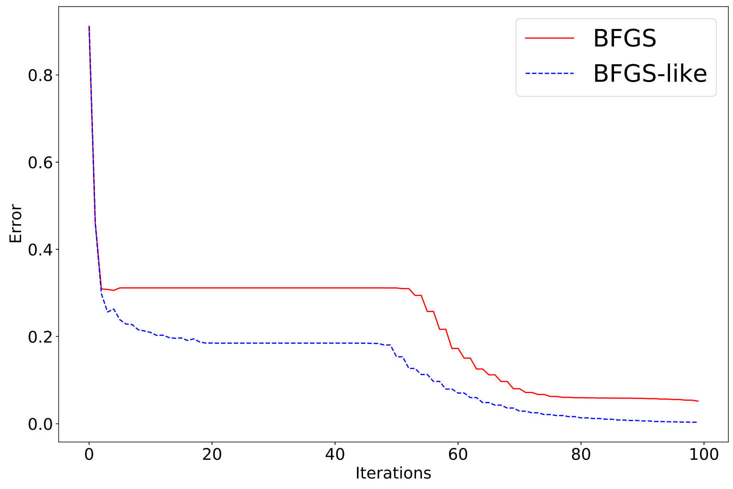


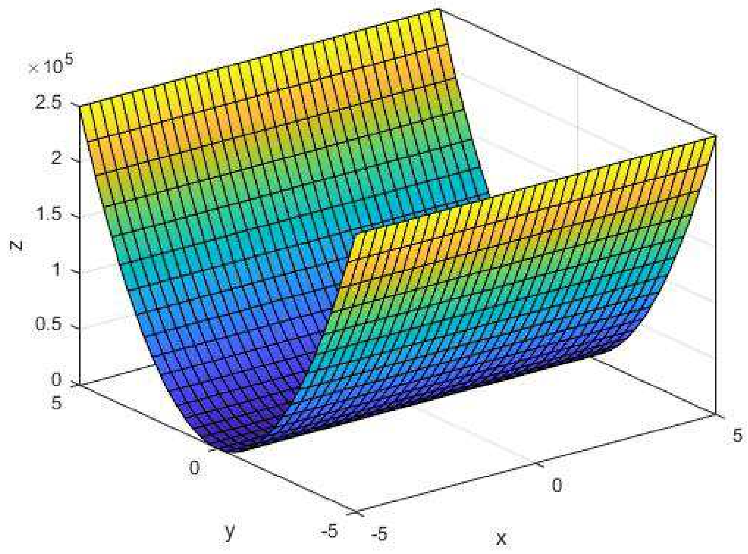


Consider the following (strictly, but nonstrongly, convex) function on $\mathbb{R}^{10}$:

$$f(x) = \sum_{i=1}^{9} (e^{x_i} - i x_i) + 10000 x_{10}^2$$

whose minimum point is $x^* \approx (0, 0.69, 1.1, 1.39, 1.61, 1.8, 1.95, 2.08, 2.20, 0)$, $f(x^*) \approx -34.1$. Taking as starting point the origin, both methods converge, but BFGS-like performs better than BFGS.

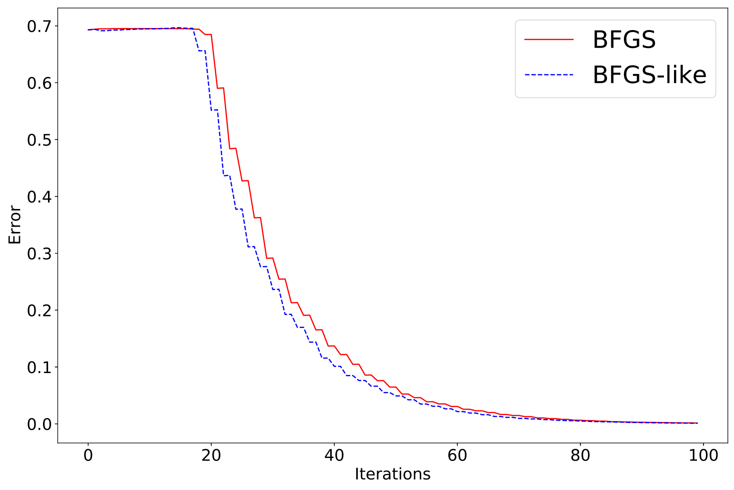


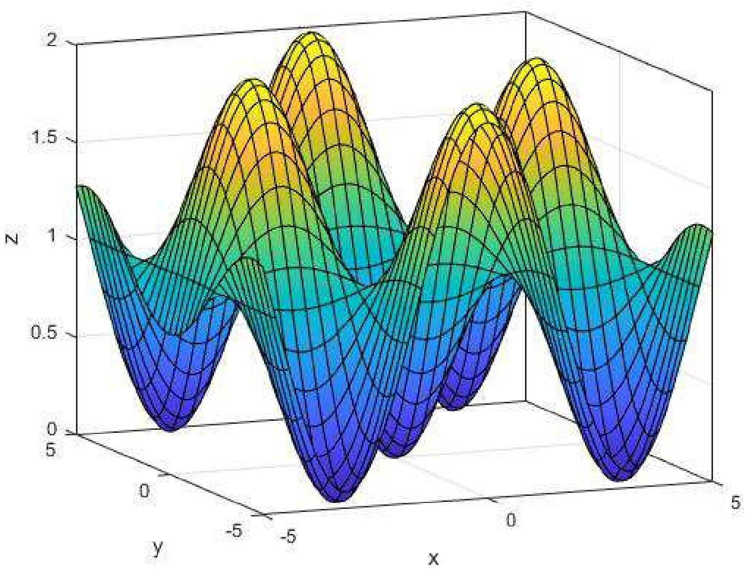


Finally, consider the (nonconvex) Griewank function in 2 dimensions:

$$f(x_1, x_2) = \frac{x_1^2}{4000} + \frac{x_2^2}{4000} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right) + 1.$$

It has an absolute minimum at $x^* = (0, 0)$, $f(x^*) = 0$. Take as initial point: $x_0 = (0.9, 0.9)$. As we can see in the figure below, both methods converge rapidly, but BFGS-like performs a little better than BFGS.





Bibliography

[1] M. Pavon, A variational derivation of a class of BFGS-like methods, *https://arxiv.org/abs/1712.00680*, *Optimization*, **67**, 11, 2081-2089, 2018.

[2] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.

[3] R. Fletcher, A New Variational Result for Quasi-Newton Formulae, *SIAM J. Optimization*, 1991, **1**, No. 1 : pp. 18-21.

[4] D. Goldfarb, A family of variable metric methods derived by variational means, *Math. Comp.*, **24**, (1970), pp. 23-26.

[5] A. Ferrante and M. Pavon, Matrix Completion *à la* Dempster by the Principle of Parsimony, *IEEE Trans. Information Theory*, **57**, Issue 6, June 2011, 3925-3931.

[6] N. Andrei, An Unconstrained Optimization Test Functions Collection, *Advanced Modeling and Optimization*, vol. 10, no. 1, pp.147-161, 2008.

[7] M. Jamil and X-S. Yang, A literature survey of benchmark functions for global optimisation problems, *Int. J. Mathematical Modelling and Numerical Optimisation*, Vol. 4, No. 2, pp.150–194.

[8] S. Surjanovic and D. Bingham, Optimization Test Functions, *https://www.sfu.ca/ ssurjano/optimization.html*, 2013.