Université de Montréal

Revisiting optimization algorithms for maximum likelihood estimation

par
Anh Tien Mai

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en computer science

Décembre, 2012

Université de Montréal
Faculté des arts et des sciences

Ce mémoire intitulé:

Revisiting optimization algorithms for maximum likelihood estimation

présenté par:

Anh Tien Mai

a été évalué par un jury composé des personnes suivantes:

Pierre l'Écuyer,	président-rapporteur
Fabian Bastin,	directeur de recherche
Michel Toulouse,	codirecteur
Jacques Ferland,	membre du jury

Mémoire accepté le: . . . .

RÉSUMÉ

Parmi les méthodes d'estimation de paramètres de loi de probabilité en statistique, le maximum de vraisemblance est une des techniques les plus populaires, comme, sous des conditions légères, les estimateurs ainsi produits sont consistants et asymptotiquement efficaces. Les problèmes de maximum de vraisemblance peuvent être traités comme des problèmes de programmation non linéaires, éventuellement non convexe, pour lesquels deux grandes classes de méthodes de résolution sont les techniques de région de confiance et les méthodes de recherche linéaire. En outre, il est possible d'exploiter la structure de ces problèmes pour tenter d'accélerer la convergence de ces méthodes, sous certaines hypothèses. Dans ce travail, nous revisitons certaines approches classiques ou récemment développées en optimisation non linéaire, dans le contexte particulier de l'estimation de maximum de vraisemblance. Nous développons également de nouveaux algorithmes pour résoudre ce problème, reconsidérant différentes techniques d'approximation de hessiens, et proposons de nouvelles méthodes de calcul de pas, en particulier dans le cadre des algorithmes de recherche linéaire. Il s'agit notamment d'algorithmes nous permettant de changer d'approximation de hessien et d'adapter la longueur du pas dans une direction de recherche fixée. Finalement, nous évaluons l'efficacité numérique des méthodes proposées dans le cadre de l'estimation de modèles de choix discrets, en particulier les modèles logit mélangés.

Mots clés : optimization, région de confiance, recherche linéaire, estimation, maximum de vraisemblance, approximation de hessien, basculement entre modèles, choix discrets, logit mélangé.

ABSTRACT

Maximum likelihood is one of the most popular techniques to estimate the parameters of some given distributions. Under slight conditions, the produced estimators are consistent and asymptotically efficient. Maximum likelihood problems can be handled as non-linear programming problems, possibly non convex, that can be solved for instance using line-search methods and trust-region algorithms. Moreover, under some conditions, it is possible to exploit the structures of such problems in order to speed-up convergence. In this work, we consider various non-linear programming techniques, either standard or recently developed, within the maximum likelihood estimation perspective. We also propose new algorithms to solve this estimation problem, capitalizing on Hessian approximation techniques and developing new methods to compute steps, in particular in the context of line-search approaches. More specifically, we investigate methods that allow us switching between Hessian approximations and adapting the step length along the search direction. We finally assess the numerical efficiency of the proposed methods for the estimation of discrete choice models, more precisely mixed logit models.

Keywords: Optimization, trust-region, line-search, estimation, maximum likelihood, Hessian approximation, model switching, discrete choice, mixed logit.

CONTENTS

RÉSUMÉ . . . . **iii**

ABSTRACT . . . . **iv**

CONTENTS . . . . **v**

LIST OF TABLES . . . . **viii**

LIST OF FIGURES . . . . **ix**

LIST OF ABBREVIATIONS . . . . **x**

NOTATION . . . . **xi**

ACKNOWLEDGMENTS . . . . **xii**

CHAPTER 1: INTRODUCTION **1**

CHAPTER 2: BACKGROUND **4**

2.1 Maximum likelihood estimation 4

2.1.1 Consistency of MLE 6

2.1.2 Asymptotic normality for MLE 8

2.1.3 Asymptotic covariance estimation for MLE 10

2.2 Optimization algorithms 11

2.2.1 Trust region method 12

2.2.2 Line search method 15

2.2.3 Stopping conditions 18

2.3 Hessian approximations 19

2.3.1 Statistical approximation 19

2.3.2 Secant approximation 20

2.3.3 Combined approximation 22

CHAPTER 3: SWITCHING APPROACHES FOR MAXIMUM LIKELIHOOD ESTIMATION 26
3.1 Model switching 26
3.2 Predictive model 30
3.3 Retrospective model 31
3.4 Multi sub-problem model 34
3.5 Multi sub-problem with the BHHH model 35

CHAPTER 4: ADAPTIVE LINE SEARCH 38

CHAPTER 5: DISCRETE CHOICE THEORY 42
5.1 Decision-maker 42
5.2 The alternatives 42
5.3 Attributes 43
5.4 Utilities and Decision rule 43
5.5 Random utility models 46
5.6 Mixed-logit model estimation 49

CHAPTER 6: NUMERICAL ASSESSMENT 52
6.1 Switching algorithms for mixed-logit models 52
6.2 Discrete choice data sets 55
6.2.1 Cybercar model 55
6.2.2 IRIS model 56
6.3 Numerical assessment with AMLET 58
6.4 Numerical experiments 59
6.4.1 Comparison between classical algorithms 59
6.4.2 Model switching algorithms with the trust-region method 67
6.4.3 Model switching algorithms with the line-search method 71
6.4.4 Comparing our adaptive line-search algorithm 73
6.5 Summary and conclusion 76

CHAPTER 7: **CONCLUSIONS AND FURTHER RESEARCH PERSPECTIVES** **79**

BIBLIOGRAPHY **85**

LIST OF TABLES

6.I List of algorithms 54
6.II Summary of stopping criteria 54
6.III The variables and their admissible levels for SP2 55
6.IV Distribution of the components of β with the real data SP2 56
6.V Distribution of the components of β with the real data IRIS 58
6.VI Optimization time of basic trust-region algorithms 65
6.VII Optimization time of basic line-search algorithms 65
6.VIII Optimization time of trust-region switching algorithms 68
6.IX Number of iterations of trust-region switching algorithms 70
6.X Rate of successful runs for difficult case [IRIS-LOGNORMAL] 70
6.XI Optimization time of line-search switching algorithms 72
6.XII Optimization time of classical and adaptive line-search algorithms 75
6.XIII Numerical results of classical and adaptive line-search for difficult case [IRIS-NORMAL] 75
6.XIV Numerical results of classical and adaptive line-search for difficult case [IRIS-LOGNORMAL] 76
6.XV Optimization time of basic algorithms 77
6.XVI Optimization time of trust-region switching algorithms 77

LIST OF FIGURES

6.1 Optimization time of basic trust-region algorithms [SP2] 60

6.2 Optimization time of basic trust-region algorithms [IRIS NORMAL] 61

6.3 Optimization time of basic trust-region algorithms [IRIS LOG-NORMAL] 62

6.4 Optimization time of combined approaches [SP2] 62

6.5 Optimization time of combined approaches [IRIS-NORMAL] 63

6.6 Optimization time of combined approaches [IRIS-LOGNORMAL] 63

6.7 Optimization time of basic trust-region algorithms [SP2] 64

6.8 Optimization time of basic trust-region algorithms [IRIS-NORMAL] . . . 64

6.9 Optimization time of basic trust-region algorithms [IRIS-LOGNORMAL] . 65

6.10 Optimization time of basic line-search algorithms [IRIS-NORMAL] 66

6.11 Optimization time of basic line-search algorithms [IRIS-LOGNORMAL] . 66

6.12 Optimization time of trust-region switching algorithms [IRIS-NORMAL] . 67

6.13 Optimization time of trust-region switching algorithms [IRIS-NORMALLOG] 68

6.14 Number of iterations of trust-region switching algorithms [IRIS-NORMAL] 69

6.15 Number of iterations of trust-region switching algorithms [IRIS-LOGNORMAL] 69

6.16 Optimization time of line-search switching algorithms [IRIS-NORMAL] . 71

6.17 Optimization time of line-search switching algorithms [IRIS-LOGNORMAL] 72

6.18 Optimization time of classical and adaptive line-search [IRIS-NORMAL] . 74

6.19 Optimization time of classical and adaptive line-search [IRIS-LOGNORMAL] 74

LIST OF ABBREVIATIONS

AMLET : Another Mixed Logit Estimation Tool;

ALNS : Adaptive line-search algorithm;

BHHH : Berndt-Hall-Hall-Hausman approximation;

BFGS : Broyden-Fletcher-Goldfarb-Shanno approximation;

BTR : Basic trust-region algorithm;

MLE : Maximum likelihood estimation;

LNS : Line-search algorithm;

SR1 : Symmetric-rank 1 approximation;

SW : Switching.

NOTATION

$\mathbb{R}$	Set of real number
$LL(\cdot)$	Log-likelihood function
$E(\cdot)$	Expectation operator
$\xrightarrow{p}$	Convergence in probability
$\lvert\cdot\rvert$	Absolute function
$\lvert\lvert\cdot\rvert\rvert$	Euclidean norm
$\nabla_\theta f(\cdot)$	Gradient of f with respect to θ
$\nabla^2_{\theta\theta} f(\cdot)$	Hessian of f with respect to θ
$\int$	Integral
$N(\mu,\sigma^2)$	Normal distribution with mean μ and variance σ^2
$\ln N(\mu,\sigma^2)$	Lognormal distribution of parameters μ and σ
$\mathscr{B}_k$	Trust-region at the k^{th} iteration
H_k	Hessian or an approximation of it at the k^{th} iteration
$\mathscr{H}_k$	Set of Hessian approximations at the k^{th} iteration

ACKNOWLEDGMENTS

First of all my deepest thanks go to my supervisor, Fabian Bastin, for his supervision, advice, and guidance from the very early stage of this research as well as giving me his profound experiences through out the work. He has always helped me to see a clearer and bigger picture, listened carefully to my ideas and analysed my idea by the best of his knowledge and belief. Working with a young and smart professor like him makes me always feel free and find a lot of passion for my research. Thanks also to my co-supervisor Michel Toulouse, for his valuable comments both in my work and in this thesis, for his support, and for many interesting discussions. Above all and the most meaningful, my supervisors always make me feel a friend, which I appreciate from my heart.

I will forever be thankful to my best friend and also my office colleague, Thuy Anh Ta, for her interesting and valuable comments to organize and correct my thesis, for her encouragements in difficult times and for a lot of motivations to study and improve myself that she brought to me.

I also want to thank all my vietnamese friends in University of Montréal, for their helps at the first time I came to Montréal, for their valuable experience about the new life and new research environment in Canada. I am grateful to my professors in Department of Computer Science and Operation Research, who gave me precious lessons and paved the way for the completion. I also truly appreciate the excellent service provided by the departmental administrators and departmental secretaries. The list is long so I do not enumerate them here.

Finally, I would like to dedicate this thesis and all of my academic achievements to my parents and all the people in my family who have been anxiously waiting for its completion.

CHAPTER 1

INTRODUCTION

This research relates to non-linear, non-convex and non-constrained programming, which is part of mathematical programming. Mathematical programming studies problems where the aim is to find the optimal value of a given mathematical function (objective function or cost function) in a defined domain. In the present work, we consider twice-continuously differentiable non-linear functions, possibly non-convex, we aim to find their minimum over an unconstrained domain. It is usual to search such minimums using iterative algorithms, starting from arbitrary initial point and then performing iterative steps that aim at finding a locally optimal value, which could be a minimal solution under mild conditions. To date, trust-region and line-search techniques are among the most commonly applied iterative techniques to address the type of functions we consider in this thesis. These techniques were originally introduced as a globalization of the locally-converging Newton technique. In this setting, they often rely on a second-order Taylor-development of the objective function, therefore requiring the Hessian of the objective function to be available. The associated numerical cost associated to Hessian evaluation is however usually not affordable, and one prefers to construct some approximation of this Hessian, leading to so-called quasi-Newton techniques. The most popular approximations are BFGS (rank-2 update) and the symmetric rank-1 (SR1) update, both of them maintaining symmetry of the matrix and satisfying the secant condition. However, they may require a significant number of iterations before the approximation is good enough for the algorithm to converge.

In this thesis, we focus more specifically on maximum likelihood estimation (MLE) problem, aiming to investigate more efficient optimization algorithms for solving this problem. An alternative Hessian approximation has been proposed in this context by Berndt, Hall, Hall, and Hausman [6]. This approximation relies on the information identity property, and appears not expensive to compute, while reflecting better the problem structure. This explains the popularity of the approach, up to this date (see for instance

Train [27], Chapter 8). Unfortunately, the conditions needed to ensure validity of the information identity are difficult to satisfy, especially as they require a correctly formulated model. In practice, these conditions are often violated, and the estimation can fail to converge. This has led Bunch [8], who considered the log-likelihood problem as a particular case of generalized regression, to propose a technique relying on more than one quadratic model to approximate the objective function. Following Bunch's idea, our work also propose to use a set of Hessian approximations at each iteration. However, we develop more complex criteria for switching between quadratic models. We propose criteria that help to select specific matrices either to build a sub-problem in trust-region methods or to compute the search-direction in line-search methods. More specifically, we propose new algorithms that differ in the way the Hessian approximation is selected at each iteration. For testing the efficiency of our algorithm, we focus on parameter estimation for mixed logit model in the context of discrete choice theory.

Considering the interdisciplinary nature of this research, some background material is provided to facilitate the discussion. Therefore, the next chapter presents a relatively large introduction to maximum likelihood estimation. Some important properties of likelihood estimation are described, the Fisher information matrix is introduced which leads to the description of the BHHH approach. Optimization methods for computing maximum likelihood estimates such as the trust-region and line-search methods are introduced as well. Methods for approximating the Hessian matrix are described.

The two next chapters, Chapter 3 and Chapter 4, contain our main research contributions. In Chapter 3 we propose four new algorithms which are based on the general idea of model switching adapted to both trust-region and line-search methods: **The predictive** and **retrospective** algorithms select an Hessian approximation by considering available information at the current and previous iteration of the trust-region and line-search iterative optimization methods. **Multi sub-problems** is an algorithm designed only for the trust region methods, where sub-problems have to be considered and solved approximately. This algorithm has proved its efficiency in some difficult cases. We also present an improvement of **Multi sub-problems**, called **Multi sub-problems with BHHH**. This algorithm is designed based on the characteristics of the **Multi sub-problems** algorithm and the classical trust-region algorithm with BHHH update. In Chapter 4, we introduce a new algorithm called **adaptive line-search** which improves the line-search method by controlling the length of the search direction.

As pointed out earlier, we consider Mixed-logit model in discrete choice theory as a good framework for testing our new algorithms. Chapter 5 describes important concepts in discrete choice theory such as discrete choice model and random utility model.

Chapter 6 measures the efficiency of each algorithm using real complex data from discrete choice problems. Our numerical results are compared with existing methods, which is then followed by discussions. The test results prove the numerical efficiency of our approach in many cases. Note that all the tests are performed with real data in discrete choice. Finally, Chapter 7 contains discussions and comments on future research.

CHAPTER 2

BACKGROUND

This chapter introduces well-know methods in optimization such as *trust-region method*, *line-search method* and *Hessian approximation*. These methods are described as steps to solve the maximum likelihood estimation, a very important problem in fields like economics. Thus, we first give the definition and the principle of the maximum likelihood estimation. Background on trust-region and line-search methods is given in the next section, as well as recent results in optimization. The last section describes the classical methods to approximate the Hessian matrix, which is one of the most important factors for the optimization algorithms.

2.1 Maximum likelihood estimation

Maximum likelihood is one the most popular techniques in statistics to estimate the parameters of a model, given some observations that are assumed to be the realizations of some random vector. More precisely, consider a random vector Y, and assume we have N observations independently drawn from this vector. Let assume for now that Y is continuous (the discrete case can be treated in a similar way). Denote by $f(Y|\theta)$ the probability density function (pdf) of Y, conditioned on a set of parameters θ. The distribution would be completely characterized if we knew the particular value of θ, say θ_0, corresponding to the population under interest. In the discrete case, we would consider the probability mass function instead of the density. Since the observations are assumed to be independent, the joint density is the product of the individual densities:

$$f(y_1, y_2, \ldots, y_N \,|\, \theta) = \prod_{i=1}^{N} f(y_i|\theta) = L(\theta|y).$$

However, we are not interested in the observations, that are known, but rather in θ, so it is convenient to consider a function of θ that would follow the value of the joint density, given the observation $y_1, \ldots, y_N$:

$$L(\theta \,|\, y_1, y_2, \ldots, y_N) = f(y_1, y_2, \ldots, y_N \,|\, \theta).$$

Since we do not know θ_0, we will approximate it by computing an estimator $\hat{\theta}_N$ of it, that can be judged as the most likely value for θ, given our observations. This is simply done by maximizing the function $L(\theta \,|\, y_1, \ldots, y_N)$ with respect to θ:

$$\hat{\theta}_N = \arg\max_{\theta \in \Theta} L(\theta \,|\, y_1, y_2, \ldots, y_N),$$

where we confine the search to the parameter space Θ, and we assume that θ_0 belongs to Θ. The function $L(\theta \,|\, y_1, y_2, \ldots, y_N)$ is called the likelihood function, and $\hat{\theta}_N$ the maximum likelihood estimator.

In practice, due to numerical stability issues, it is often more convenient to work with the logarithm of the likelihood function, called the log-likelihood:

$$LL_N(\theta) = \ln L(\theta \,|\, y_1, \ldots, y_N) = \sum_{i=1}^{N} \ln f(y_i|\theta) \tag{2.1}$$

or the average log-likelihood

$$\frac{1}{N} \sum_{i=1}^{N} \ln f(y_i|\theta). \tag{2.2}$$

The likelihood function can be denoted simply by $L(\theta)$ or by its logarithm $LL_N(\theta)$. Maximizing the log-likelihood is equivalent to maximize the likelihood since the logarithm operator is concave:

$$\hat{\theta}_N = \arg\max_{\theta \in \Theta} LL_N(\theta).$$

For the maximum likelihood estimation (MLE) problem, we have to consider the **identifiability** condition. Identifiability is a necessary condition for the limiting objective function to have a unique maximum. In the context of maximum likelihood estimation, identification is defined as:

> The parameter vector θ_0 is identifiable (or estimable) if for any other parameter vector θ' and for some data y we have: $f(y|\theta') \neq f(y|\theta_0)$.

MLE is also attractive because of its asymptotic properties. First, it is consistent as $\hat{\theta}_N$ almost surely converges to θ_0 as N grows to infinity. While almost sure convergence is the strongest type of convergence in statistics, it only expresses that the estimator is close to the true parameter when the number of observations is high. Another important property, called asymptotic normality, shows that the distribution function of $\sqrt{N}(\hat{\theta}_N - \theta_0)$ converges to the multinormal distribution function with mean zero and variance-covariance matrix $\mathscr{V}$, i.e, $\sqrt{N}(\hat{\theta}_N - \theta_0) \xrightarrow{d} N(0, \mathscr{V})$. The variance-covariance matrix $\mathscr{V}$ of the limiting distribution is referred to as the asymptotic variance-covariance matrix of $\hat{\theta}_N$. In the section below we describe in more details these properties. We will denote by $E_0[\cdot]$ or $E[\cdot|\theta_0]$ the expectation based on the true parameter θ_0.

2.1.1 Consistency of MLE

If the function $LL_N(\theta)$ converges in probability to $E_0[\ln f(y|\theta)]$ for each θ when N goes to infinity, and if $E_0[\ln f(y|\theta)]$ reaches its maximum for $\theta = \theta_0$, then the limit of the sequence $\hat{\theta}_N, N \geq 1$, should be θ_0, under conditions allowing interchanging the maximization and limit operations. The point wise convergence of $LL_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ln f(y_i|\theta)$ to $E_0[LL_N(\theta)]$ is given by *the law of large numbers*. Moreover, $E_0[\ln f(y|\theta)]$ has a unique maximum at the true parameter under the *information inequality* below:

> If θ_0 is identifiable and $E_0[|\ln f(y|\theta)|] < \infty \forall \theta \in \Theta$, then $E_0[\ln f(y|\theta)]$ has a unique maximum at θ_0.

Sufficient conditions for the maximum of the limit to be the limit of the maximum are that the convergence of the log-likelihood converges in probability uniformly on the parameters set Θ, i.e.

$$\sup_{\theta\in\Theta} |LL_N(\theta) - E_0[\ln f(y|\theta)]| \xrightarrow{P} 0.$$

and that Θ is compact. For more details, see Newey and McFadden, Section 2 [22].

The consistency result for the MLE problem can be formulated as follows:

Suppose that:

C1. (Identification) θ_0 is identifiable $(f(y|\theta) \neq f(y|\theta_0), \forall\theta \neq \theta_0)$.

C2. (Compactness of parameter space) $\theta_0 \in \Theta$, which is an compact subset of $\mathbb{R}^K$, $K < N$.

C3. (Continuity of the log-likelihood) $\ln f(y_i|\theta)$ is continuous in Θ with probability one, $i = 1,\ldots,N$.

C4. (Dominance condition) $E_0[\sup_{\theta\in\Theta} |\ln f(y|\theta)|] < \infty$.

Then $\hat{\theta}_N \xrightarrow{p} \theta_0$.

That is, if the four conditions above are satisfied then the MLE has the consistency property. Since $f(y|\theta_0)$ is a density function, we have $\int f(y|\theta_0)dy = 1$, implying that

$$\nabla_\theta \int f(y|\theta_0)dy = 0.$$

Moreover, if the function $\ln f(y|\theta)$ is continuously differentiable in a neighbourhood $\mathscr{N}$ of θ_0, then $\int \ln f(y|\theta) f(y|\theta_0)dy$ is continuously differentiable and one may interchange the integral and gradient operators:

$$\nabla_\theta \int \ln f(y|\theta) f(y|\theta_0)dy = \int \nabla_\theta [\ln f(y|\theta) f(y|\theta_0)]dy.$$

Therefore, the score function $g(y,\theta) = \nabla_\theta \ln f(y|\theta)$ has a mean equal to zero when evaluated at θ_0 because

$$E_0[g(y,\theta_0)] = E_0[\nabla_\theta \ln f(y|\theta_0)] = E_0[\frac{\nabla_\theta f(y|\theta_0)}{f(y|\theta_0)}]$$

$$\Rightarrow E_0[g(y,\theta_0)] = \int \nabla_\theta f(y|\theta_0) dy = \nabla_\theta \int f(y|\theta_0) dy = 0.$$

Consequently, the expectation of the gradient of the log-likelihood, evaluated at the true parameters, is also equal to zero:

$$E_0[\nabla_\theta LL_N(\theta_0)] = E_0\left[\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \ln f(y_i|\theta_0)\right] = 0.$$

The equation $E_0[\nabla_\theta LL_N(\theta_0)] = 0$ is called the **likelihood equation**.

2.1.2 Asymptotic normality for MLE

We add the following conditions in order to establish the asymptotic normality of the MLE.

Suppose that the condition C1 through C4 are satisfied and

C5. θ_0 and $\hat{\theta}_N = \arg\max_{\theta\in\Theta} LL_N(\theta)$ belongs to some open subset of Θ almost surely, for N large enough.

C6. $f(y|\theta)$ is twice continuously differentiable and $f(y|\theta) > 0$ in a neighbourhood $\mathscr{N}$ of θ_0.

C7. $\int \sup_{\theta\in\mathscr{N}} ||\nabla_\theta f(y|\theta)|| dy < \infty$, $\int \sup_{\theta\in\mathscr{N}} ||\nabla^2_{\theta\theta} f(y|\theta)|| dy < \infty$.

C8. The Fisher information matrix $I(\theta_0) = E_0[g(y,\theta_0)g^T(y,\theta_0)]$ exists and is non-singular.

C9. $E_0[\sup_{\theta\in\mathscr{N}} ||\nabla^2_{\theta\theta} \ln f(y|\theta)||] < \infty$.

Then the log-likelihood is differentiable and $\hat{\theta}_N$ is in the interior of the parameter set Θ, C5 implies that

$$\nabla_\theta LL_N(\hat{\theta}_N) = 0.$$

Assuming twice continuous differentiability of the log-likelihood, the first term of the Taylor-series expansion of $\nabla_\theta LL_N(\hat{\theta}_N)$ around $\hat{\theta}_N$ gives

$$0 = \nabla_\theta LL_N(\hat{\theta}_N) = \nabla_\theta LL_N(\theta_0) + \nabla^2_{\theta\theta} LL_N(\bar{\theta})(\hat{\theta}_N - \theta_0),$$

where $\bar{\theta}$ is a mean value on the line joining $\hat{\theta}_N$ and θ_0 and $\nabla^2_{\theta\theta}$ denotes the Hessian matrix of the second derivative. Multiplying through by $\sqrt{N}$ and solving for $\sqrt{N}(\hat{\theta}_N - \theta_0)$, we have

$$\sqrt{N}(\hat{\theta}_N - \theta_0) = -[\nabla^2_{\theta\theta} LL_N(\bar{\theta})]^{-1}\sqrt{N}\nabla_\theta LL_N(\theta_0).$$

By the zero mean of score (mentioned above) and the central limit theorem, the term $\sqrt{N}\nabla_\theta LL_N(\theta_0) = \frac{1}{\sqrt{N}}\sum_{i=1}^N g(y_i,\theta_0)$ converges in distribution to $N(0, I(\theta_0))$, where $I(\theta_0) = E_0[g(y,\theta_0)g^T(y,\theta_0)]$, the second moment of the score, also known as the **Fisher information matrix**. Also, since $\bar{\theta}$ is between $\hat{\theta}_N$ and θ_0, it will be consistent if $\hat{\theta}_N$ is, so that by the law of the large numbers, the term $\nabla^2_{\theta\theta} LL_N(\bar{\theta}) = \frac{1}{N}\sum_{i=1}^N \nabla^2_{\theta\theta} \ln f(y_i|\theta_0)$ converges in probability to $H_0 = E_0[\nabla^2_{\theta\theta} \ln f(y|\theta_0)]$. Then the inverse of $\nabla^2_{\theta\theta} LL_N(\bar{\theta})$ converges in probability to H_0^{-1} by continuity of the inverse at a non-singular matrix.

It then follows from the *Slutzky theorem* that

$$\sqrt{N}(\hat{\theta}_N - \theta_0) \xrightarrow{d} N(0, H_0^{-1} I(\theta_0) H_0^{-1}).$$

From the conditions above, we can interchange the order of differentiation and integration for the first and second derivatives operations, and using similar arguments as before, we obtain the well-known information matrix equality:

$$H_0 = E_0[\nabla^2_{\theta\theta} \ln f(y|\theta_0)] = -I(\theta_0).$$

So, in the context of MLE, if the conditions from C1 to C9 hold, the asymptotic normality becomes

$$\sqrt{N}(\hat{\theta}_N - \theta_0) \xrightarrow{d} N(0, I(\theta_0)^{-1}).$$

This expansion shows that the maximum likelihood estimator is approximately equal to a linear combination of the average score in large samples, so that asymptotic normality follows by the central limit theorem applied to the score. This result is the prototype for many other asymptotic normality results. It has several components, including a first-order condition that is expanded around the true parameter, convergence of an inverse Hessian, and a score that follows the central limit theorem. For more details, see Newey and McFadden, Section 3 [22].

One condition that is not essential to asymptotic normality is the information matrix equality. If the distribution is misspecified (i.e we use an approximation instead of $f(y|\theta)$) then the MLE may still be consistent and asymptotically normal. However, if the distribution is misspecified, this will result in the more complicated form $N(0, H_0^{-1} I(\theta_0) H_0^{-1})$. This more complicated form must be allowed to construct a consistent asymptotic variance estimator under misspecification. Moreover, it is often stated that the Fisher information matrix is defined as the opposite of H_0, the Hessian of the log-likelihood. But many popular models do not meet the requirement needed for the information matrix equality. Therefore it is more correct and safer to keep *the second moment of the score* as the definition of the Fisher information matrix.

2.1.3 Asymptotic covariance estimation for MLE

As we discussed above, the distribution of maximum likelihood estimator $\hat{\theta}_N$ tends to a normal distribution

$$\hat{\theta}_N \xrightarrow{d} N[\theta_0, \{I(\theta_0)\}^{-1}].$$

The asymptotic variance of the maximum likelihood estimator is $\{I(\theta_0)\}^{-1}$, which is the inverse of the Fisher information matrix. It can be consistently estimated from $\hat{I}^{-1}$, where $\hat{I}$ is a consistent estimator of the information matrix. Recall that the Fisher information matrix has the form $I(\theta_0) = E_0[g(y, \theta_0) g(y, \theta_0)^T]$. That is, $I(\theta_0)$ is the expectation of the outer product of the score. This form suggests that $I(\theta_0)$ might be estimated by the method of moments, replacing expectations by sample averages and unknown parameter values by estimates

$$\hat{I}_1 = \frac{1}{N}\sum_{i=1}^{N} g(y_i|\hat{\theta}_N)g(y_i|\hat{\theta}_N)^T. \tag{2.3}$$

The consistency of this estimator can be proved by considering the law of large numbers and the consistency of estimator $\hat{\theta}_N$. Moreover, by the information matrix equality, $I(\theta_0) = -E_0[\nabla^2_{\theta\theta} \ln f(y|\theta)]$, the estimator might be estimated by the formula

$$\hat{I}_2 = -\frac{1}{N}\sum_{i=1}^{N} \nabla^2_{\theta\theta} \ln f(y_i|\hat{\theta}_N).$$

The second estimator is just the negative of the Hessian and it will be consistent under the law of large numbers and the consistency of Hessian matrix.

In many cases, the second estimator is rarely available because the second derivatives of the log-likelihood function are complicated or even impossible to calculate. Otherwise the first estimator is just the reciprocal of the sum of squares of the first derivatives. This estimator is extremely convenient in most cases because it does not require any computation beyond the one required to solve the likelihood equation. It has the added virtue that it is always non-negative definite. The estimator in (2.3) is known as the BHHH estimator. It is also the estimator that we consider in our work. This estimator gives us a formula to approximate the Hessian matrix and it can be applied for both the trust-region and line-search methods which will be discussed in the next section.

2.2 Optimization algorithms

In the previous section, we have considered the maximum likelihood (or log-likelihood) estimation problem. The purpose of our work is to propose some effective algorithms to compute the value of MLE. In the context of mathematical programming, we want to maximize the likelihood or log-likelihood function which is non-linear and often non-convex, and in many cases, very complex. The MLE can be expressed as a unconstrained, non-convex, non-linear problem as follows

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

where $f(\theta) = L(\theta)$ or $LL_N(\theta)$ is a general notation of the likelihood or log-likelihood function. For solving this problem, we would like the optimization algorithms to behave in the following manner:

1. They should reliably converge to a local minimizer from an arbitrary starting point.
2. They should do so as quickly as possible.

We note that these optimization algorithms are iterative. They begin with an initial guess of the optimal values of the variables and generate a sequence of improved estimations until they reach a solution. Algorithms which satisfy the first above requirement are called globally convergent, and we say that they use a global strategy (which is different from saying that the algorithm will find a global minimizer). Most strategies make use of the values of the objective function f and possibly the first and second derivatives (Hessian) of these functions. Some effective algorithms require the exact Hessian, which in general is computationally expensive and difficult to program. It is therefore common practice to use an approximation to the Hessian, with the hope of retaining fast local convergence at a lower cost. Selection of a particular Hessian approximation method defines a local strategy. We present in the section below a review of some candidate methods. In this section, assuming that the Hessian or the approximation of Hessian matrix is available, we present two strategies in which the Hessian or its approximation is used as an important factor: **line-search method** and **trust-region method**.

2.2.1 Trust region method

Trust-region method defines a region around the current iterate x_k within which they trust the model to be an adequate representation of the objective function. This region is defined as

$$\mathscr{B}_k = \{x \in \mathbb{R}^n | \, ||x - x_k||_k \leq \Delta_k\} \tag{2.4}$$

where Δ_k is called *the trust-region radius* and $\|\cdot\|_k$ is an iteration-dependent norm. A classical choice is the 2-norm. After defining the trust-region, the trust-region method chooses the step to be the approximate minimizer of the model in this trust-region. Trust-region methods choose the direction and length of the step simultaneously. If a step is not acceptable, they reduce the size of the region and find a new minimizer. In general, the step direction changes whenever the size of the trust-region is altered.

For the trust-region method, we describe a model function $m_k(p)$ at each iteration which is identical to the first two terms of the Taylor-series expansion of f around x_k

$$f(x_k+p) \approx m_k(p) = f(x_k) + \nabla f_k^T p + \frac{1}{2} p^T H_k p \tag{2.5}$$

where ∇f_k^T is the first derivative and H_k is the second derivative of the objective function at point x_k. We also have, from the mean value theorem, that

$$f(x_k+p) = f(x_k) + \nabla {f_k}^T p + \frac{1}{2} p^T \nabla^2 f(x_k + tp) p$$

for some scalar $t \in (0,1)$, and since $m_k(p) = f(x_k) + \nabla f_k^T p + O(||p||^2)$ and $f(x_k+p) = f(x_k) + \nabla {f_k}^T p + O(||p||^2)$, the difference between two values of $m_k(p)$ and $f(x_k+p)$ is $O(||p||^2)$. Therefore the approximation error is also small when p is small. The *trust-region algorithm* is described in **Algorithm 2.1** below using this approximation.

In this description, reasonable parameters of (2.6) are for instance,

$$\eta_1 = 0.9,\ \eta_2 = 0.01,\ \text{and}\ \gamma = 0.5$$

but other values can be selected. In our implementation of this algorithm we have set $\eta_1 = 0.75$, $\eta_2 = 0.01$ and $\gamma = 0.5$. The quadratic model m_k has the form $m_k(p) = f(x_k) + \nabla f_k^T p + \frac{1}{2} p^T H_k p$ where H_k is either the Hessian or some approximation of it. For the MLE problem, the evaluation of the true Hessian is an expensive task, very difficult to compute and to program, so we use an approximation instead. The methods for approximating the Hessian give rise to many approaches for solving the MLE problem that we will discuss in the next section.

Algorithm 2.1 : Basic trust-region (BTR) algorithm

Step 0. Initialization Given an initial point x_0 and an initial trust-region with radius Δ_0. The constants η_1, η_2, γ are also given which satisfy:

$$1 > \eta_1 > \eta_2 > 0 \text{ and } 0 < \gamma < 1 \tag{2.6}$$

Choose an initial matrix H_0 and set $k = 0$,

Step 1. Step calculation Calculate step $p_k \in \mathbb{R}^n$ by solving approximately the problem:

$$\min_{p \in \mathbb{R}^n} \{m_k(p) | x_k + p \in \mathscr{B}_k\}$$

Evaluate ρ_k

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{f(x_k) - m_k(p_k)}. \tag{2.7}$$

If $\rho_k > \eta_2$ then define: $x_{k+1} = x_k + p_k$, otherwise we set $x_{k+1} = x_k$.

Step 2. Trust-region radius update We update the trust-region radius as follow:

$$\Delta_{k+1} = \begin{cases} \max\{2||p_k||, \Delta_k\} & \text{If } \Delta_k \geq \eta_1 \\ \Delta_k & \text{If } \eta_1 > \Delta_k \geq \eta_2 \\ \gamma\Delta_k & \text{If } \Delta_k \leq \eta_2 \end{cases}$$

Set $k \leftarrow k + 1$ and go to step 1.

The main idea of the **trust-region method** is to compare the decrease of the predicted value $m_k(p_k)$ with the actual value of the objective function $f(x_k + p_k)$. If the agreement is sufficiently good, the trial point becomes the new iterate and the trust-region is maintained or enlarged. If this agreement is poor, the trust-region is shrunk in order to improve the quality of the model. The problem

$$\min_{p \in \mathbb{R}^n} \{m_k(p) | x_k + p \in \mathscr{B}_k\} \tag{2.8}$$

is also called the trust-region sub-problem. At each iteration we have to solve (2.8) to obtain the step p_k. The exact minimization of the sub-problem is expensive and often unnecessary, so instead of solving this problem exactly, it is more efficient to solve (2.8) approximately. Many methods have been proposed to compute a p_k. One popular approach is the Steihaug-Toint method (see Conn, Gould, and Toint [10] Section 7.5.1 or Nocedal and Wright [23], page 75).

2.2.2 Line search method

Another broad class of global approaches for solving non-linear unconstrained mathematical problems is the line-search methods. In the line search strategy, the algorithm chooses a direction d_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 d_k can be found by approximately solving the following one-dimensional minimization problem which finds a step length α.

$$\min_{\alpha>0} f((x_k + \alpha d_k)) \tag{2.9}$$

By solving this problem exactly, we would derive the maximum benefit from the direction d_k, but an exact minimization is also expensive and unnecessary. Instead, in a line search method, a limited number of trial step lengths is generated until one is found that loosely approximates the minimum of (2.9). At the new point, a new search direction and step length are computed, and the process is repeated. Each iteration of the line search method computes a search direction d_k and then decides how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k d_k$$

where the positive scalar α_k is called the step length. The success of the line search method depends on effective choices of both the direction d_k and the step length α_k. One effective strategy consists to perform an inexact line search such to identify a step length that achieves adequate reductions in f at minimal cost. Typically, inexact linear search compute step length α_k that satisfies some conditions. The algorithm tries out a sequence of candidate values for α, accepting one of these values when certain conditions are satisfied such as the *Wolfe condition* or the *Goldstein conditions* (Nocedal and Wright [23], p.41).

With line-search, a simple condition that we could impose on α_k is that it provides a reduction in f but in many cases, it is not enough. A popular inexact line search condition stipulates that α_k should first give a *sufficient decrease* in the objective function f, as measured by the following inequality:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T d_k \tag{2.10}$$

for some constant c_1. This condition is also the first condition of the Wolfe conditions, sometimes called the *Armijo condition*. The sufficient decrease condition is not enough by itself to ensure that the algorithm makes reasonable progress, because if it is satisfied for all sufficiently small values of α, it will make the algorithm slow or will never converge. To rule out unacceptably short steps, we introduce a second requirement, called the *curvature condition*, which requires α_k to satisfy the condition

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f_k^T d_k \tag{2.11}$$

for some constant $c_2 \in (c_1, 1)$. We note that this requirement makes sense because if the slope of $\phi'(\alpha) = \nabla_\alpha f(x_k + \alpha d_k)$ is strongly negative, we have an indication that we can reduce f significantly by moving further along the chosen direction. On the other hand, if the slope is only slightly negative or even positive, it is a sign that we cannot expect much more decrease in f in this direction, so it might make sense to terminate the line search.

The *sufficient decrease* and *curvature conditions* are known collectively as the **Wolfe conditions**. Beside the Wolfe conditions, the strong Wolfe conditions can be written as:

$$\boxed{\begin{aligned} f(x_k + \alpha_k d_k) &\leq f(x_k) + c_1 \alpha_k \nabla f_k^T d_k. \\ |\nabla f(x_k + \alpha_k d_k)^T d_k| &\geq c_2 |\nabla f_k^T d_k|. \end{aligned}} \tag{2.12}$$

where c_1, c_2 are two constant satisfying $1 > c_2 > c_1 > 0$. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$. $c_2 = 0.9$ if d_k is chosen by a Newton or quasi-Newton method, and equal 0.1 if d_k is obtained from a non-linear conjugate gradient (Nocedal and Wright p.39 [23]).

Most line search algorithms require d_k to be a descent direction, one for which $d_k^T \Delta f_k < 0$, because this property guarantees that the function f can be reduced along this direction. Moreover, the search direction often has the form:

$$d_k = -H_k^{-1} \nabla f_k$$

where H_k is a Hessian or an approximation of the Hessian matrix, and has to be a symmetric and non-singular matrix. In many cases, if computing the Hessian is an expensive task, an approximation of Hessian H_k is used in place of the true Hessian. The approximate Hessian has to be updated at each iteration using a secant approximation such as BFGS. Methods for approximating the Hessian matrix will be discussed in next section. We also remark that in the context of line-search method, the matrix H_k has to be positive-definite. The numerical optimization algorithm based on linear search is described in **algorithm 2.2**.

Algorithm 2.2: Line-search algorithm

Step 0. Initialization: Given an initial point x_0, an initial Hessian or approximation Hessian matrix H_0 and $k = 0$,

Step 1. Search direction calculation: Compute search direction d_k which satisfies the equation:

$$H_k d_k = -\nabla f(x_k) \tag{2.13}$$

Step 2. Step calculation: Compute α_k which satisfies the Wolfe condition as in (2.12) and obtain the step $s_k = \alpha_k d_k$.
Set $x_{k+1} = x_k + s_k$.
Set $k \leftarrow k+1$, update matrix H_{k+1} and go to step 1.

We note that for the secant approximation, the update will choose H_{k+1} that satisfies the secant condition

$$H_{k+1}(x_{k+1} - x_k) = \nabla f_{k+1} - \nabla f(x_k).$$

Further the matrix H_{k+1} cannot be positive definite if $y_k^T s_k < 0$, because $s_k^T H_{k+1} s_k =$

$y_k^T s_k < 0$. Typically, quasi-Newton methods that use the secant update employ a line search to locate a point for which $y_k^T s_k > 0$. The step length α_k satisfies the Wolfe conditions, therefore we have:

$$\begin{aligned} &\nabla f_{k+1}^T d_k \geq c_2 \nabla f_k^T d_k \\ \Rightarrow\ &(\nabla f_{k+1}^T - \nabla f_k^T) d_k \geq -(1-c_2) \nabla f_k^T d_k \\ \Rightarrow\ &(\nabla f_{k+1}^T - \nabla f_k^T) s_k \geq -(1-c_2) \alpha_k \nabla f_k^T d_k. \end{aligned}$$

As in (2.13), we have $d_k^T \nabla f_k = -d_k^T H_k d_k < 0$. Therefore if H_k is positive definite, we always have $(\nabla f_{k+1}^T - \nabla f_k^T) s_k \geq -(1-c_2) \alpha_k \nabla f_k^T d_k > 0$, which means that the matrix H_{k+1} is positive definite. When paired with the BFGS update, a line search using the Wolfe conditions will produce a positive-definite sequence of matrices $\{H_k\}$.

2.2.3 Stopping conditions

All iterative algorithms need to verify at each iteration whether a stopping condition has been met. A common stopping condition is when an iteration produces a small value of the norm of the gradient, in this case we have a successful execution process. Another popular stopping condition is when an insignificant objective decreases is produced or when the number of iterations is too large or the trust-region radius is too small (in trust-region algorithm), in which case we have an unsuccessful execution process.

The next two stopping conditions are used in our implementation. A first stopping condition used at each iteration is a classical test based on the gradient. The algorithm is terminated as soon as

$$\nabla f(x_k) \leq \varepsilon,$$

where ε is a small constant. We also use a modification of this classical test which is based on the relative gradient (Dennis and Schnabel, 1983 [12], chapter 7). The algorithm can be terminated when

$$\bar{\nabla}(x_k) \overset{\text{def}}{=} \max_{0 \leq c \leq [\text{size of } x_k]-1} \left(\frac{|[\nabla f(x_k)]_c|, \max\{[x_k]_c, 1.0\}}{\max\{|f(x_k)|, 1.0\}} \right) \leq \varepsilon$$

where v_c is c^{th} component of the vector v.

2.3 Hessian approximations

The previous section describes two common classes of methods to optimize non-linear unconstrained functions. In many cases, these methods requires that we compute derivatives of the function that needs to be optimized, which can be an expensive task. For example, in the context of the trust-region and the line-search iterative methods above, an Hessian matrix has to be computed for each iteration. The computation of the Hessian is costly, so we use an approximation instead of true Hessian. The methods selected for approximating the Hessian decide of the behavior of the optimization method. In this section, we describe three methods for approximating the Hessian matrix. The secant approximation and statistical approximation are popular in optimization and maximum likelihood estimation. We also present one special method, called **combined approximation**, which is based on the special structure of the MLE problem.

2.3.1 Statistical approximation

In the context of maximum likelihood estimation, we have presented the BHHH estimator which allows to approximate the Fisher information matrix by the sum of outer products of the scores:

$$\hat{I}_1 = \frac{1}{N}\sum_{i=1}^{N} g(y_i, \hat{\theta}_N) g_(y_i, \hat{\theta}_N)^T$$

where $g(y_i, \theta)$ is the first derivative of the function $\ln f(y_i|\hat{\theta}_N)$, with respect to θ. If the information matrix equality holds, we have another estimator of the information matrix as follows

$$\hat{I}_2 = -\frac{1}{N}\sum_{i=1}^{N} \nabla^2_{\theta\theta} \ln f(y_i|\hat{\theta}_N) = -\nabla^2_{\theta\theta} LL_N(\hat{\theta}_N).$$

Under the conditions of the information equality, both estimators are consistent. This result gives a formula for approximating the Hessian matrix

$$\hat{H}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} g(y_i,\theta)g(y_i,\theta)^T \quad \text{(BHHH)} \tag{2.14}$$

where $\hat{H}(\theta)$ is an approximation of the Hessian matrix evaluated at θ. This approach has been originally proposed by Berndt, Hall, Hall and Hausman [6], as a cheap Hessian approximation technique as we have to evaluate the individual gradients only, and don't have to deal with the second derivative at all.

Another formulation of the BHHH estimator can be obtained by subtracting the mean score before taking the outer product

$$\hat{H}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} (g(y_i,\theta)-\overline{g})(g(y_i,\theta)-\overline{g})^T \quad (BHHH2) \tag{2.15}$$

where $\overline{g} = \frac{1}{N}\sum_{i=1}^{N} g(y_i,\theta)$. The BHHH2 makes sense when the iterative process is not at the maximum, the average score is not zero and $H(\theta)$ does not represent the covariance of the scores (Train [27], p.195).

2.3.2 Secant approximation

The approach can be motivated by observing the quadratic expansion of the objective function around an iterate of the optimization process:

$$m_k(p) = f(x_k) + \nabla f_k^T p + \frac{1}{2}p^T H_k p$$

where H_k is an symmetric matrix that is updated at every iteration. Note that the value and the gradient of this model at $p = 0$ match $f(x_k)$ and $\nabla f(x_k)$ respectively. Suppose that that we have generated a new iterate x_{k+1} and wish to construct a new quadratic model with the new matrix H_{k+1}

$$m_{k+1}(p) = f(x_{k+1}) + \nabla f(x_{k+1})^T p + \frac{1}{2}p^T H_{k+1} p \tag{2.16}$$

such that the gradient of $m_{k+1}(p)$ matches the gradient of the objective function for the latest two iterations x_k and x_{k+1}. We can realize that $\nabla m_{k+1}(0) = \nabla f(x_{k+1})$ precisely when the gradient of m_{k+1} matches the gradient of the objective function at x_{k+1}. For a matching at x_k, we have to have:

$$\nabla m_{k+1}(x_k - x_{k+1}) = \nabla f(x_k) \tag{2.17}$$

Taking derivatives of both sides of (2.16) at $p = x_k - x_{k+1}$ and using (2.17) , we obtain

$$\nabla f(x_k) = \nabla f(x_{k+1}) + H_{k+1}(x_k - x_{k+1}).$$

Rearranging, we obtain

$$H_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k). \tag{2.18}$$

If we define vectors $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, (2.18) becomes

$$H_{k+1}s_k = y_k. \tag{2.19}$$

We refer to this formula as the secant condition and consider two methods constructed based on this formula, called the BFGS method and the SR1 method. With the BFGS method, named from its discoverers Broyden, Fletcher, Goldfarb, and Shanno, the matrix H_{k+1} is updated at each iteration by the formula

$$\boxed{H_{k+1} = H_k - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} + \frac{y_k y_k^T}{y_k s_k} \qquad \text{(BFGS)}} \tag{2.20}$$

This is the fundamental idea of quasi-Newton updating: instead of recomputing the iteration matrices from scratch at every iteration, we apply a simple modification that combines the most recently observed information about the objective function with the existing knowledge embedded in our current Hessian approximation.

In the BFGS updating formula, the updated matrix H_{k+1} differs from its predecessor

H_k by a rank 2 matrix. There is also a simple rank-1 update that maintains symmetry of the matrix and allows it to satisfy the secant equation, called SR1, which is described in the formula below

$$\boxed{H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k} \qquad \text{(SR1)}} \tag{2.21}$$

Note that this update doesn't maintain the positive definiteness and the matrices generated by this formula tend to be very good approximations of the Hessian matrix, often better than the BFGS approximations. This remark was considered a major drawback when we used line search method, but with the advent of trust-region methods, the SR1 updating formula has proved to be quite useful, and its ability to generate indefinite Hessian approximations can actually be regarded as one of its chief advantages, as we can capitalize on negative curvature directions.

2.3.3 Combined approximation

Secant approximation and statistical approximation are well-known approaches in mathematical programming for the case when the exact Hessian is too hard or impossible to compute. Another approximation of the Hessian can be obtained by considering the special structure of the maximum log-likelihood problem as in the work of D.Bunch ([8]). For this, we consider the objective function under generalized regression model:

$$f(\theta) = \rho(R(\theta)), \quad R : \mathbb{R}^p \to \mathbb{R}^N, \quad \rho : \mathbb{R}^N \to \mathbb{R}^1.$$

In the context of the generalized regression, the N components of $R(\theta)$ are the generalized residuals for the N data points. We have $R(\theta) = [r_1(\theta), r_2(\theta), \dots, r_n(\theta)]^T$ where $r_i(\theta)$ is the i^{th} generalized residual and the function $\rho(p)$, $p \in \mathbb{R}^N$, is the sum of N criterion functions $\phi_i(t)$, $t \in \mathbb{R}$. So the function has the form:

$$f(\theta) = \sum_{i=1}^{N} \phi_i(r_i(\theta)).$$

Selection of the specific functional forms for $r_i(\cdot)$, and $\phi(\cdot)$ corresponds to various regression problems. In the MLE problem, if we define $\phi(\cdot) \equiv \ln(\cdot)$ and $r_n(\cdot) \equiv f(y_i|\cdot)$ then we obtain the log-likelihood function under generalized regression:

$$LL_N(\theta) = f(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ln f(y_i|\theta),$$

the first and the second-order development under generalized regression:

$$\nabla f(\theta) = R'(\theta)^T \nabla \rho(R(\theta))$$

$$\nabla^2 f(\theta) = R'(\theta)\nabla^2 \rho(R(\theta))R(\theta) + \sum_{i=1}^{N} \delta_i \rho(R(\theta))\nabla^2 r_n(\theta)$$

and in the context of the MLE problem:

$$\nabla_\theta LL_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{\nabla_\theta f(y_i|\theta)}{f(y_i|\theta)}$$

$$\nabla^2_{\theta\theta} LL_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{\nabla^2_{\theta\theta} f(y_i|\theta)}{f(y_i|\theta)} - \frac{1}{N}\sum_{i=1}^{N} \frac{[\nabla_\theta f(y_i|\theta)][\nabla_\theta f(y_i|\theta)]^T}{f(y_i|\theta)^2}$$

Therefore the Hessian can be written as the sum of two matrices

$$\nabla^2_{\theta\theta} LL_N(\theta) = A + C$$

where

$$A = \frac{1}{N}\sum_{i=1}^{N} \frac{\nabla^2_{\theta\theta} f(y_i|\theta)}{f(y_i|\theta)} \text{ and } C = -\frac{1}{N}\sum_{i=1}^{N} \frac{[\nabla_\theta f(y_i|\theta)][\nabla_\theta f(y_i|\theta)]^T}{f(y_i|\theta)^2}.$$

Moreover, (2.14) shows that $C = -\sum_{i=1}^{N} g_i g_i^T$, where g_i is the first derivative of $f(y_i|\theta)$ or score function, is also the matrix obtained by the BHHH formula. The Hessian matrix is the sum of a matrix C, which is easy to calculate, and the matrix A which requires the calculation of N expensive Hessian matrices. The expensive term A can be approximated by the secant approximation. We present here two approaches to approximate the second term of the Hessian matrix considering the special structure of maximum likelihood estimation.

We suppose that at iteration k, the matrix H_k is available to approximate the next Hessian H_{k+1}. Following (2.18), the new approximation can be obtained by specifying an appropriate secant condition, which takes the form

$$H_{k+1}s_k = y_k \tag{2.22}$$

where H_{k+1} is a new matrix approximation. We can write $H_{k+1} = C + A_{k+1}$ where the matrix C is computed by the BHHH formula and where the second term A_{k+1} is the approximation of A for the next iteration. If we set the matrix $A_{k+1} = 0$ for all iteration k, the approximation becomes the statistical approximation (BHHH).

As discussed in Bunch ([8]), we first consider the secant condition (2.22). The matrix A_{k+1} can then be expressed as

$$A_{k+1}s_k = y_k - Cs_k.$$

So if we set $\bar{y}_k = y_k - Cs_k$, we have a secant equation for updating matrix A_{k+1} - called the *default* secant condition in the terminology of Dennis and Schnabel [12] :

$$A_{k+1}s_k = \bar{y}_k \tag{2.23}$$

Otherwise, we can consider the structure of the matrix A (at iteration k+1)

$$A = \frac{1}{N}\sum_{i=1}^{N} \frac{\nabla^2_{\theta\theta} f(y_i|\theta_{k+1})}{f(y_i|\theta_{k+1})} \tag{2.24}$$

Under the secant approach, we consider each term $\frac{\nabla^2_{\theta\theta} f(y_i|\theta_{k+1})}{f(y_i|\theta_{k+1})}$ and note that

$$\nabla^2_{\theta\theta} f(y_i|\theta_{k+1})(\theta_{k+1} - \theta_k) \approx \nabla_\theta f(y_i|\theta_{k+1}) - \nabla_\theta f(y_i|\theta_k).$$

Substituting into (2.24) gives us

$$As_k \approx \frac{1}{N}\sum_{i=1}^{N}\frac{\nabla_\theta f(y_i|\theta_{k+1}) - \nabla_\theta f(y_i|\theta_k)}{f(y_i|\theta_k)}.$$

If we set $\hat{y}_k = \frac{1}{N}\sum_{i=1}^{N}\frac{\nabla_\theta f(y_i|\theta_{k+1}) - \nabla_\theta f(y_i|\theta_k)}{f(y_i|\theta_k)}$, we obtain another secant equation to approximate the matrix A_{k+1}

$$A_{k+1}s_k = \hat{y}_k \tag{2.25}$$

(2.23) and (2.25) gives us two secant conditions to approximate the Hessian term A. These are two approaches to approximate the Hessian matrix that we call *combined approximation*. We note that the BFGS or SR1 methods can be used in both of the two secant conditions above to obtain these approximations. With the secant equation in (2.23), the matrix A_{k+1} can be estimated by BGFS formula

$$A_{k+1} = A_k - \frac{A_k s_k s_k^T A_k}{s_k^T A_k s_k} + \frac{\bar{y}_k \bar{y}_k^T}{\bar{y}_k s_k}$$

or by SR1 update

$$A_{k+1} = A_k + \frac{(\bar{y}_k - A_k s_k)(\bar{y}_k - A_k s_k)^T}{(\bar{y}_k - A_k s_k)^T s_k}.$$

With the equation (2.25), the formulas is similar but we use $\hat{y}_k = \sum_{i=1}^{N}\frac{\nabla_\theta f(y_i|\theta_{k+1}) - \nabla_\theta f(y_i|\theta_k)}{f(y_i|\theta_k)}$ instead of $\bar{y}_k$.

This section terminates the background of the maximum likelihood estimation and the approaches to solve it. The principle of the problem and the methods for estimating MLE were presented. The MLE problem occurs in many applications. In our work, we focus specifically on discrete choice theory. We implemented our algorithms for the discrete choice problem, especially for the mixed-logit model, as a good framework to test the efficiency of our algorithms. The main concepts and principles of the discrete choice theory will be presented in Chapter 5, but before that, we will present our contributions about the optimal algorithms to estimate the MLE in the next chapter, based on the idea that we can switch between the Hessian approximation methods.

CHAPTER 3

SWITCHING APPROACHES FOR MAXIMUM LIKELIHOOD ESTIMATION

In the previous chapter, we have considered maximum likelihood estimation and some related concepts such as the Fisher information matrix and the information matrix equality. We have also seen that the BHHH method is appropriate given the special properties of the likelihood function and the maximum likelihood estimation. Moreover, based on the special structure of MLE, Bunch [8] has proposed a new method to correct the BHHH approximation using standard secant Hessian approximations, presented in the previous chapter under the name of combined approximation.

The existence of many methods to approximate the Hessian matrix has lead us to consider combining several Hessian approximations in order to obtain a better model at each iteration. Bunch [8] has the first proposed combining several Hessian approximations under the name of *model switching*. Our work develops the generalization of the method proposed by Bunch, in which a set of Hessian approximations is considered. In trust-region methods, this generalization generates a set of sub-problems while for line-search methods it creates a set of search directions. We propose approaches to select the most reasonable sub-problem or search direction for determining the step at each iteration.

In this chapter, we first introduce two new general frameworks based on model switching for trust-region methods and for line-search methods. They are key in the development of our algorithms. Next we describe our own algorithms: *predictive algorithm*, *retrospective algorithm*, *multi sub-problems algorithm* and *multi sub-problems with BHHH algorithm*.

3.1 Model switching

The key idea of model switching is that we can switch between quadratic models at each iteration. Each quadratic model correspond to one Hessian approximation. We consider a set of Hessian approximations, then we obtain the set of corresponding quadratic models.

In the previous chapter, we have introduced three methods to approximate Hessian matrices in the context of maximum likelihood estimation:

– statistical approximation or BHHH approximation.

– secant approximation, either the SR1 and the BFGS formula.

– combined approximation (based on the work of Bunch [8]).

The available methods to approximate the Hessian matrix constitute the set of Hessian approximations. We denote by $\mathscr{H}_k = \{H_k^i, i = 1, 2\ldots\}$ the set of Hessian approximations at iteration k with i indexing the Hessian approximation method. H_{k+1}^i will represent the update of matrix H_k^i produced by one of the approximation methods (BFGS, SR1, BHHH or combined approximation).

For trust-region methods based on a set of Hessian approximations, we denote by $\{m_k^i(p),\, i = 1, \ldots\}$ a set of quadratic models where each model $m_k^i(p)$ is defined by

$$m_k^i(p) = f(x_k) + \nabla f_k^T p + \frac{1}{2} p^T H_k^i p, \quad H_k^i \in \mathscr{H}_k. \tag{3.1}$$

The set of quadratic models yield a set of sub-problems $\{\min_{p \in \mathscr{B}_k} m_k^i(p),\, H_k^i \in \mathscr{H}_k\}$.

On the other hand, for line-search methods, the search direction is computed by calculating the vector d_k satisfying the condition $H_k d_k = -\nabla f(x_k)$ where H_k is an Hessian matrix approximation. The set of Hessian approximations provides a set of search directions $\{d_k^i,\, i = 1, 2, \ldots\}$, where each element satisfies the condition

$$H_k^i d_k^i = -\nabla f(x_k), \quad H_k^i \in \mathscr{H}_k. \tag{3.2}$$

The general form of model switching is described by **Algorithm 3.1** for trust-region methods and by **Algorithm 3.2** for line-search methods.

Obviously, a key issue in these two algorithms is the identification for step 1 of a method to select *the best Hessian matrix approximation*. We do not have a direct answer to this question but we propose some approaches to predict which matrix is better based

```
Algorithm 3.1 : General model switching algorithm (trust-region methods)
  Step 0. Initialization Given an initial point $x_0$ and an initial trust-region with
      radius $\Delta_0$. The constants $\eta_1$, $\eta_2$, $\gamma$ are also given as in Algorithm 2.1.
      Choose an initial set of matrices $\mathscr{H}_0$ and set $k = 0$.
  Step 1. Step calculation Consider a set of models $\{m_k^i(p)\}$, where each model
      $m_k^i(p)$ is defined as in (3.1). Select only one model $m_k^{i^*}(p)$ and solve ap-
      proximately the corresponding sub-problem
          $\min_{p \in \mathscr{B}_k} m_k^{i^*}(p)$
      to obtain step $p_k^{i^*}$.
      Evaluate $\rho_k$
          $\rho_k = \frac{f(x_k) - f(x_k + p_k^{i^*})}{f(x_k) - m_k(p_k^{i^*})}$.
      If $\rho_k > \eta_2$ then define: $x_{k+1} = x_k + p_k^{i^*}$, otherwise set $x_{k+1} = x_k$.
  Step 2. Trust-region radius update Identical to step 2 of Algorithm 2.1.
```

on the properties of the quadratic model. For trust-region methods, we note that the model $m(p)$ is defined based on a Taylor-serie expansion of f around x, which is also the prediction of the objective function around x. Thus we *define* the best model as the model that is the closest to the objective function, i.e, the model $m^{i^*}(p)$ where i^* minimizes the different between the quadratic function and the objective function

$$i^* = \arg\min_i |m^i(p) - f(x+p)|.$$

Another possible model choice is to select the one giving the best prediction of the objective function decrease.

For line-search methods, the quadratic model $m(d)$ is also used to predict the objective function. The search direction d can be computed by minimizing the quadratic function. Thus we take the derivative of $m(d)$ and set it equal to zero to find the solution.

$$\nabla_d m(d) = 0.$$

```
Algorithm 3.2 : General model switching algorithm (line-search methods)
  Step 0. Initialization: Given an initial point $x_0$, an initial set of Hessian approximation matrices $\mathscr{H}_0$ and set $k = 0$.
  Step 1. Search direction calculation: Compute the set of search directions $\{d_k^i\}$ as in (3.2). Search direction $d_k^{i^*}$ is somehow deemed to be better than the other ones and can be chosen to define the step.
  Step 3. Step calculation: Compute $\alpha_k$ by solving approximately the subproblem
          $\min_{\alpha>0} f(x_k + \alpha d_k^{i^*})$
      Set $x_{k+1} = x_k + \alpha_k d_k^{i^*}$ and go to step 1.
```

But $m(d) = f(x) + \nabla f^T d + \frac{1}{2} d^T H d$, so we have $\nabla f^T + d^T H = 0$. This give the equation to calculate the search direction d.

$$d^T H = -\nabla f^T.$$

The real step is obtained by searching along the search direction $s = \alpha d$. Like trust-region methods, the search direction can be selected by minimizing the difference between the quadratic function and the objective function, i.e, such that i^* minimizes the value $|m^i(s) - f(x+s)|$.

Our purpose is to apply algorithms 3.1 and 3.2 to complex functions, so we have to avoid computing the value of the objective function or its derivatives many time, otherwise our algorithms will be slow. A predictive algorithm and a retrospective algorithm are developed, which use the available information and try to avoid calculating more than one objective function and its derivative at each iteration. The multi sub-problems algorithm is another approach in which we compute more than one step and compare the decrease in the objective function. This algorithm requires computing the objective function more than once at each iteration, but we can show that the number of needed iterations is much smaller when compared with other approaches. Among the approximations of the Hessian matrix, statistical approximation has proved its efficiency at the beginning of the iterative process. Therefore we can start with BHHH approximation and switch to multi sub-problem when the current point is close enough to the solution. This is also the idea to improve the multi sub-problems approach, presented in the section below.

3.2 Predictive model

The idea of predictive model is to use available information from the current iteration to select an approximation of the Hessian in the next iteration.

For trust-region algorithms, the step p_k is computed by approximating the solution of the sub-problem $\min_{p\in\mathscr{B}_k} m_k(p)$. If this step is not accepted, the trust-region is reduced and the sub-problem is solved again, otherwise this step can be used to select the Hessian approximation for the next iteration. Recall that we use a quadratic model to approximate the objective function:

$$f(x_k+p) \approx m_k^i(p) = f(x_k) + \nabla f_k^T p + \frac{1}{2} p^T H_k^i p.$$

We denote by $\delta_k^i(p)$ the approximation error of the quadratic function $m_k^i(p)$, which is defined as follows

$$\begin{aligned} \delta_k^i(p) &= |m_k^i(p) - f(x_k+p)| \\ &= |f(x_k+p) - f(x_k) - p^T \nabla f_k - \frac{1}{2} p^T H_k^i p| \end{aligned} \tag{3.3}$$

where $H_k^i \in \mathscr{H}_k$ is one Hessian approximation. We assume that the best Hessian approximation minimizes the approximation error of the quadratic function. So we can predict the best Hessian approximation for the next iteration by finding i^* which satisfies:

$$i^* = \arg\min_i |\delta_k^i(p_k)|. \tag{3.4}$$

In case we apply the predictive model for line-search, the index i^* satisfies

$$i^* = \arg\min_i |\delta_k^i(s_k)|. \tag{3.5}$$

where $s_k = \alpha_k d_k$ is the step at the current iteration. We note that, for evaluating the approximation errors $\delta_k^i(p_k)$ or $\delta_k^i(s_k)$, we use the computed objective functions. Thus we don't have to calculate more than one objective function at each iteration.

Moreover, in the context of trust-region method, if $\rho_k \geq \eta_2$, the iteration k is said to be successful since the candidate point $x_k + p_k$ is accepted, otherwise the iteration is declared unsuccessful and the new point is rejected. Moreover, if $\rho_k \geq \eta_1$, the agreement between the model and the function is particularly good, so the iteration is said to be very successful. If $\rho_k \leq \eta_2$, the iteration is said to have failed. For trust-region algorithms, we expect the iteration to be very successful and to increase the trust-region, otherwise we have to keep or reduce the trust-region to obtain a bigger agreement ρ_k. Thus the agreement ρ_k is said to be used to predict the next trust-region.

In view of model switching, it seems to be reasonable to use the value of ρ_k to select the Hessian approximation of the next iteration, which gives us another approach to obtain the next Hessian approximation $H_{k+1}^{i^*}$ in the context of predictive model. Let

$$\rho_k^i = \frac{f(x_k) - f(x_k + p_k)}{f(x_k) - m_k^i(p_k)},$$

and let $\{\rho_k^i | H_k^i \in \mathscr{H}_k\}$ be a set of agreements. The Hessian approximation $H_{k+1}^{i^*}$ is found by solving

$$i^* = \arg\max_i \rho_k^i. \tag{3.6}$$

The details of the predictive algorithm are given in **Algorithm 3.3**.

3.3 Retrospective model

In the predictive model, the prediction of the Hessian approximation H_k for the next iteration occurs at the end of iteration $k-1$ given that the value of the objective function $f(x_{k-1} + p_{k-1})$ has been already calculated. The predictive algorithm uses this value to evaluate the accuracy of the quadratic model m_{k-1} around x_{k-1}. But this might seem unnatural since the Hessian approximation H_k is used to determine the model m_k, not the previous model m_{k-1}. A more reasonable approach consist to determine the matrix

```
Algorithm 3.3: Predictive algorithm
At iteration $k$:
(we note that the best Hessian approximation $H_k$ is determined at the previous iteration).
  2. Trust-region method.
      2.1 Step calculation:
          Evaluate the step $p_k$ by solving approximately the sub-problem
              $\min_{x_k+p \in \mathscr{B}_k} m_k(p).$
          Evaluate $\rho_k$
              $\rho_k = \frac{f(x_k) - f(x_k + p_k)}{f(x_k) - m_k(p_k)}.$
          If $\rho_k > \eta_2$ set $x_{k+1} = x_k + p_k$, otherwise set $x_{k+1} = x_k$.
      2.2 Predict the Hessian approximation:
          The next Hessian approximation $H^{i^*}_{k+1}$ is predicted by solving (3.4) or
          (3.6). Set $H_{k+1} = H^{i^*}_{k+1}$ (to be used at next iteration).
      2.3 Trust-region radius update:
          Identical to step 2 of Algorithm 2.1.
  3. Line-search method.
      3.1 Search direction calculation:
          Identical to step 1 of Algorithm 2.2
      3.2 Step calculation:
          Compute step length $\alpha_k$ which satisfies the Wolfe conditions and set
          $x_{k+1} = x_k + \alpha_k d_k$.
      3.3 Predict the Hessian approximation: Predict the next Hessian approxi-
          mation $H^{i^*}_{k+1}$ by (3.5). Set $H_{k+1} = H^{i^*}_{k+1}$ (to be used at next iteration).
```

H_k at the beginning of iteration k, by considering the quadratic model m_k. To avoid computing more than one objective value at each iteration, the retrospective algorithm use the available objective value at the previous iteration to evaluate the model m_k.

We evaluate the approximation error (defined in (3.3)) at the point $-p_h$ (where $h < k$ is the largest successful iterate before iteration k, $p_h \neq 0$) by

$$\delta^i_k(-p_h) = |f(x_h) - f(x_k) + p_h^T \nabla f_k - \frac{1}{2} p_h^T H^i_k p_h|.$$

The calculation of $\delta_k^i(-p_h)$ does not require any new calculations of the objective function. And finally the matrix $H_k^{i^*}$ can be obtained by minimizing the approximation error

$$i^* = \arg\min_i \delta_k^i(-p_h) \tag{3.7}$$

for trust region algorithms or

$$i^* = \arg\min_i \delta_k^i(-s_{k-1}) \tag{3.8}$$

for the line-search algorithms.

In the classical framework of the trust-region algorithms, the trust-region radius is updated at the end of each iteration. The ratio ρ_k is used to predict the trust-region radius for the next iteration. Bastin et al. [4] propose the *retrospective algorithm* in which the trust-region radius is updated after each successful iteration $k-1$ (that is at the beginning of iteration k) on the basis of the retrospective ratio $\tilde{\rho}_k$ which is defined as follows

$$\tilde{\rho}_k = \frac{f(x_k) - f(x_{k-1})}{f(x_k) - m_k(-p_{k-1})}.$$

In the context of model switching, we also defined the set of retrospective ratios $\{\tilde{\rho}_k^i, 1 = 1,2,\dots\}$ where each element is determined as follows

$$\tilde{\rho}_k^i = \frac{f(x_k) - f(x_h)}{f(x_k) - m_k^i(-p_h)}$$

where $h < k$ is the largest successful iterate before iteration k. Based on the role of retrospective ratios in the *retrospective algorithm*, the Hessian approximation can be determined by choosing the index i^* which maximizes the retrospective ratio

$$i^* = \arg\max_i \tilde{\rho}_k^i \tag{3.9}$$

which provides another method to select the Hessian approximation for the current iteration. Equations (3.7) and (3.9) provide two approaches to select the Hessian approximation in the context of model switching, and it is straightforward to show the two approaches are not equivalent. The retrospective algorithm is described in **Algorithm 3.4**.

```
Algorithm 3.4 : Retrospective algorithm
At iteration $k$:
  1. Define the set of Hessian approximations $\mathscr{H}_k$.
  2. Trust-region method:
      2.1 Step calculation:
        Select the Hessian approximation $H_k^{i^*}$ by solving (3.7) or (3.9).
        Calculate the step $p_k$ by solving approximately the sub-problem
            $\min_{x_k+p\in\mathscr{B}_k} m_k^{i^*}(p).$
        Evaluate $\rho_k$
            $\rho_k = \frac{f(x_k)-f(x_k+p_k)}{f(x_k)-m_k(p_k)}.$
        If $\rho_k > \eta_2$ set $x_{k+1} = x_k + p_k$, otherwise set $x_{k+1} = x_k$.
  2.2 Trust-region radius update:
      Identical to step 2 of Algorithm 2.1.
  3. Line-search method:
      3.1 Search direction calculation:
        Select the Hessian approximation $H_k^{i^*}$ by solving (3.8).
        The search direction $d_k$ satisfies $H_k^{i^*} d_k = \nabla f_k$
      3.2 Step calculation:
        Identical to step 2 of Algorithm 2.2
```

3.4 Multi sub-problem model

Each iteration in trust region methods defines a sub-problem. Solving approximately this sub-problem determines the current step. In the context of model switching, at each iteration there is a set of sub-problems:

$$\min_{p\in\mathscr{B}_k} m_k^i(p) = \min_{p\in\mathscr{B}_k} \{f(x_k) + \nabla f_k^T p + \frac{1}{2} p^T H_k^i p\}, \quad H_k^i \in \mathscr{H}_k.$$

For trust region methods, we solve (approximately) the sub-problem to get the step. It is natural to consider a step as *good* if it decreases significantly the objective function. This leads to a way to solve approximately all the available sub-problems in order to obtain the set of steps p_k^i and to choose the step which minimizes the objective function. We call this approach the *multi sub-problems algorithm*. The step $p_k^{i^*}$ is chosen if it satisfies:

$$i^* = \arg\min_i f(x_k + p_k^i) \tag{3.10}$$

or

$$i^* = \arg\max_i |f(x_k + p_k^i) - f(x_k)|. \tag{3.11}$$

When the predictive and retrospective algorithm choose the sub-problem by evaluating how well the quadratic model predict the objective function, the multi sub-problem algorithm has a more natural approach since the sub-problem is chosen by evaluating the decreasing of the objective function made by the steps. However this algorithm requires solving the multi sub-problem and calculating more than one objective function at each iteration. It violates one of the purposes of the switching model mentioned above. In some cases, the algorithm can be slower than the two previous algorithms, but it requires less iterations and it converges in some difficult cases (where retrospective and predictive algorithms cannot converge). Details of the results will be presented in next chapter with some real data of choice model. **Algorithm 3.5** describes the *multi sub-problems algorithm*.

3.5 Multi sub-problem with the BHHH model

The algorithm that uses BHHH approach to update the Hessian approximation, can reach very fast the neighbourhood of solutions, but often it does not converge (as shown in next chapter). On the other hand, the multi sub-problems algorithm has good convergence, it requires less iterations than other approaches but it is slow because it computes more than one objective value at each iteration. At the beginning of the iterative process, if we use the BHHH approach instead of multi sub-problems, we can avoid computing

```
**Algorithm 3.5** : Multi sub-problems algorithm
At iteration $k$:
  1. Define the set of Hessian approximation $\mathscr{H}_k$.
  2. Step calculation:
       Calculate the set of steps $\{p_k^i,\ i = 1,2\dots\}$ by solving approximately all the
       sub-problems.
           $\min_{p \in \mathscr{B}_k} \{m^i(p),\ H_k^i \in \mathscr{H}_k\}.$
       Determine the best step $p_k^{i^*}$ by solving (3.10) or (3.11).
       Compute the ratio $\rho_k$
           $\rho_k = \frac{f(x_k) - f(x_k + p_k^{i^*})}{f(x_k) - m_k(p_k^{i^*})}.$
       If $\rho_k > \eta_2$ set $x_{k+1} = x_k + p_k$, otherwise set $x_{k+1} = x_k$.
  3. Trust-region radius update: Identical to step 2 of Algorithm 2.1.
```

unnecessary objective values. From this analysis, we present an improvement of the multi sub-problems approach which combines the BHHH method and the multi sub-problems approach. The algorithm uses the BHHH method for the first iterations, but when the current point is in the neighborhood of the solution, it switches to the multi sub-problems algorithm.

The decision to switch from the BHHH method to the multi sub-problems method has an impact on the algorithm, but it is difficult to determine when a point is in the neighborhood of the solution. We consider two approaches. One is that we determine that a point is in the neighborhood of the solution if the norm of the gradient at this point is small. A second one considers the length of the step, if $\max\{||p_u||, ||p_v||\} < \varepsilon$ where p_u and p_v is two successive successful steps, and ε is a small constant. **Algorithm 3.6** describes the *Multi sub-problems with BHHH algorithm.*

Note that switching in multi sub-problems with BHHH occurs only one time, i.e, if the algorithm switch to multi sub-problems algorithm it will never switch back to the BHHH approach.

```
**Algorithm 3.6** : Multi sub-problems with BHHH algorithm
Give a constant $\varepsilon > 0$.
At iteration $k$:
  1. Step calculation:
      1. If $\max\{||p_u||, ||p_v||\} > \varepsilon$, the Hessian approximation $H_k$ is computed by
         the BHHH formula. The rest of the step is identical to step 1 of Algorithm 2.1.
      2. If $\max\{||p_u||, ||p_v||\} \leq \varepsilon$, the step $p_k$ is computed by multi sub-problem
         approach, identical to step 2 of Algorithm 3.5.
  2. Trust-region radius update: Identical to step 2 of Algorithm 2.1.
```

CHAPTER 4

ADAPTIVE LINE SEARCH

The previous chapter has introduced new algorithms inspired from model switching. We now describe our new line-search algorithm called *adaptive line-search*. This algorithm is developed from the idea that we can adapt the length of search directions to obtain better steps. Adaptive line-search algorithm is described in this chapter as a new optimization algorithm to solve MLE.

In line-search methods, the step length needs to be computed at each iteration. The following sub-problem needs to be solved at each iteration

$$\min_{\alpha>0} \phi_k(\alpha) = f(x_k + \alpha d_k)$$

where d_k is the search direction, which can be computed by the formula $H_k d_k = \nabla f(x_k)$. In computing the step length α_k, we would like to choose α_k such to obtain a substantial reduction of f, but at the same time, we do not want to spend too much time making this choice. The ideal choice would be the global minimizer of the univariate function $\phi_k(\alpha)$. But in general, it is too expensive to identify this value. Finding even a local minimizer of ϕ_k with a moderate precision generally requires too many evaluations of the objective function f and possibly its derivatives. 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 a sequence of candidate values for α and accept one of these values when certain conditions are satisfied (i.e the *Wolfe conditions*, described in previous chapter). A popular inexact line-search condition stipulates that α_k should first give a *sufficient decrease* in the objective function f

$$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T d_k. \tag{4.1}$$

But the *sufficient decrease* condition is not enough by itself to ensure that the algorithm makes reasonable progress as it can be satisfied for all sufficiently small values of α. To rule out unacceptably short steps, we consider a second condition, called the *curvature condition*, which requires α_k to satisfy

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f_k^T d_k. \tag{4.2}$$

The iterative process to compute the step length which satisfies the *Wolfe conditions* (more simply we can call it *line-search procedure*) requires an initial estimate α_k^0 and generate a sequence $\{\alpha_k^i\}$ that either terminates with a step length α_k satisfying the conditions or determines that such a step length does not exist. A typical line-search procedure consists of two phases: a bracketing phase that finds an interval $[\alpha_k^i, \alpha_k^{i+1}]$ containing acceptable step lengths, and a selection phase that zooms in to locate the final step length α_k. The selection phase usually reduces the bracketing interval during its search for the desired step length and interpolates some of the function and derivative information gathered on earlier steps to guess the location of the minimizer (see Nocedal and Wright [23], Chapter 2).

The first phase of line-search procedure generates a sequence of trial step lengths $\{\alpha_k^i,\ i = 0, 1, \ldots\}$ which is monotonically increasing. The procedure uses the knowledge that the interval $(\alpha_k^i, \alpha_k^{i+1})$ contains step lengths satisfying the Wolfe conditions if α_k^{i+1} violates the *sufficient condition*, or $\phi_k(\alpha_k^{i+1}) \geq \phi_k(\alpha_k^i)$, or $\nabla_\alpha \phi_k(\alpha_k^{i+1}) \geq 0$. From the interval $(\alpha_k^i, \alpha_k^{i+1})$ generated by the first phase, the second phase define the interval $(\alpha_k^{\text{lo}}, \alpha_k^{\text{hi}})$ which starts by $(\alpha_k^i, \alpha_k^{i+1})$. Each iteration of the second phase generates $(\alpha_k)_j$ between $(\alpha_k^{\text{lo}}, \alpha_k^{\text{hi}})$ then replaces these endpoints by $(\alpha_k)_j$. This process stop at the value α_k that satisfies the curvature condition.

For the line-search procedure, an initial step length $\alpha_k^0 = 1$ is usually be used as the initial trial step. The iterative process starts generating the sequence step $\{\alpha_k^i d_k,\ i = 0, 1, \ldots\}$ from the step $\alpha_k^0 d_k$, which should usually be equal to d_k. But if the length of d_k is too large or too small, the iterative process could start from an unreasonable starting point and the algorithm could be unstable (the numerical results in the next chapter will show that). A solution for this problem is to normalize the search direction by the formula

$$d_k \leftarrow \frac{d_k}{||d_k||}.$$

Then the length of the search direction is fixed, equal to 1. However, the line-search procedure is an expensive task. The iterative process at each iteration requires many calculations of the objective function and its derivatives. The line-search procedure usually limit the number of iterations. Consequently, sometime, the procedure cannot find a step length which satisfies the Wolfe condition. Thus, the initial step of a line-search procedure $\alpha_k^0 d_k$ is important and is expected to be closed to the final step $\alpha_k d_k$. Obviously, it is difficult to estimate whether the initial step satisfies this expectation without computing several values of objective function f and its derivatives. Rather, we propose a scale of search-direction, called *line-search scale* Δ_k, which multiplies with the search direction,

$$d_k \leftarrow \Delta_k d_k.$$

The scale is adaptive at each iteration. The behaviour of the adapting Δ_k is as follows: if the step length of the current iteration is greater than the one of the previous iteration, the line-search scale Δ_k is increased. Otherwise it is decreased. From that we propose a new algorithm, called **adaptive line-search algorithm**. The details of this algorithm are described in **Algorithm 4.1**.

Algorithm 4.1: Adaptive Line-search algorithm

```
Step 0. Initialization: Given the starting point $x_0$, an initial Hessian approximation $B_0$, an initial line-search scale $\Delta_0$ and $k = 0$. Given two constant $\gamma_1$, $\gamma_2$ satisfying
        $\gamma_1 > 1 > \gamma_2 > 0$
Step 1. Search direction calculation: Compute the search direction by finding $d_k$ that satisfies the equation:
        $H_k d_k = -\nabla f(x_k)$
        where $H_k$ is a Hessian approximation matrix.
Step 2. Step normalization: Set $d_k \leftarrow \frac{\Delta_k . d_k}{||d_k||}$.
Step 3. Step calculation: Compute $\alpha_k$ that satisfies the Wolfe condition in (4.1) and in (4.2). Note that the initial trial step length $\alpha_k^0 = 1$.
        Set $x_{k+1} = x_k + \alpha_k d_k$.
Step 3. Update line-search scale: Set $\mu_k = ||\alpha_k d_k||$.
        $\Delta_{k+1} = \begin{cases} \min\{\gamma_1 \Delta_k, \mu_k\} & \text{if } \Delta_k \leq \mu_k \\ \max\{\gamma_2 \Delta_k, \mu_k\} & \text{if } \Delta_k > \mu_k \end{cases}$
        Set $k \leftarrow k + 1$ and go to step 1.
```

CHAPTER 5

DISCRETE CHOICE THEORY

Discrete choice problems have been of interest to researchers for many years in a variety of disciplines. Examples of many possible applications can be found in mathematical psychology (Luce [16]), in marketing (see McFadden and Train [18]), in transportation (Sheffi [26], Chapter 10) and econometric studies (see McFadden [17]).

A discrete choice model is one in which decision makers choose among a set of alternatives to fit within a discrete choice framework. The decision makers can be people, households, firms, or any other decision-making unit, and the alternatives might represent competing products, courses of action, or any other options or items over which choices must be made. Following the framework given by Ben Akiva and Lerman [5], we consider the choice as the outcome of a sequential decision making process, which includes 4 steps: i) Definition of the choice problem, ii) generation of alternatives; iii) evaluation of attributes of the alternatives iv) choice and implementation. We describe more clearly this decisional process in the context of the discrete choice theory below.

5.1 Decision-maker

A decision-maker in discrete choice theory is assumed to be an individual who makes decisions. The concept of individual may be extend, depending on the particular application, it may be one person or it may be a group of persons (for example a household). The internal decisions within the group are then ignored and we only consider the decisions of the group as a whole. We refer to decision-maker and individual interchangeably. We denote by I the population size or number of individuals.

5.2 The alternatives

The alternatives are the objects that are chosen by decision-makers. The set containing these alternatives is called the *choice set*. In discrete choice, the choice set needs to exhibit three characteristics: alternatives need to be mutually exclusive, alternatives must be exhaustive and the number of alternatives must be finite. Therefore, a discrete choice set, which we denote by $\mathscr{A}$, contains a finite number of alternatives that can be explicitly listed.

Two concepts of choice set are considered: the *universal choice set* and the *reduced choice set*. While the universal choice set contains all potential alternatives in the context of the application, the reduced choice set on the other hand is a subset of a universal choice set as observed by some particular individuals. Alternatives in the universal choice set that are not available to the individual under consideration are excluded. In our research, the choice set refers to the reduced choice set. We denote $\mathscr{A}(i) \in \mathscr{A}$ the set of alternatives available for individual i ($i = 1, 2, \ldots, I$).

5.3 Attributes

Each alternative in the choice set is characterized by a set of attributes. The attributes may be generic to all alternatives or may be specific to only one alternative. An attribute is not necessarily a directly observed quantity. It can be any function of the available data, and it depends on the particular application.

5.4 Utilities and Decision rule

In this section, we define the notions of *utilities* and *decision rule* which are used to describe the behavior of decision-makers. For each individual i, each alternative available $A_j \in \mathscr{A}(i)(j = 1, 2, \ldots, |\mathscr{A}(i)|)$ has an associated utility U_{ij}, which depends on the individual characteristics and the relative attractiveness of the alternative. Here we focus on random utility models since they constitute the most common framework for generating discrete choice models.

In random utility models, each alternative has some probability to be chosen by an individual. Each probability is modelled as a function of the socio-economic characteristics of the individual and the relative attractiveness of the alternative. The utility U_{ij} is a random variable assumed to have the form:

$$U_{ij} = V_{ij} + \varepsilon_{ij} \tag{5.1}$$

where $V_{ij} = \phi(\beta_j, x_{ij})$ is a function of a vector β_j to be estimated and a vector x_{ij} containing all the attributes of alternative j. ε_{ij} is a random term representing the unobserved part of the utility, reflecting the idiosyncrasies and particular tastes of each individual. We can also think of V_{ij} as the systematic component of a decision maker's utility and ε_{ij} as the stochastic component. A popular and simple expression for $V_{ij}(j = 1, 2 \ldots, |\mathscr{A}(i)|)$ is the linear utility:

$$\phi(\beta_j, x_{ij}) = \beta_j^T x_{ij} = \sum_{k=1}^{K_j} \beta_j^k x_{ij}^k \tag{5.2}$$

where K_j is the number of observed attributes for alternative $A_j(j = 1, \ldots, |\mathscr{A}(i)|)$. The parameter vector $\beta_j(j = 1, \ldots, |\mathscr{A}(i)|)$ is assumed to be constant for all individuals but may vary across alternatives. Linearity simplifies the formulation and the estimation of the model, but the non-linear effects can still be captured in the attributes definitions, as a function of available data.

A random utility model assume that the decision-maker belongs to a given homogeneous population, acts rationally and has a perfect discrimination capability. The decision-maker chooses the alternative with the highest utility by choosing alternative j if and only if $U_{ij} \geq U_{it} \quad \forall t \neq i, t = 1, 2 \ldots |\mathscr{A}(i)|$. The analyst cannot observe the utility of decision maker but can observe some attributes x_{ij} of the alternatives and some attributes of the decision maker, labelled β_i. The analyst can also specify a function that relates these observed factors to the decision maker's utility. This function $V_{ij} = V(x_{ij}; \beta_n) \forall j$ is also called representative utility.

The derivation of random utility models is based on a specification of utility as defined above. The decision rules then assume that individual i selects the alternative that maximizes its utility. In other terms the individual chooses A_j if and only if

$$U_{ij} \geq U_{it} \quad \forall t \neq i, t = 1, 2 \dots |\mathscr{A}(i)| \tag{5.3}$$

On the derivation of random utility, (5.3) equivalent with:

$$V_{ij} + \varepsilon_{ij} \geq V_{it} + \varepsilon_{it} \quad \forall t \neq i, t = 1, 2 \dots |\mathscr{A}(i)|$$

The researcher does not know ε_{ij}, and therefore treats these terms as random. The joint density of the random vector ε_{ij} is denoted $f(\varepsilon_{ij})$. With this density, the analyst can make probability statements about the choice of the decision maker. In other words, the probability that decision maker i choose alternative j is simply:

$$\begin{aligned} P_{ij} &= Prob[U_{ij} > U_{it}, \forall t \neq j] \\ &= Prob[V_{ij} + \varepsilon_{ij} > V_{it} + \varepsilon_{it}, \forall t \neq j] \\ &= Prob[V_{ij} - V_{it} > \varepsilon_{it} - \varepsilon_{ij}, \forall t \neq j] \end{aligned}$$

We realize that this probability is a cumulative distribution, the probability that each random term $\varepsilon_{ij} - \varepsilon_{it}$ is below the observed quantity $V_{ij} - V_{it}$. Using the density $f(\varepsilon_i) = f(\varepsilon_{i1}, \varepsilon_{i2}, \dots, \varepsilon_{i|\mathscr{A}(i)|})$, this cumulative probability can be written as:

$$\begin{aligned} P_{ij} &= Prob[V_{ij} - V_{it} > \varepsilon_{it} - \varepsilon_{ij}, \forall t \neq j] \\ &= \int_{\mathbb{R}^{|\mathscr{A}(i)|}} I(V_{ij} - V_{it} > \varepsilon_{it} - \varepsilon_{ij}, \forall t \neq j) f(\varepsilon_i) d\varepsilon_i \end{aligned} \tag{5.4}$$

where $I(V_{ij} - V_{it} > \varepsilon_{it} - \varepsilon_{ij}, \forall t \neq j)$ is the indicator function, equalling 1 when the expression in parentheses is true and 0 otherwise. This is a multidimensional integral the density function $f(\varepsilon_i)$ of the unobserved portion of the utility. This integral takes a closed form only for certain specifications of density function $f(\cdot)$. The actual form of the distribution of the residual ε_{ij} leads to different families of models. The next sub-section presents some common random utility models.

5.5 Random utility models

Probit, logit, nested-logit and mixed-logit are the common models described in this section. **logit** and **nested-logit** have closed-form expressions for the integral. They are derived under the assumption that the unobserved portion of the utility is i.i.d. extreme value of type, also called Gumbel distribution, and a type of generalized extreme value (GEV), respectively. Another model, **probit** is derived under the assumption that $f(\cdot)$ is a multivariate normal and **mixed-logit** is derived under the assumption that the unobserved portion of the utility comprises a part that follows any distribution desired by the analyst and a part that is i.i.d extreme value. The integral of probit and mixed-logit have no closed form solutions, we have to evaluate them numerically through simulation. The details of these models is presented below.

Probit model or the multinomial Probit model is derived from the assumption that the random vector $\varepsilon_i = (\varepsilon_{i1}, \ldots, \varepsilon_{iJ})^T$ is multivariate normal distributed with a vector mean μ_ε and a $J \times J$ variance-covariance matrix Σ_ε. The probit model is motivated by the central limit theorem, assuming that the error terms are the sum of independent unobserved quantities. With probit, the probability function (5.4) has no closed analytical form which is the main limitation of this model.

Logit model is derived from assumption that the residuals ε_{ij} are independent and identically Gumbel distributed with mean 0 [1] and scale factor μ, the probability that the individual i chooses the alternative $A_j \in \mathscr{A}(i)$ can be expressed by the expression

$$P_{ij} = \frac{e^{\mu V_{ij}}}{\Sigma_{m=1}^{|\mathscr{A}(i)|} e^{\mu V_{im}}} \tag{5.5}$$

μ is often set to 1, leading to the standard Gumbel distribution. We note that with Gumbel distributed, the density for each unobserved component of the utility is

$$f(\varepsilon_{ij}) = \frac{1}{\mu} e^{-\frac{\varepsilon_{ij}-\alpha}{\mu}} e^{-e^{-\frac{\varepsilon_{ij}-\alpha}{\mu}}}$$

1. More generally, any constant mean can be assumed, as long as it is equal among the alternatives.

where α is the mode of the Gumbel, and the cumulative distribution is

$$F(\varepsilon_{ij}) = e^{-e^{-\frac{\varepsilon_{ij}-\alpha}{\mu}}}.$$

The variance of this distribution is $\frac{\pi^2}{6}\mu^2$ and the mean is $\alpha + \alpha\gamma$, where γ is the Euler-Mascheroni constant.

The difference between two extreme value variables is distributed logistic. That is, if ε_{ij} and ε_{it} are i.i.d extreme value, then $\varepsilon^*_{ijt} = \varepsilon_{ij} - \varepsilon_{it}$ follows the logistic distribution

$$F(\varepsilon^*_{ijt}) = \frac{e^{\varepsilon^*_{ijt}}}{1 + e^{\varepsilon^*_{ijt}}} \tag{5.6}$$

We now can derive the logit choice probability by using the probability function in (5.4) and (5.6).

Nested logit models are an extension of the multinomial logit model which is designed to capture correlations among alternatives. It is based on the partitioning of the choice set $\mathscr{A}$ into disjoint subsets $\mathscr{A}_k$ (which are called nests).

$$\mathscr{A} = \bigcup_{k=1}^{n} \mathscr{A}_k, \quad \text{and } \mathscr{A}_k \cap \mathscr{A}_l = \emptyset, \quad \forall k \neq l. \tag{5.7}$$

A direct extension of the nested-logit models consists in partitioning some or all nests into sub-nests. Because of the complexity of these models, their structure is usually represented as a tree (see [11]). The number of potential correlation structures can be very large and no technique currently exist to identity the most appropriate one from the data.

Generalized extreme value (GEV) models are derived from the Generalized extreme value distribution. In probability theory and statistics, the generalized extreme value distribution is a family of continuous probability distributions developed within the extreme value theory to combine the Gumbel, Fréchet and Weibull families [15]. In a GEV model, the probability of an individual i choosing alternative $A_j \in \mathscr{A}(i)$ is given by

$$P_{ij} = \frac{e^{V_{ij}} \frac{\delta G}{\delta x_j}(e^{V_{i1}}, \dots, e^{V_{iA_n}})}{\mu G(e^{V_{i1}}, \dots, e^{V_{iA_n}})} \tag{5.8}$$

where $G : \mathbb{R}^n_+ \to \mathbb{R}$ with the following properties:

1. $G(\cdot)$ is differentiable.
2. $G(x) \geq 0 \quad \forall x \in \mathbb{R}^n_+$.
3. $G(\cdot)$ is homogeneous of degree $\mu > 0$, that is $G(\alpha x) = \alpha^\mu G(x) \quad \forall x \in \mathbb{R}^n_+$.
4. $\lim_{x_t \to \infty} G(x_1, \dots, x_t, \dots, x_n) = +\infty$ for all $l \in [1, n]$.
5. The k^{th} partial derivative with respect to k distinct x_j is non-negative if k is odd and is non-positive if k is even, that is $\forall j_1, \dots, j_k$ such that $1 \leq l \leq n \quad \forall l \in [1, k]$ and $j_l \neq j_m \quad l \neq m$ and $l, m \in [1, k]$, we have:

$$\frac{\delta^k G}{\delta x_{j1} \dots \delta x_{jk}}(x) = \begin{cases} \geq 0 & \text{k is odd} \\ \leq 0 & \text{if k is even.} \end{cases}$$

We note that Logit and Nested-logit are both special case of GEV models. We can obtain Logit model if $G(x) = \sum_{j=1}^n x_j^\mu$, and Nested-logit if $G(x) = \sum_{k=1}^n (\sum_{j \in \mathscr{A}_k} e^{\sigma_k x_i})^{\mu / \sigma_k}$.

Mixed-logit models have been known for many years but have only become fully applicable since the advent of simulation. Mixed-logit, presented by McFadden and Train [19], is a highly flexible model that can approximate any random utility model. It obviates the three limitations of standard logit by allowing for random taste variation, unrestricted substitution patterns, and correlation in unobserved factors over time. Mixed-logit models can be derived under a variety of different behavioural specifications, and each derivation provides a particular interpretation. The mixed-logit model is defined on the basis of the functional form for its choice probabilities. Any behavioural specification whose derived choice probabilities take this particular form is called a mixed-logit model. The first application of mixed-logit was apparently the demand for electricity-using goods ([13]).

In mixed-logit models we assume that each parameter vector $\beta(i), (i = 1, \dots, I)$ is a realization of a random vector β. Furthermore, β is itself derived from a random vector ω and a parameter vector θ, which we express as $\beta = \beta(\omega, \theta)$. ω typically specifies the random nature of the model and the vector parameters θ quantifies the population characteristic for the model. If we know the realization $\omega(i)$ for some individual i, we have $\beta(i) = \beta(\omega(i), \theta)$. The probabilities that individual i chooses alternative j would then be given by the standard logit formula

$$L_{ij}(\beta) = L_{ij}(\omega, \theta) = \frac{e^{V_{ij}(\beta(i), x_{ij})}}{\sum_{t=1}^{|\mathscr{A}(i)|} e^{V_{it}(\beta(i), x_{it})}}. \tag{5.9}$$

Because vector β is random, we need to compute the associated unconditional probability, which is obtained by integrating (5.9) over the random parameters ω:

$$P_{ij} = E_P[L_{ij}(\omega, \theta)] = \int L_{ij}(\omega, \theta) f(\omega) d\omega$$

where P is the probability measure associated to ω, $E_P[L_{ij}(\omega, \theta)]$ is the mathematical expectation of logit probability over the probability measure P and $f(\cdot)$ is the density function. The evaluation of P_{ij} requires the evaluation of one multidimensional integral per individual. The value is therefore replaced by some approximation, obtained by the Monte Carlo simulation and setting by sampling over (see Bastin et al. [2]), and given by:

$$P_{ij} \approx SP_{ij}^R(\theta) = \frac{1}{R} \sum_{r_i=1}^{R} L_{ij}(\omega_{r_i}, \theta) \tag{5.10}$$

where R is the number of random draws ω_{r_i}, taken from the distribution function of ω.

Estimating a mixed-logit model is numerically very expensive, even when Monte-Carlo approximation are used, the choice of an adequate optimization procedure is therefore crucial. In the next section, we will explore the problem of maximum likelihood and its application in mixed-logit models.

5.6 Mixed-logit model estimation

Having defined the form of the choice probabilities, we now face the problem of estimating the parameters vector β in the alternative utilities. This is usually done by the means of the maximum likelihood (ML) method. Assume that we have a sample of I

individuals from an homogeneous population. If this population is large, we can assume that the observations in the sample are independent. The likelihood function is then the product of individual choice probabilities:

$$L(\beta) = \prod_{i=1}^{I} P_{ij_i}(\beta)$$

If $\hat{\beta} = \arg\max L(\beta)$, then $\hat{\beta}$ corresponds to the parameters vector which has the greatest probability of having generated the observed sample. In practice, as we described it in the likelihood section, when I is large, evaluating of the likelihood function is numerically stable since $0 \leq P_{ij_i} \leq 1$, $(i = 1, \ldots, I)$, and, more importantly, the maximization of a product is often less stable than the maximization of a sum. To avoid these difficulties, it is preferable to consider the logarithm of the likelihood

$$LL(\beta) = \sum_{i=1}^{I} \ln P_{ij_i}(\beta)$$

In the context of mixed-logit model, the parameter vector β is itself derived from a random vector ω and a parameter vector θ, and the probability is estimated by integrating over the random parameters ω as in (5.10). From that, the vector of parameters θ is estimated by maximizing the log-likelihood function, i.e. by solving the problem below:

$$\max_{\theta} LL(\theta) = \max_{\theta} \sum_{i=1}^{I} \ln(P_{ij_i}(\theta))$$

where j_i is the alternative choice made by the individual i. We note that the normalization factor $\frac{1}{I}$ is often used for consistency with the stochastic programming literature (Shapiro [25]). As a result, the value of θ is estimated by solving the log-likelihood simulation problem:

$$\max_{\theta} SLL(\theta) = \max_{\theta} \frac{1}{I} \sum_{i=1}^{I} \ln(SP^{R}_{ij_i}(\theta)),$$

where $SP^R_{ij_i}(\theta)$ is an approximation of $P_{ij_i}(\theta)$, which can be obtained by the Monte Carlo simulation (see 5.10). We notice that this problem can be viewed as a special case of the stochastic programming problem, which we restate for clarity as:

$$\min_\theta g_R(\theta) = \min_\theta -SLL(\theta) = \min_\theta -\frac{1}{I}\sum_{i=1}^{I}\ln(SP^R_{ij_i}(\theta))$$

This is a non-linear optimization problem that must be solved by some iterative technique. The methods for solving this problem are presented in the above chapter. Recall that the log-likelihood function can be approximated by Monte-Carlo method, see Fabian and al. [3], or by RQMC methods (see for instance Munger and al. [21]). Consequently the log-likelihood simulation problem is very expensive to solve. The optimizations which are designed for this class of problems have to avoid repeatedly calculating the likelihood values. In the next chapter, we report and analysis the numerical results of the optimal algorithm which are introduced above for the real data sets of discrete choice model. We also point out the dominance of our new algorithms, compares to the classical algorithms, especially with the complex discrete choice model with panel data.

CHAPTER 6

NUMERICAL ASSESSMENT

Chapter 3 propose several new approaches to select under model switching the most reasonable sub-problem or search direction for determining the step at each iteration of line-search and trust-region methods. In the present chapter, we describe numerical experiments that have been conducted on real discrete choice data sets. We evaluate the performance of our new approaches using comparisons with results from classical methods for estimating the parameters that we have coded and executed on the same discrete choice data sets. Moreover, in the context of line-search, Chapter 4 proposes a new line-search algorithm, called adaptive line-search, as a new optimization algorithm for non-linear non-convex problems. We evaluate as well the performance of this new algorithm based on numerical results obtained from real discrete choice data sets.

6.1 Switching algorithms for mixed-logit models

Chapter 5 introduced the mixed-logit estimation problem which can be viewed as a special case of the stochastic programming problem. The mixed-logit estimation problem is restated here:

$$\min_{\theta} g_R(\theta) = \min_{\theta} -SLL(\theta) = \min_{\theta} -\frac{1}{I}\sum_{i=1}^{I} \ln(SP^R_{ij_i}(\theta))$$

where $SP^R_{ij_i}(\theta)$ is the approximation of the probability P_{ij_i} by Monte Carlo simulation

$$P_{ij_i}(\theta) \approx SP^R_{ij_i}(\theta) = \frac{1}{R}\sum_{t=1}^{R} L_{ij_i}(\omega_t, \theta)$$

in which $(\omega_1, \omega_2, ..., \omega_R)$ are R random draws taken from the distribution of the random parameter ω. The problem $\min_{\theta\in\mathbb{R}^N} g_R(\theta)$ is considered as a non-linear non-convex problem. Bastin and al. [2] proposed a new algorithm, called *Trust-region algorithm* *with dynamic accuracy*, which allows to adapt the number of draws such to reduce the computational time of likelihood functions. The tests described in this chapter keep the number of draws constant along the iterative process, but eventually we can modify our algorithms based on the idea in [2] for better performance.

Chapter 3 describes four techniques to select Hessian approximations: Predictive, Retrospective, Multi sub-problem and Multi sub-problem with BHHH. We have included these techniques individually into the general switching model for trust-region method (**Algorithm 3.1**) and into the general switching model for line-search method (**Algorithm 3.2**), each yielding a different optimization algorithm. In the context of the switching model for the trust-region method, we identify these algorithms as BTR-SW-PRED, BTR-SW-RETRO, BTR-SW-MULTI and BTR-SW-MULTI-BHHH respectively for trust-region using the Predictive approach (**Algorithm 3.3**), the Retrospective approach (**Algorithm 3.4**), the Multi sub-problem approach (**Algorithm 3.5**) and the Multi sub-problem with BHHH approach (**Algorithm 3.6**). We compare the performance of our algorithms with the basic trust-region method described in Algorithm 2.1 for different Hessian approximations. We have implemented BTR with respectively BHHH, BFGS and SR1 updates yielding algorithms BTR-BHHH, BTR-BFGS, BTR-SR1. We also compare with the combined approximation algorithm introduced by Bunch. The combined approximation algorithm adds a correction term to the BHHH update. Dependent on the correction term that is used, we obtain different algorithms. We have implemented the algorithms BTR-CB-BFGS, BTR-CB-SR1 which use the secant equation 2.23 to approximate the second term of the combined approximation while the algorithms BTR-CB2-BFGS, BTR-CB2-SR1 apply the secant condition in 2.25.

Similarly, in the context of the switching model for line-search algorithm, we identify our algorithms as LNS-SW-PRED and LNS-SW-RETRO respectively for line-search with the Predictive approach (**Algorithm 3.3**) and the Retrospective approach (**Algorithm 3.4**). We compare the performance of our algorithms with other line-search methods that we have implemented, LNS-BHHH, LNS-BFGS, LNS-CB-BFGS which apply respectively the basic line-search method with statistical, BFGS and combined approximation. The ALNS-BHHH and ALNS-BFGS refer to implementations of our adaptive line-search described in **Algorithm 4.1**, with BHHH and BFGS update, respectively. These algorithms are summarized in Table 6.I below.

Methods	Algorithms	Description
Trust-region	BTR-BHHH	Basic trust-region algorithm (BTR) with BHHH update
	BTR-BFGS	BTR with BFGS update
	BTR-SR1	BTR with SR1 update
	BTR-CB-BFGS	BTR with combined approximation and BFGS
	BTR-CB-SR1	BTR with combined approximation and SR1
	BTR-CB2-BFGS	BTR with combined approximation and BFGS
	BTR-CB2-SR1	BTR with combined approximation and SR1
	BTR-SW-PRED	BTR with predictive model (**Algorithm 3.3**)
	BTR-SW-RETRO	BTR with retrospective model(**Algorithm 3.4**)
	BTR-SW-MULTI	Multi sub-problems model (**Algorithm 3.5**)
	BTR-SW-MULTI-BHHH	Multi sub-problems with BHHH model (**Algorithm 3.6**)
Line-search	LNS-BHHH	Basic line-search algorithm (LNS) with BHHH update
	LNS-BFGS	LNS with BFGS update
	LNS-CB-BFGS	LNS with combined approximation
	LNS-SW-PRED	LNS with predictive model (**Algorithm 3.3**)
	LNS-SW-RETRO	LNS with retrospective model (**Algorithm 3.4**)
Adaptive line-search	ALNS-BHHH	Adaptive line-search with BHHH update (**Algorithm 4.1**)
	ANLS-BFGS	Adaptive line-search with BFGS update (**Algorithm 4.1**)

Table 6.I: List of algorithms

Chapter 2 also introduced some stopping conditions. Beside criteria for successful processes, algorithms also need some stopping conditions for when they cannot converge to an optimal solution. Table 6.II presents these other stopping tests which are used in our implementations.

Criteria	Stopping test	Description
$\nabla f_k \leq \varepsilon$	GRADIENT	Successful
$\bar{\nabla} f(x_k) \overset{\text{def}}{=} \max_c \left(\frac{\lvert[\nabla f(x_k)]_c\rvert, \max\{[x_k]_c, 1.0\}}{\max\{\lvert f(x_k)\rvert, 1.0\}} \right) \leq \varepsilon$	RELATIVE GRADIENT	Successful
$k \geq$ MAX-ITER	ITERATION	Fail
$0 < x_{k+1} - x_k \leq \varepsilon$	STEP	Fail
$\Delta_k \leq \varepsilon$	TRUST-REGION RADIUS	Fail
	LINE-SEARCH RADIUS	Fail

Table 6.II: Summary of stopping criteria

6.2 Discrete choice data sets

We use two real-life data sets which are complex enough to validate our algorithms. The two data sets are described in sections 6.2.1 and 6.2.2 below.

6.2.1 Cybercar model

This is a data set that has been collected in April 2008 at the Baltimore/Washington Intentional airport (BWI), it concerns the use of an automated vehicle technology called Cybercars (Cirilo and Xu [9]). The respondents were met in a waiting area of the airport and the responses were recorded during face-to-face interviews. The final sample contains information from 274 respondents. Both Revealed Preference data (RP) and Stated Preference (SP) information were collected. For SP, the experiment includes two parts: SP1 (SP game 1) is a between-mode experiment and SP2 (SP game 2) is a within mode experiment. SP1 is mainly about ground access mode choices, it includes the hypothetical cybercar service as well as three other existing modes: car, transit and taxi. SP2 proposes two different cybercar services over which the respondents are asked to express their preferences. In each model, the respondents were presented with 9 scenarios, where the attribute level of variations were based upon the respondents real trip to the airport as reported in the RP questionnaire. We therefore have a total 2466 observations. In our tests we use only SP2. Table 6.III below lists the variables that describe the service in this game.

A number of parametric models for the distributions will be estimated and compared. The retained model assumes that the waiting time distribution parameters are fixed cost individuals, that the cost is log-normal, and that the remaining service level variables are

Dropping area	Ternimal bulding, parking lot
Manoeuvring system	Full automated, human driver with ITS, human driver
Waiting time	5, 10, 15, 20 (in minutes)
Travel cost	70% of taxi, 85% of taxi, same as taxi
Track structure	Guideway, grade with rubber tire

Table 6.III: The variables and their admissible levels for SP2

normally distributed. The distribution of the components of β are given in Table 6.IV, where the first three components of β have constant value, then we have five components that have to be simulated. Note that $N(\mu,\sigma^2)$ and $\ln N(\mu,\sigma^2)$ refer to the normal and log-normal distributions, respectively, with parameter μ and σ. Therefore the vector of parameters θ (to be estimated) is $\theta = (\theta_1,\theta_2,\theta_3,\mu_1,\sigma_1,\mu_2,\sigma_2,\mu_3,\sigma_3,\mu_4,\sigma_4,\mu_5,\sigma_5)$.

Coordinate index	Distribution
1	constant
2	constant
3	constant
4	$\ln N(\mu_1,\sigma_1^2)$
5	$N(\mu_2,\sigma_2^2)$
6	$N(\mu_3,\sigma_3^2)$
7	$N(\mu_4,\sigma_4^2)$
8	$N(\mu_5,\sigma_5^2)$

Table 6.IV: Distribution of the components of β with the real data SP2

6.2.2 IRIS model

In order to evaluate the performance of our new algorithms on a large-scale model, where statistical approximation do not work well, we consider a data set that was collected in Autumn 2002 in Brussels (Belgium). One of the main objective of this survey was to test the propensity to switch from car to a more efficient Public Transport service, with better access, new high-speed lanes and improved comfort. This is part of a larger survey conceived for the estimation of a new regional transport model, called **IRIS**. Car users were asked to fill out a questionnaire under the direct assistance of interviewers. They were asked to consider three scenarios based on their current trip and then to express their choices. The survey contains seven variables, of which six present three levels of variations and one has two levels of variations. In order to reduce the size of the total data set, we adopted an orthogonal design. The levels of variations depend on the total car distance from the origin to the destination; we distinguish three classes: less than 10 km, between 10 and 30 km, greater than 30 km. In summary, the variables and the levels were chosen as follows (in brackets we indicate the value levels for classes of distance between 10 and 30 km and greater than 30 km)

- Car time: +5(10, 20) min, +10(20, 30) min, +20(30, 50) min compared to actual car time;
- Car cost: +0.06 €/km compared to actual cost.
- Toll: 1, 3, 7 €.
- Delayed Departure Time: -45 min, +30 min, +60 min on the actual departure time;
- PT time: -10(-20,-30) min, +5(0,-5), +15(+10,+10) min, compared to actual car time;
- PT Cost (ticket + parking/per month): 25(40, 50), 45(70, 85), 75(105, 130) €;
- Comfort: No seats available-very crowded, no seats available-not crowded, seats available.

For this model, only trips with work as final destination have been considered. After cleaning the data set, a total number of 2602 observations from 871 individuals were entered into the model. There were four choices of options available to the respondents: car, car with delayed departing time, car on a high occupancy vehicle (HOV) dedicated lane and public transport (PT). Each option was specified with a different utility for car drivers (CD) and for passengers (CP), giving a total of eight alternatives; in particular the High Occupancy Vehicle lane was toll free when at least two passengers shared the same car.

The model contained a total of 18 exogenous variables, of which four alternatives specific constants (car passenger with delayed departure time, car as driver on HOV lane, shared car on HOV and Public Transport). Seven levels of service variables (congested and free flow time, cost, HOV toll, origin destination distance, comfort on two levels of variations), three departure time variables, two variables representing socio-economic characteristics (being manager or self-employed) and the remaining describing trip characteristics (trip frequency per week, dummy for stopping to pick up/drop off children). Seven of the explanatory variables are randomly distributed, with two of them assumed to be normal or log-normal (congested and free flow time coefficients) and the remaining five assumed to be normal. If the congested and free flow time coefficients have normal distribution we have the **IRIS-NORMAL** model, if they have a log-normal distribution we have the **IRIS-LOGNORMAL** model. Thus the vector of parameter (to be estimated) contains 25 components. For more details, we refer the reader to Bastin et al. [3]. Table 6.V below summarizes the distribution of the variables.

Variable	Distribution
Car passenger (CP)	constant
HOV (HOV)	$N(\mu_1, \sigma_1^2)$
Shared car on HOV (HOVs)	$N(\mu_2, \sigma_2^2)$
Public transport (PT)	constant
Congested travel time (LN)	$N(\mu_3, \sigma_3^2)$ or $\ln N(\mu_3, \sigma_3^2)$
Free-flow travel time (LN)	$N(\mu_4, \sigma_4^2)$ or $\ln N(\mu_3, \sigma_3^2)$
Cost	constant
Toll (HOV)	constant
Dist. (CD,CP,CDs,CPs,HOV,HOVs)	$N(\mu_5, \sigma_5^2)$
Trip frequency-once a week (PT)	constant
Comfort no-seats (PT,PTs)	constant
Comfort no seats, crowded (PT,PTs)	constant
Earlier departure time (CP,CPs)	$N(\mu_6, \sigma_6^2)$
Later departure time (CP,CPs)	$N(\mu_7, \sigma_7^2)$
Much later departure time (CP,CPs)	constant
Self-employed (CD,HOV)	constant
Manager (HOV)	constant
Number of cars-3 per HHLD (CD)	constant

Table 6.V: Distribution of the components of β with the real data IRIS

6.3 Numerical assessment with AMLET

Numerical evaluation of the algorithms is based on the package AMLET, initially developed by Fabian Bastin [1]. AMLET stands for **Another Mixed Logit Estimation Tool**. As its name suggests, it is a software originally designed to estimate various kind of mixed-logit models, while offering various tools for simulation and optimization. AMLET is available in open source at the address *http://amlet.slashbin.net*, along with its companion libraries ORATIO and OPHELIA.

In order to limit as much as possible the timing differences between the three algorithms due to implementation, all algorithms were rewritten directly in the core of AMLET, taking into account the standard recommendations in the existing literature.

For BTR, we have followed the guidelines proposed by Conn et al. [10], while for the basic line-search, we have followed the suggestions given by Nocedal and Wright [23]. In particular, we have implemented More-Thuente line-search [20], which is currently considered as the best line-search technique. It worths observing at this point that the trust-region approach is simpler to implement efficiently than the line-search method.

6.4 Numerical experiments

We now describe the experiments that we have conducted on the data sets SP2 and IRIS. For the SP2 model, we estimate with 1021 random draws per individual, and evaluate the results over 10 independent simulations. The IRIS model is estimated with 2000 random draws per individual and also over 10 simulations. We chose the zero vector as the starting point while the threshold to stop the iterative processes is set at $\varepsilon = 5\times 10^{-5}$ for SP2 and $\varepsilon = 2.9873708\times 10^{-6}$ for IRIS.

We estimated both models (SP2 and IRIS) with our new algorithms and also with classical algorithms (basic trust-region and line-search using the classical Hessian approximation) in order to evaluate numerically our proposed algorithms. The number of iterations, the computational times (always reported in seconds) and the values at convergence are observed to evaluate and compare the performance of the algorithms. Note that computational times are provided for the 10 simulations [1].

6.4.1 Comparison between classical algorithms

In this section we evaluate the performance of some methods with which we compare our algorithms. We present numerical results of the classical trust-region algorithms BTR-BHHH, BTR-BFGS, BTR-SR1 and the line-search algorithms LNS-BHHH and LNS-BFGS. We also provide numerical results for the combined approximation algorithm. In this section we report computational times, we compare algorithms along this criterion. We do not report the number of iterations as for the trust region algorithms

1. Though those simulations are independent, in the figures, numerical results from a same algorithm are connected by an edge to help contrast the performances between different algorithms.

the number of iterations is proportional to the computational time. We do not report the number of iterations for line-search algorithms unless the number of iterations varies more than a certain rule among methods. In this section, we will report failures to converge that are detected by exceeding the number of iterations stopping criterion. Failing to converge in this case is observed indirectly from the computational times.

Figures 6.1 and 6.2 report computational times for the three trust-region algorithms on data sets SP2 and IRIS-NORMAL. For both models, the algorithm BTR-BHHH is slightly better than the others algorithms, especially for the IRIS-NORMAL model. For the 10 simulations on the SP2 model, BTR-BHHH needs 25.9(s) on average to converge while the average computational time is 50.6(s) for BTR-BFGS and 41(s) for BTR-SR1. BTR-BHHH compares even better when we observe the numerical results for the IRIS-NORMAL model. The average computational time for BTR-BHHH is 170.4(s), which is 15% of the average computational time of BTR-BFGS (1159s) and 10% of BTR-SR1 (1754.8s). These results show that statistical approximation compared much better than other classical approximations which explains why in many cases BHHH is the favourite approach for MLE.

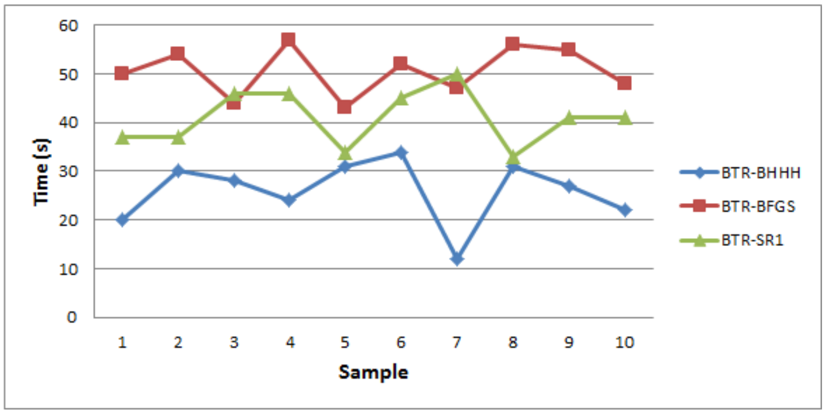

Figure 6.1: Optimization time of basic trust-region algorithms [SP2]

With SP2 and IRIS-NORMAL, the BHHH update is the best choice to estimate parameters. However, for the more complex model IRIS-LOGNORMAL, the advantage of the BHHH approach over the other methods disappears as the average optimization time of the BHHH method is significantly larger than for the other methods, in some runs,

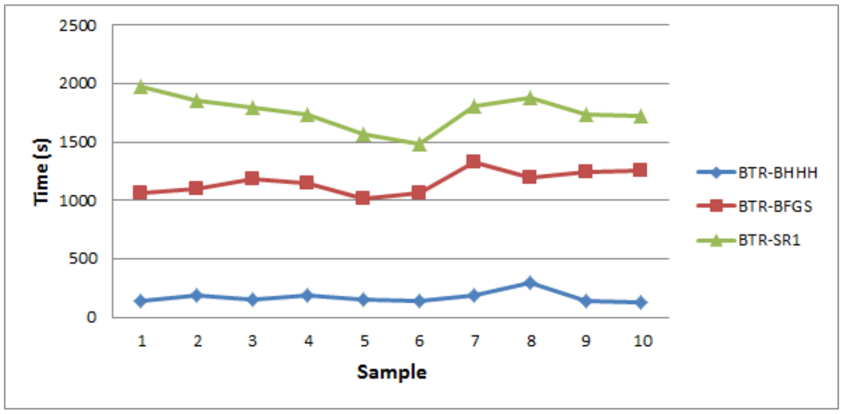

Figure 6.2: Optimization time of basic trust-region algorithms [IRIS NORMAL]

BHHH does not even converge. The computational times for the IRIS-LOGNORMAL data set are given in Figure 6.3. This figure shows that optimization time depends on the sample for BHHH, while it is more stable for BFGS and SR1. On the 10 simulations, eight of them (80%) have been especially time consuming (greater than 2500 seconds), while only two (20%) have converged very rapidly. We also note that all eight runs with optimization time over than 2500 seconds are failure runs, the iterative process has been stopped because the number of iterations exceeded the maximum number of iterations allowed before convergence. This happens typically because information identity does not hold, so that the BHHH approximation is poor close to the solution of maximum likelihood. Recall the information identity property assume a correctly specified model, but unfortunately, perfect information remains elusive. Moreover, observing the numerical results for the SR1 approach, we realize the advantage of the BFGS update over the SR1. With the SP2 model, the SR1 approach is faster than BFGS, but for the two complex models from the IRIS data set, BFGS is slightly better.

The numerical results for the combined approximation method are presented in Figures 6.4, 6.5 and 6.6. For these algorithms, there are no failures. Further, though this is not shown in the figures, we report that in our tests they have all converged to the same solution. In 6.4, 6.5 and 6.6, algorithm BTR-CB-BFGS appears slightly better than algorithm BTR-CB2-BFGS. For the IRIS-NORMAL data set, algorithm BTR-CB-BFGS is comparable to algorithm BTR-CB2-BFGS, but with the SP2 and the IRIS-

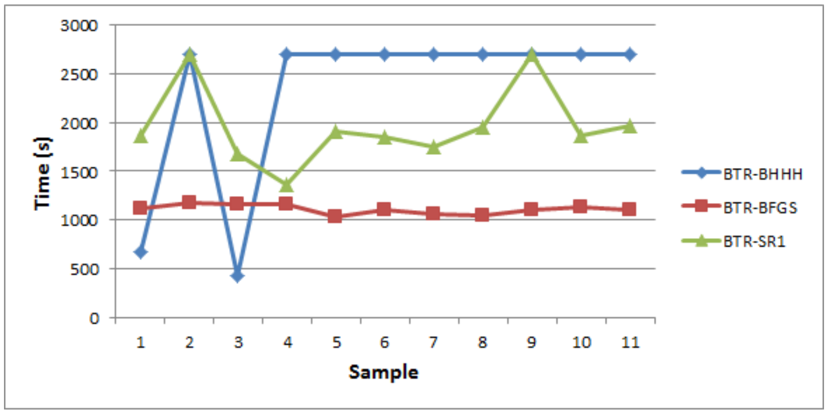

Figure 6.3: Optimization time of basic trust-region algorithms [IRIS LOG-NORMAL]

LOGNORMAL data sets, computational times indicate a clear dominance of BTR-CB-BFGS over the BTR-CB2-BFGS. According to these results, from now on, we only consider the combined approximation approach based on the secant equation 2.23 to analyze and compare other approaches.

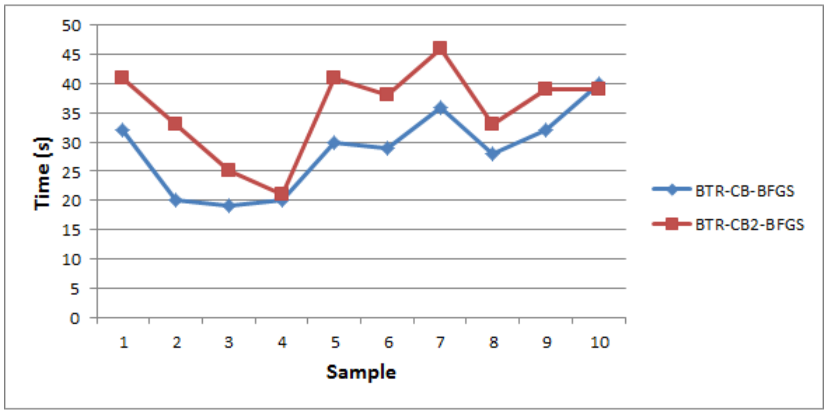

Figure 6.4: Optimization time of combined approaches [SP2]

Now we compare the two methods to approximate the second term of combined approximation: BFGS and SR1. While comparing these two methods, BTR-CB-BFGS and BTR-CB-SR1, we also add the results of other classical algorithms. These comparisons appear in Figures 6.7, 6.8 and 6.9.

Figure 6.7 and Figure 6.8 show that combined approximation performs much better than secant approximation, in particular with the IRIS-NORMAL model, but is

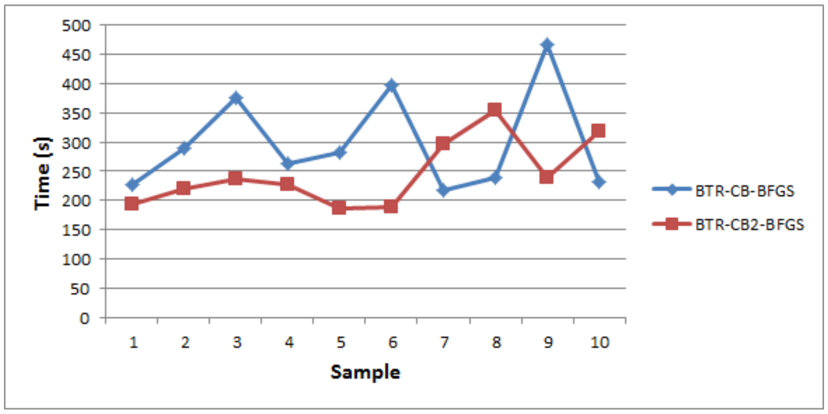

Figure 6.5: Optimization time of combined approaches [IRIS-NORMAL]

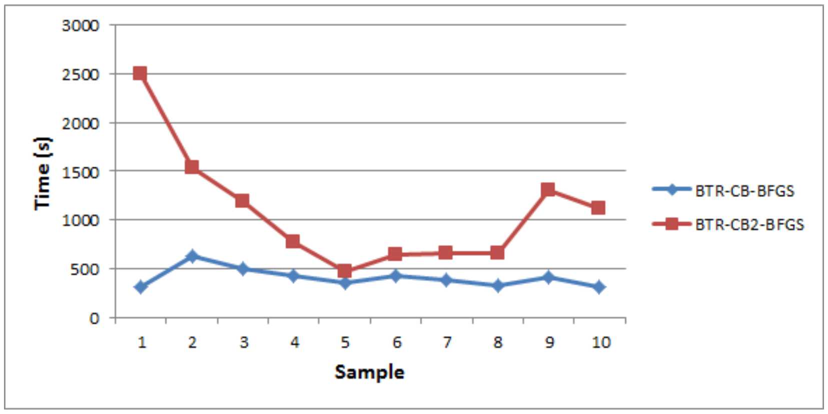

Figure 6.6: Optimization time of combined approaches [IRIS-LOGNORMAL]

still slower than the BHHH approximation. With SP2, the average computational time of BTR-BHHH is 25.9s, which is faster than BTR-CB-BFGS (28.6s) and BTR-CB-SR1(33.6s). For the IRIS-NORMAL model, the average optimization time of BTR-CB-BFGS is 300s, greater than BTR-BHHH (170s), while BTR-CB-SR1 is very slow (1295s). When estimating the IRIS-LOGNORMAL model, BHHH update has difficulties to reach convergence and is very slow as well. Figure 6.9 shows that BTR-CB-BFGS is still very effective, better than BTR-CB-SR1 and much better than the BHHH approximation. In summary, the results show that, in the context of the trust-region method, the combined approximation with BFGS can be a good replacement of the statistical approximation because it is more stable than the BHHH update and its optimization time is smaller than the secant approximation and the combined approximation with SR1.

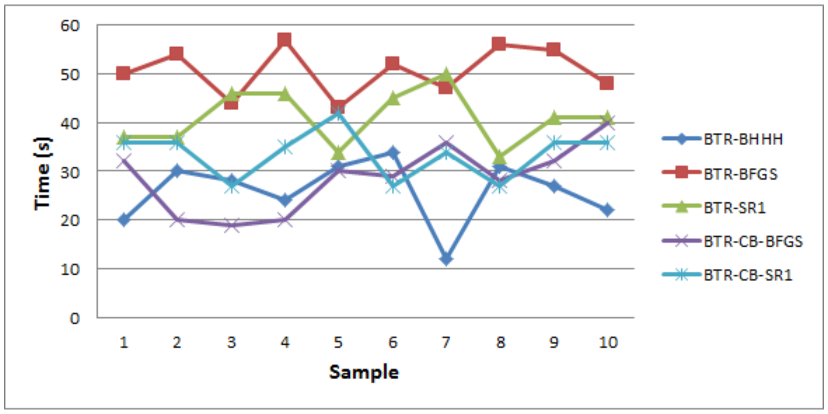

Figure 6.7: Optimization time of basic trust-region algorithms [SP2]

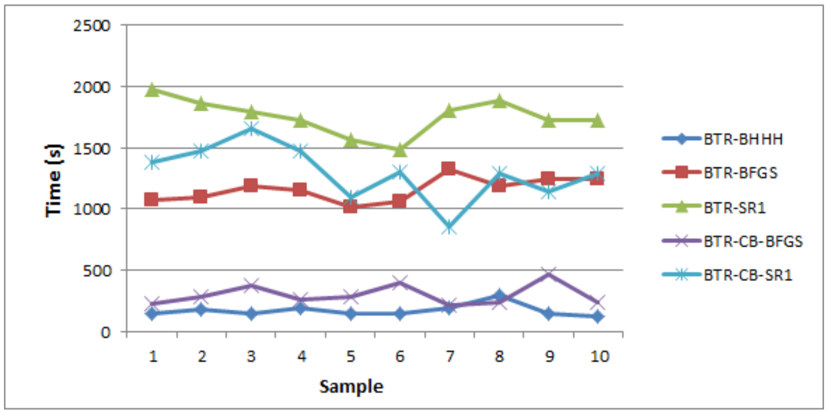

Figure 6.8: Optimization time of basic trust-region algorithms [IRIS-NORMAL]

The details of the average optimization times for trust-region algorithms based on the statistical, secant and combined approximation are summarized in Table 6.VI. The red numbers indicate that the corresponding algorithm has some failure runs, the small numbers on the top indicate the number of failure runs (i.e the number $2110s^8$ in the first line indicates that with IRIS-LOGNORMAL data, the algorithm BTR-BHHH has 8 failures over 10 simulations).

In the context of line-search, we note that the Hessian approximations have to be positive definite. We therefore consider three basic algorithms LNS-BHHH, LNS-BFGS and LNS-CB-BFGS with BHHH, BFGS and combined approximation, respectively. The numerical results can be found in the Figures 6.10 and 6.11 corresponding respectively to the IRIS-NORMAL and IRIS-LOGNORMAL models. The results are similar to the

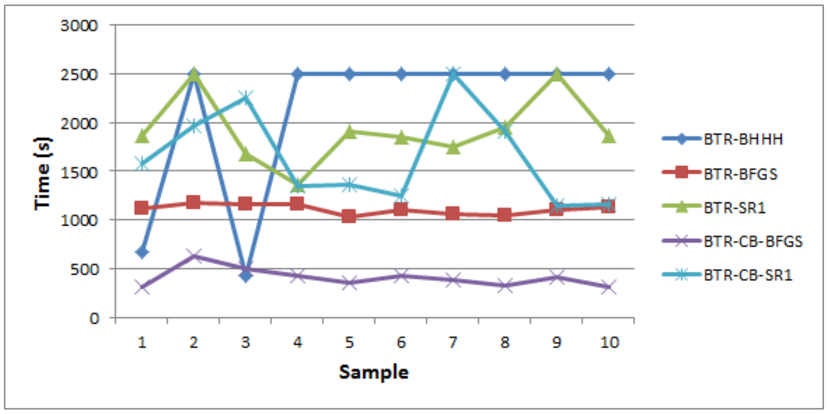

Figure 6.9: Optimization time of basic trust-region algorithms [IRIS-LOGNORMAL]

Algorithms	SP2	IRIS-NORMAL	IRIS-LOGNORMAL
BTR-BHHH	26s	170s	$2110s^8$
BTR-BFGS	51s	1159s	1110s
BTR-SR1	41s	1755s	$1925s^2$
BTR-CB-SR1	37s	1295s	$1083s^1$
BTR-CB2-BFGS	36s	246s	$1648s^2$
BTR-CB-BFGS	29s	299s	405s

Table 6.VI: Optimization time of basic trust-region algorithms

trust-region algorithms, algorithms LNS-BHHH and LNS-CB-BFGS are competitive on data sets SP2 (see Table 6.VII) and IRIS-NORMAL, but algorithm LNS-CB-BFGS is clearly faster and more stable than LNS-BHHH for the more complex data set IRIS-LOGNORMAL. The line-search algorithm with BFGS update is always the slowest algorithm. The average computational time of these algorithms is summarized in the Table 6.VII.

Algorithms	SP2	IRIS-NORMAL	IRIS-LOGNORMAL
LNS-BHHH	27s	143s	633s
LNS-BFGS	30s	795s	1602s
LNS-CB-BFGS	34s	240.4s	370s

Table 6.VII: Optimization time of basic line-search algorithms

We note for the IRIS-LOGNORMAL model, BHHH approximation under line-search

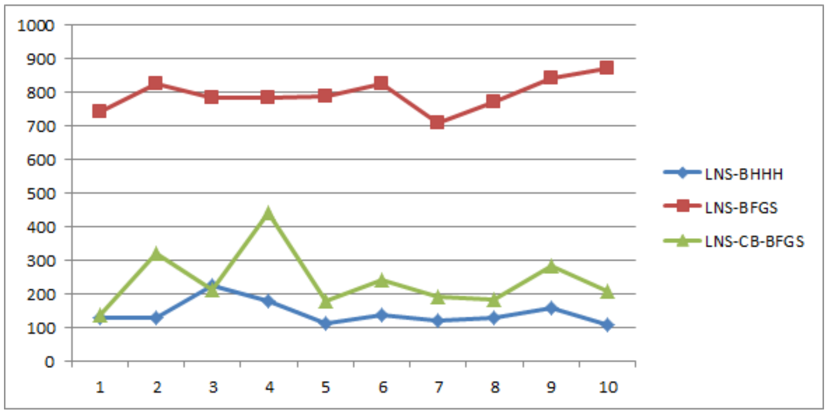

Figure 6.10: Optimization time of basic line-search algorithms [IRIS-NORMAL]

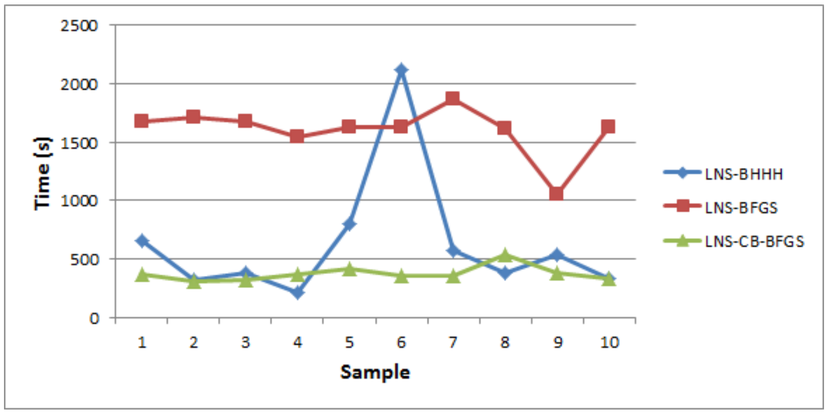

Figure 6.11: Optimization time of basic line-search algorithms [IRIS-LOGNORMAL]

has no failures while for the trust-region method the algorithm has 80% failure runs. The advantages and disadvantages of both line-search and trust-region methods will be discussed in more details in the final section of this chapter. We also notice that our results exhibit a well-known behaviour of the BHHH method. Under the conditions of information identity, the BHHH approach is very fast compared to the secant approximations. With more complex models, the advantage of the BHHH method disappears: the algorithm becomes unstable and very slow, especially close to the solution, as the BHHH update may not converge to the true Hessian. However, even when it is no longer effective, it provides as significant speed-up when used in combination with a secant method, for instance the BFGS technique.

6.4.2 Model switching algorithms with the trust-region method

We have proposed four algorithms for the trust-region method: BTR-SW-RETRO, BTR-SW-PRED, BTR-SW-MULTI and BTR-SW-MULTI-BHHH. Each of these algorithms requires a set of Hessian approximations or a set of methods to approximate the Hessian matrix at each iteration. The numerical results and the analysis in the previous section show that BHHH and combined approximation are better than the secant approximations. Therefore we use BHHH and combined approximation as the set of Hessian approximations for our model switching algorithms. We chose the BFGS update to estimate the second term of combined approximation. Numerical results are reported for the IRIS-NORMAL and IRIS-LOGNORMAL models. For SP2, the computational time for these algorithms is very similar. Consequently we have omitted the graph of the optimization times for SP2, but average computational times are given in Table 6.VIII.

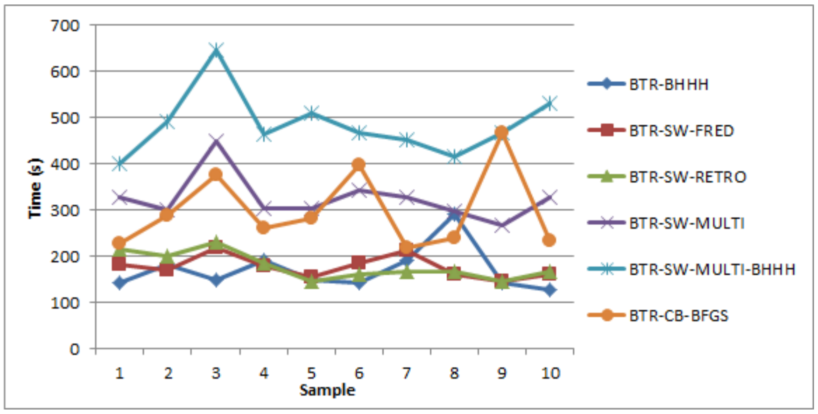

Figure 6.12: Optimization time of trust-region switching algorithms [IRIS-NORMAL]

Figures 6.12 and 6.13 report the optimizations times of our four algorithms for the trust-region method which are compared with the classical algorithms BTR-BHHH and BTR-CB-BFGS for the IRIS-NORMAL model and the IRIS-LOGNORMAL model (based on our previous numerical results, the trust-region algorithms with the classical Hessian approximation BTR-BFGS and BTR-SR1 perform poorly, so we do not consider these last two algorithms). For the IRIS-NORMAL model, our two algorithms, BTR-SW-PRED and BTR-SW-RETRO, give optimization times similar to BTR-BHHH (average time $\approx 175s$), they are also the three fastest algorithms. They are slightly bet-

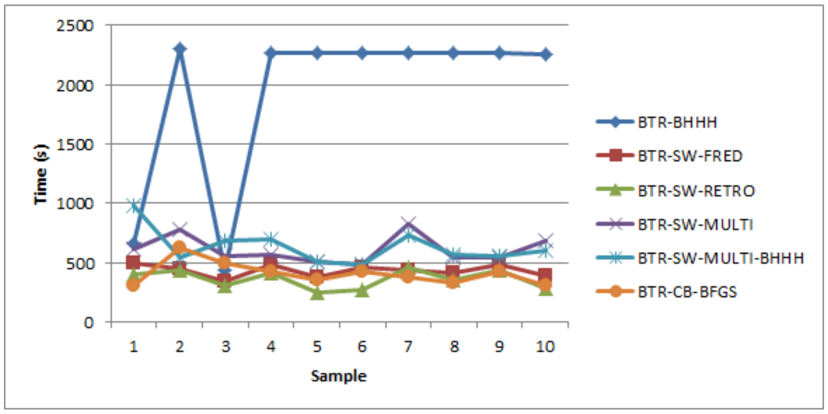

Figure 6.13: Optimization time of trust-region switching algorithms [IRIS-NORMALLOG]

ter than BTR-CB-BFGS (299s). For the IRIS-LOGNORMAL model, BTR-CB-BFGS, BTR-SW-PRED, BTR-SW-RETRO are the fastest algorithms. Algorithm BTR-BHHH is the worst with only 20% successful runs. The optimization times of the three fastest algorithms is: BTR-SW-RETRO (359s); BTR-CB-BFGS (405s); BTR-SW-PRED (430s). The average optimization time is summarized in Table 6.VIII.

Algorithms	SP2	IRIS-NORMAL	IRIS-LOGNORMAL
BTR-BHHH	26s	170s	2110s[8]
BTR-CB-BFGS	29s	299s	405s
BTR-SW-RETRO	25s	176s	359s
BTR-SW-FRED	17s	177s	430s
BTR-SW-MULTI	51s	324s	609s
BTR-SW-MULTI-BHHH	32s	485s	631s

Table 6.VIII: Optimization time of trust-region switching algorithms

Figures 6.12 and 6.13 show that the algorithms BTR-SW-MULTI and BTR-SW-MULTI-BHHH are the two slowest algorithms. However, comparisons along computational times do not portrait entirely the performance behaviour of the multi sub-problem approaches. These methods solve many sub-problems at each iteration. The computational time of each iteration is much larger than for trust-region algorithms or our algorithms with the predictive and retrospective approaches. Figures 6.14 and 6.15 report observations on the number of iterations with the IRIS-NORMAL and IRIS-

LOGNORMAL models respectively to converge to an optimal solution. In terms of the number of iterations over 10 simulations, BTR-SW-MULTI on the IRIS-NORMAL model is competitive, while on the complex IRIS-LOGNORMAL model it requires a significantly smaller number of iterations compared to other algorithms. Therefore, considering the number of iterations, we find that multi sub-problem approaches can find optimal solutions in a smaller number of iterations, i.e. the rate of convergence is higher. The average number of iterations is given in Table 6.IX.

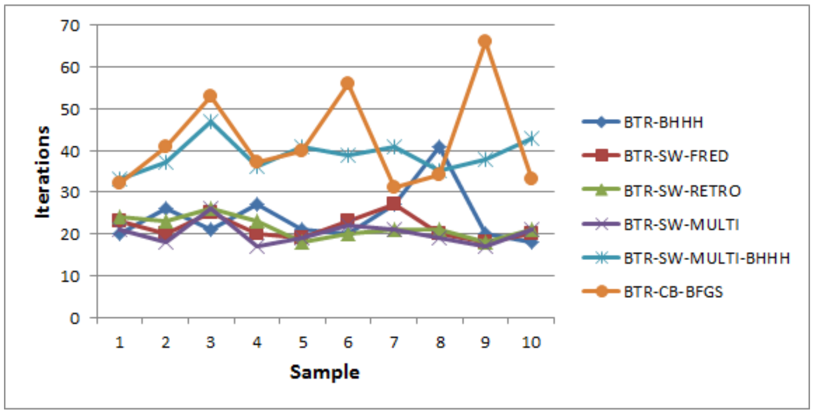

Figure 6.14: Number of iterations of trust-region switching algorithms [IRIS-NORMAL]

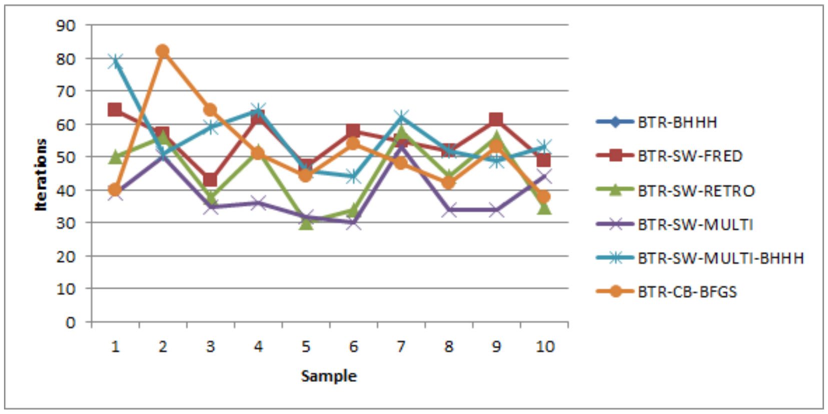

Figure 6.15: Number of iterations of trust-region switching algorithms [IRIS-LOGNORMAL]

In all previous experiments, we chose the standard starting point $x_0 = 0$ as the initial point of the iterative process. To evaluate the performance of the algorithms in difficult

Algorithms	IRIS-NORMAL	IRIS-LOGNORMAL
BTR-SW-MULTI	20 iters	39 iters
BTR-SW-MULTI-BHHH	39 iters	56 iters
BTR-SW-RETRO	22 iters	45 iters
BTR-SW-PRED	22 iters	55 iters
BTR-CB-BFGS	42 iters	52 iters

Table 6.IX: Number of iterations of trust-region switching algorithms

case, we chose a special starting point for IRIS-LOGNORMAL data set, the most complex model, which is very far from the optimal solution. Table 6.X reports the numerical results of the algorithms with the IRIS-LOGNORMAL model when the initial parameter is unnatural θ_0 = (20.0, -25.0, -20.0, 13.0, 21.0, 30.0, -14.0,-21.0, -13.0, -1.0, 31.0, -8.0, -22.0, 0.0, 4.0, -32.0, 11.0, -11.0, 32.0, -1.5, 12.0,15.2,-11.5, -0.6, 32.7). We note that the optimal parameter of this model is $\hat{\theta} \approx$ (-1.1,-5.5, 4.9, -7.3, 6.5, -0.64, -2.8, 1.0, -2.97, -1.10, 0.27, -0.52, 0.216, 0.24, 3.21, -1.14, -1.84, -3.28, -2.83, -2.45, 2.51, -2.71, 1.86, 1.37, 1.91), where the optimal log-likelihood value is ≈ -3.15, much bigger than the initial log-likelihood value (≈ -275.28). The results in Table 6.X suggest that the multi sub-problem with BHHH approach is especially efficient when the starting point is far from the optimal solution.

Algorithms	Successful	Fail	average computational time (of successful runs)
BTR-SW-MULTI-BHHH	60%	40%	1170s
BTR-SW-MULTI	50%	50%	1340s
BTR-SW-RETRO	50%	50%	1186s
BTR-SW-PRED	30%	70%	1176s
BTR-CB-BFGS	30%	70%	802s
BTR-BHHH	10%	90%	1040s
BTR-BFGS	0%	100%	*

Table 6.X: Rate of successful runs for difficult case [IRIS-LOGNORMAL]

The objective function in mixed-logit models is often very complex. The ability to converge can be reduced if the choice problem is complex and the starting point of the iterative process is difficult to chose. Consequently the optimization procedure needs to be selected carefully. Our results show that algorithms like BTR-SW-MULTI and BTR-SW-MULTI could decrease the number of iterations and increase the rate of successful runs. Second, the speed of convergence is important when we estimate complex models. The above results show that the BHHH approach can be very fast with some model, but it can be very slow with some others. Furthermore, with IRIS-LOGNORMAL model, the ratio of successful runs of BTR-BHHH is very low. The above results show that the speed and the ability to converge can be dramatically improved using the model switching approach, in particular using the retrospective model. The retrospective algorithm allows us to speed up the iterative process, and often to successfully terminate earlier.

6.4.3 Model switching algorithms with the line-search method

To test the performance of model switching under line-search, we have implemented two algorithms, LNS-SW-RETRO and LNS-SW-PRED, which are compared with three classical algorithms: LNS-BHHH, LNS-BFGS and LNS-CB-BFGS. For LNS-SW-PRED and LNS-SW-RETRO, we use the BHHH and the combined approximation (with the BFGS method to update the second term) to obtain the set of Hessian approximations at each iteration. Numerical results are derived from the three data sets: SP2, IRIS-NORMAL and IRIS-LOGNORMAL. However, as SP2 is pretty simple, the computational time of all algorithms is quite similar. We do not show the graph of computational times for the SP2, we only report the average computational times in tables.

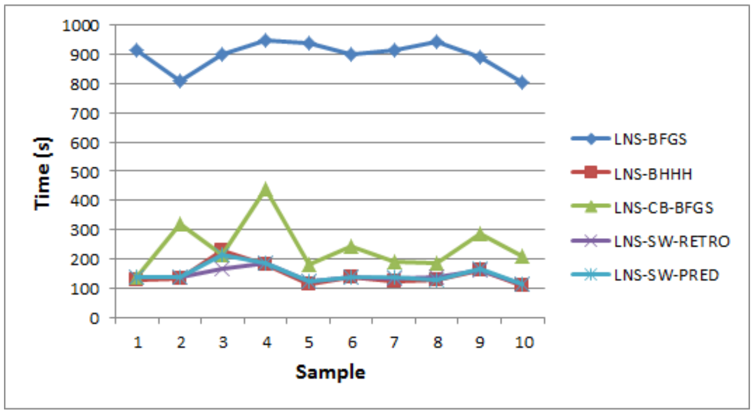

Figure 6.16: Optimization time of line-search switching algorithms [IRIS-NORMAL]

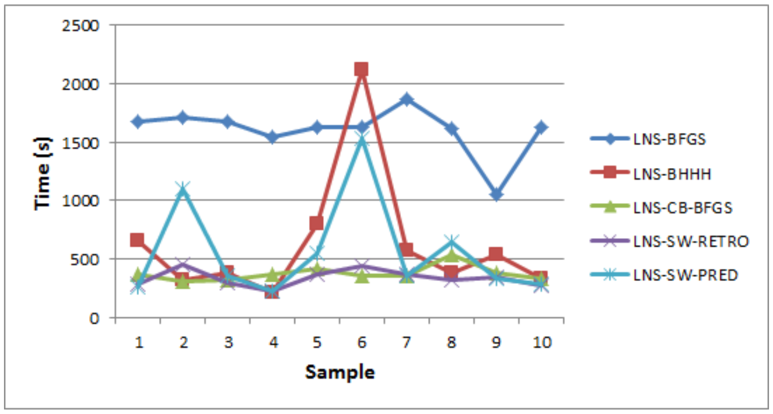

Figure 6.17: Optimization time of line-search switching algorithms [IRIS-LOGNORMAL]

Figure 6.16 show the results for the IRIS-NORMAL model. The figure shows that convergence for the basic linear-search with BFGS is much slower than for the other algorithms. The computational times of the basic line-search algorithm with BHHH and the two switching algorithms LNS-SW-RETRO and LNS-SW-PRED are quite similar and they are slightly better than the line-search algorithm with combined approximation LNS-CB-BFGS. The results for the IRIS-LOGNORMAL model are given in Figure 6.17. A close observation reveals that LNS-SW-RETRO and LNS-CB-BFGS have the best computational times, while LNS-BHHH and LNS-SW-PRED are slightly slower and they are unstable. The classical line-search with BFGS update is still the slowest algorithm. The average computational times are given in Table 6.XI.

Algorithms	SP2	IRIS-NORMAL	IRIS-LOGNORMAL
LNS-BHHH	27s	143s	633s
LNS-BFGS	30s	896s	1602s
LNS-CB-BFGS	34s	240s	371s
LNS-SW-FRED	32s	147s	563s
LNS-SW-RETRO	28s	142s	336s

Table 6.XI: Optimization time of line-search switching algorithms

The obtained results show that the behaviour of the classical line-search and switching line-search algorithms are quite similar to the behaviour of the trust-region method (refer to Figure 6.12 on page 67 and Figure 6.13 on page 68). More specifically, with both line-search and trust-region method, the BFGS approximation is the slowest approach with computational times that are always greater four or five times that of the other approaches (BHHH, combined and switching algorithms). The combined approximation is one of the three fastest algorithms with the IRIS-NORMAL data set, but still it is significantly slower than the switching algorithms. Moreover, the retrospective algorithm based on BHHH and combined approximation delivers significant speed-up compared to the standard techniques (and it is faster than the predictive algorithm). This suggests that the switching strategy is especially efficient when the data becomes more complex. These results also show that the retrospective approach is slightly better than the predictive one, in both line-search and trust-region approaches.

6.4.4 Comparing our adaptive line-search algorithm

We compare numerically our adaptive line-search algorithm with the basic line-search algorithm. The difference between our algorithm and the basic algorithm is the behaviour of the search direction along the iterative process. While the basic line-search algorithm set $p_k = -H_k^{-1}\nabla f_k$, the adaptive line-search adapts the length of search direction using parameter Δ_k, the value of this parameter can increase or decrease depending on the step size at each iteration. We have tested our adaptive line-search on the two most complex models: IRIS-NORMAL and IRIS-LOGNORMAL. With the SP2 model, the difference between the computational time of the algorithms over the simulations is not clear, so we just report the average computational times in Table 6.XI. We have two implementations of this adaptive line-search, one with the BHHH update (ALNS-BHHH) and one with the BFGS update (ALNS-BFGS). The numerical experiments show the efficiency of these implementations.

Figure 6.18 and 6.19 report the computational times of our two adaptive line-search implementations together with two basic line-search methods one with the BHHH update and the other with the BFGS update. Numerical results for IRIS-NORMAL data set in Figure 6.18 show that the adaptive line-search ALNS-BHHH performs better than ALNS-BFGS, algorithm ALNS-BFGS performs better than LNS-BFGS while ALNS-

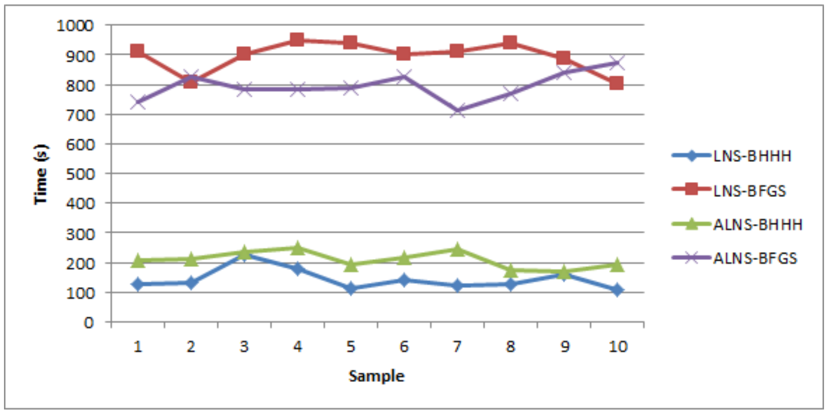

Figure 6.18: Optimization time of classical and adaptive line-search [IRIS-NORMAL]

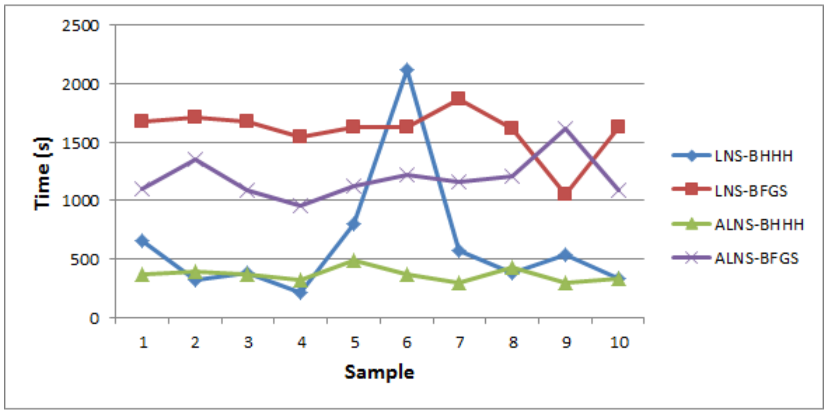

Figure 6.19: Optimization time of classical and adaptive line-search [IRIS-LOGNORMAL]

BHHH is slightly slower than LNS-BHHH. Numerical results for IRIS-LOGNORMAL data set in Figure 6.19 show a clear dominance of the adaptive line-search algorithms over the classical algorithms. For more details, the average optimization times can be found in Table 6.XII.

We note that in the above tests there were no failures. All algorithms almost converge to the optimal solution. The results change clearly when we observe the experiment for difficult cases (using the starting point far from the optimal solution defined in Section 6.4.2). Tables 6.XIII and 6.XIV show a clear dominance of the adaptive line-search over the basic line-search while this algorithm can converge very fast even when the initial point of the iterative algorithm is very far from the optimal solution and the log-

Algorithms	IRIS-NORMAL	IRIS-LOGNORMAL
LNS-BHHH	143s	633s
LNS-BFGS	896s	1602s
ALNS-BHHH	209s	363s
ALNS-BFGS	795s	1192s

Table 6.XII: Optimization time of classical and adaptive line-search algorithms

Algorithms	Simulation index	Computational time (s)	Log-likelihood value	Description
ALNS-BHHH	1	509	-3.14264	SUCCESS
	2	470	-3.14504	SUCCESS
	3	404	-3.14769	SUCCESS
	4	484	-3.14403	SUCCESS
	5	422	-3.13882	SUCCESS
	6	328	-3.14993	SUCCESS
	7	415	-3.14513	SUCCESS
	8	439	-3.15032	SUCCESS
	9	528	-3.14721	SUCCESS
	10	464	-3.14413	SUCCESS
LNS-BHHH	1	16	-233.228	FAILURE
	2	15	-232.84	FAILURE
	3	16	-234.79	FAILURE
	4	14	-232.069	FAILURE
	5	15	-234.404	FAILURE
	6	15	-235.206	FAILURE
	7	29	-228.959	FAILURE
	8	14	-231.673	FAILURE
	9	34	-232.479	FAILURE
	10	22	-233.242	FAILURE

Table 6.XIII: Numerical results of classical and adaptive line-search for difficult case [IRIS-NORMAL]

likelihood is much bigger than the optimal log-likelihood. With IRIS-NORMAL model, the adaptive algorithm has 100% successful runs with average computational time approximately 446.3 seconds, but the classical line-search algorithm has 10 failure over 10 simulations (LNS-BHHH fails on the too small STEP stopping criterion, i.e. it converges to the wrong solution). With IRIS-LOGNORMAL model, the adaptive algorithm is also very effective with 100% successful runs and takes on average 657.1 seconds to reach convergence, compare to 70% failure runs for the basic line-search.

Algorithms	Simulation index	Computational time (s)	Log-likelihood value	Description
ALNS-BHHH	1	508	-3.15772	SUCCESS
	2	1154	-3.16018	SUCCESS
	3	419	-3.15954	SUCCESS
	4	866	-3.15826	SUCCESS
	5	686	-3.15918	SUCCESS
	6	577	-3.16193	SUCCESS
	7	434	-3.15934	SUCCESS
	8	517	-3.15527	SUCCESS
	9	668	-3.15809	SUCCESS
	10	742	-3.15615	SUCCESS
LNS-BHHH	1	1219	-4.37112	FAILURE
	2	60	-9.04975	FAILURE
	3	402	-3.15947	SUCCESS
	4	754	-3.15826	SUCCESS
	5	68	-6.05811	FAILURE
	6	340	-5.34041	FAILURE
	7	792	-3.15768	SUCCESS
	8	7320	-11.767	FAILURE
	9	1666	-4.68267	FAILURE
	10	68	-6.7226	FAILURE

Table 6.XIV: Numerical results of classical and adaptive line-search for difficult case [IRIS-LOGNORMAL]

6.5 Summary and conclusion

Table 6.XV summarizes the numerical results that we have obtained on discrete choice data sets using our implementations of classical trust-region and line-search methods. We see that the algorithms based on the BHHH update often converge more rapidly to the optimal solution. However, with the complex model, the BHHH update may not converge to the Hessian of the objective, leading to poor performances close to the solution. Our numerical results show that the corrections to the BHHH approximation, called combined approximation, yield very effective and stable methods, even when the BHHH is no longer effective, when the BFGS update starts to be used, relying on the secant condition described e.g. in Dennis and Schnabel [12]. Within the classical approaches, our numerical results show that the combined approximation can be a good alternative of BHHH to solve MLE problems.

Table 6.XVI summarizes the numerical results of our algorithms for model switching, under both trust-region and line-search methods. Table 6.XVI also includes results from the two best classical algorithms to contrast with the performance of our model

Algorithms	SP2	IRIS-NORMAL	IRIS-LOGNORMAL
BTR-BHHH	26s	170s	*failure*
LNS-BHHH	27s	143s	633s
BTR-BFGS	51s	1159s	1110s
LNS-BFGS	30s	795s	1602s
BTR-CB-BFGS	29s	299s	405s
LNS-CB-BFGS	34s	240.4s	370s

Table 6.XV: Optimization time of basic algorithms

switching algorithms. We see in this table that the selection of the Hessian approximation based on the retrospective approach allows to greatly speed up the iterative process. In all cases, when the BHHH update in the fastest approach, the retrospective algorithm is always one of the fastest algorithms. The dominance of the retrospective approach is clearer when we observe the numerical results for the complex model (IRIS-LOGNORMAL). Results in Table 6.XVI also allow us to conclude that the retrospective approach is more appropriate in the context of model switching than the predictive approach, at least in terms of speed of convergence.

Algorithms	SP2	IRIS-NORMAL	IRIS-LOGNORMAL
BTR-BHHH	26s	170s	*failure*
LNS-BHHH	27s	143s	633s
BTR-BFGS	51s	1159s	1110s
LNS-BFGS	30s	896s	1602s
BTR-CB-BFGS	29s	299s	405s
LNS-CB-BFGS	34s	240s	371s
BTR-SW-RETRO	25s	176s	359s
LNS-SW-RETRO	28s	142s	336s
BTR-SW-FRED	17s	177s	430s
LNS-SW-FRED	32s	147s	563s
BTR-SW-MULTI	51s	324s	609s
BTR-SW-MULTI-BHHH	32s	485s	631s

Table 6.XVI: Optimization time of trust-region switching algorithms

The model switching algorithms with multi sub-problems for the trust-region are not the fastest, but as shown in Table 6.IX, they have a better rate of convergence, at least for BTR-SW-MULTI. The results shows that the multi sub-problem algorithm always requires the least number of iterations to converge to the optimal solution. As shown in Table 6.X, the advantage both multi sub-problem algorithms increases when we set the initial point very far from the optimal solution. In this case, the multi sub-problem and multi sub-problem with BHHH algorithms are the two most effective algorithms, with the highest rate of successful runs.

Our thesis also propose a new adaptive line-search algorithm which is a new optimization algorithm to solve MLE problems but also the class of non-linear, non-convex problems. Tables 6.XIII and 6.XIV show that our new line-search algorithm based on an adaptive approach is significantly better, especially with the more complex data sets and difficult cases (difficult initial solutions). However, given our narrow experimental tested, we cannot conclude that the adaptive line-search is better than the basic line-search or can be a good replacement for the classical algorithms, but it can certainly be an interesting approach to investigate.

CHAPTER 7

CONCLUSIONS AND FURTHER RESEARCH PERSPECTIVES

In this thesis, we have reviewed and proposed new algorithms aimed to maximize likelihood functions, assuming no constraints on the parameters. In this setting, we face unconstrained, non-linear and often non-convex mathematical programming problems. Our algorithms help improve two well-know approaches: line-search method and trust-region method. We have revisited some optimization approaches for maximum likelihood estimation, particularly the design of methods to approximate the Hessian matrix. We have exploited the idea of combining available Hessian approximation techniques in order to obtain better step at each iteration, an idea which we have framed into a general model switching method. We have designed the next four algorithms which apply model switching.

Predictive algorithm

The predictive algorithm proposes a method to predict the next Hessian approximation at the end of an iteration without any new calculation of the objective function or its derivatives. This is achieved by minimizing the approximation error or maximizing the trust-region ratio.

Retrospective algorithm

In the predictive algorithm, the next Hessian approximation is predicted at the end of a current iteration using the information available in that iteration. The retrospective algorithm is a more natural approach as it selects the Hessian approximation at the beginning of current iteration. To avoid computing more than one objective function at each iteration, the retrospective algorithm uses the computed objective function from the previous iteration. The relationship between the basic trust-region algorithm and the predictive algorithm, between the retrospective trust-region algorithm and the retrospective algorithm under model switching can be realized easily.

Multi sub-problems algorithm

The multi sub-problems algorithm is a natural one in the context of model switching. At each iteration we have a set of sub-problems with their corresponding Hessian approximations. We solve approximately the sub-problems to obtain a set of steps. Given that the purpose of our optimization is to minimize an objective function $f(x)$, we select the step that maximizes the decrease in the objective function. The multi sub-problems algorithm requires more than one objective function at each iteration, but it requires less iterations to converge.

Multi sub-problems with BHHH algorithm

This algorithm improves on the multi sub-problems algorithm by making use of the BHHH approach. In this algorithm, the iterative steps of the optimization process can be separated in two stages. In the first stage, the BHHH approximation is used to approximate the Hessian matrix. When the step length gets small enough, the iterations enter in a second stage, where the multi sub-problems algorithm is used to determine the step. Like the multi sub-problems algorithm, the multi sub-problems with BHHH is designed just for the trust-region method.

Adaptive line-search

Adaptive line-search algorithm is new line-search algorithm which is presented in our thesis as a new optimization algorithm to solve the MLE problem. This algorithm addresses the unstable behaviour of the original line-search algorithm in some complex cases. In our adaptive line-search algorithm, the length of search direction is adapted at each iteration like for the trust-region radius. The numerical results showed good convergence of this algorithm, better than the original line-search algorithm.

Application to discrete choice theory

Mixed logit models are currently very popular among practitioners in discrete choice theory. But they are numerically difficult to solve since they involve random parameters, which are usually assumed to be continuous, leading to choice probabilities that are multidimensional integrals. We have shown that the mixed logit problem can be seen as a maximum likelihood estimation problem where the objective function and its derivatives are very expensive to compute. Our algorithms have been developed and adapted to address these difficulties and they have been implemented with AMLET library. The numerical results exhibit favorable results, especially for the retrospective algorithm and the adaptive line-search, in comparison with standard approaches in non-linear non-convex programming.

Further research perspectives

Numerical results show the efficiency and competitiveness of our new algorithms in comparison with the basic optimization algorithms. We can adapt several methods to select the Hessian approximation. This can lead to new algorithms based on the general idea of model switching.

Analysing the switching criteria of the retrospective switching algorithms, we can see there is a problem if the set of Hessian approximations contains more than one matrix which satisfies the secant equation. To be more precise, consider the switching criteria of the retrospective approach

$$i^* = \arg\min_i |m_k^i(-s_{k-1}) - f(x_{k-1})| \tag{7.1}$$

where the quadratic model is defined based on the Hessian approximation matrix

$$m_k^i(p) = f(x_k) + p^T \nabla f(x_k) + \frac{1}{2} p^T H_k^i p.$$

If matrix H_k^i is updated by the secant equation $H_k^i s_{k-1} = y_{k-1}$, the approximation error becomes:

$$\begin{aligned}\delta_k^i(-s_{k-1}) &= |m_k^i(-s_{k-1}) - f(x_{k-1})| \\ &= |f(x_k) - s_{k-1}^T \nabla f(x_k) + \frac{1}{2} s_{k-1} y_{k-1} - f(x_{k-1})|\end{aligned}$$

and do not depended on index i, therefore (7.1) has not a unique solution. It implies that the retrospective algorithm can not run well if the set $\mathscr{H}_k^i$ contains several secant approximation matrices (we note that this issues does not happen with other algorithms). Unfortunately, the retrospective approach is the one that has performed the best, at least for our testing environment. In order to solve this issue, we propose another algorithm, more general by considering **several previous iterations**. Suppose that we have several successive and successful iterations before the current one:

$$f(x_{h_1}) > f(x_{h_2}) > \ldots > f(x_{h_\kappa}) > f(x_k)$$

The nearest previous point is used to evaluate the quadratic model:

$$\omega = \arg\min_{h_i} ||x_{h_i} - x_k||$$

If $\omega = h_\kappa$, to avoid the secant equation, the second nearest point can be chosen as:

$$\omega = \arg\min_{h_i | h_i < h_\kappa} ||x_{h_i} - x_k||$$

Under the retrospective approach, the Hessian approximation can be selected by taking the minimization of the approximation errors:

$$i^{retro} = \arg\min_{i} |m_k^i(x_\omega - x_k) - f(x_\omega)|.$$

We also consider another solution to solve this issue as a potential research direction that we will like to investigate. We are interested in some modified quasi-Newton approximations which no longer satisfy the secant equation. For example, a variational BFGS, proposed by Biggs [7], can be used:

$$H_{k+1} = H_k + \frac{1}{s_k^T y_k}\left\{\left(\frac{1}{t_k} + \frac{y_k^T H_k y_k}{s_k^T y_k}\right) s_k s_k^T - s_k y_k^T H_k - H_k y_k s_k^T\right\}$$

where

$$t_k = \frac{2}{s_k^T y_k}(f_k - f_{k+1} + s_k^T \nabla f_{k+1}).$$

It is noted that the performance of Biggs' update is better than the original BFGS update (see. for instance, Phua and Setiono (1992)[24]). Another variational secant equation can be considered. Xu and Zhang in [28] proposed a modified Quasi-Newton equation:

$$H_{k+1} s_k = \hat{y}_k$$

where

$$\hat{y}_k = (1 + \frac{\theta_k}{s_k^T y_k}),\ \theta_k = 6(f(x_k) - f(x_{k+1})) + 3(\nabla f(x_k) + \nabla f(x_{k+1})).$$

Another possible research investigation is where the switching criteria is considered based on the **condition number** of Hessian approximation. Note that the condition number of a matrix A can be computed from the eigenvalues of the matrix:

$$\kappa(A) = \left|\frac{\lambda_{max}(A)}{\lambda_{min}(A)}\right|$$

Where $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the maximal and the minimal eigenvalues of A respectively. Phua (1997) [14] proposed a switching algorithm to switch between BFGS and SR1 updates based on the condition number. The algorithm choose the Hessian approximation that minimizes the condition number. He also proposed a computationally inexpensive method for estimating the condition number of the BFGS and SR1 matrix. This idea can be applied for switching with other type of Hessian approximation (BHHH), but we have to note that the computation of the condition number $\kappa(BHHH)$ may be expensive, especially when the size of matrix becomes larger, then we will need an effective method for estimating its condition number. This is one of our future research perspective as well.

Recall that, beside the switching method, we also proposed a new line-search algorithm. The adaptive line-search needs more investigation before making a general conclusion. We expect, in the future, to investigate and develop a effective line-search algorithm based on the very first idea of adaptive line-search algorithm.

BIBLIOGRAPHY

[1] Fabian Bastin, Cinzia Cirillo, and Philippe L. Toint. Numerical experiments with AMLET, a new Monte-Carlo algorithm for estimating mixed logit models. Electronic Proceeding (CD-ROM) of the 10th International Conference on Travel Behaviour Research, 2003.

[2] Fabian Bastin, Cinzia Cirillo, and Philippe L. Toint. An adaptive Monte Carlo algorithm for computing mixed logit estimators. Computational Management Science, 3(1):55–79, 2006.

[3] Fabian Bastin, Cinzia Cirillo, and Philippe L. Toint. Application of an adaptive Monte Carlo algorithm to mixed logit estimation. Transportation Research Part B, 40(7):577–593, 2006.

[4] Fabian Bastin, Vincent Malmedy, Mélodie Mouffe, Philippe L. Toint, and Dimitri Tomanos. A retrospective trust-region method for unconstrained optimization. Mathematical Programming, 123(2):395–418, 2010.

[5] Moshe Ben-Akiva and Steven R. Lerman. Discrete Choice Analysis: Theory and Application to Travel Demand. The MIT Press, Cambridge, MA, USA, 1985.

[6] Ernst K. Berndt, Bronwyn H. Hall, Robert E. Hall, and Jerry A. Hausman. Estimation and inference in nonlinear structural models. Annals of Economic and Social Measurement, 3/4:653–665, 1974.

[7] M. C. BIGGS. Minimization algorithms making use of non-quadratic properties of the objective function. IMA Journal of Applied Mathematics, 8(3):315–327, 1971. doi: 10.1093/imamat/8.3.315. URL `http://imamat.oxfordjournals.org/content/8/3/315.abstract`.

[8] David S. Bunch. Maximum likelihood estimation of probabilistic choice models. SIAM J. Sci. Stat. Comp., 8(1):56–70, 1987.

[9] C. Cirillo and R. Xu. Forecasting cybercar use for airport ground access: A case study at bwi (baltimore washington international airport). Journal of Urban Planning and Development, 136(3):186–194, 2010.

[10] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Trust-Region Methods. SIAM, Philadelphia, USA, 2000.

[11] Andrew Daly. Estimating ”tree” logit models. Transportation Research Part B, 21 (4):251–268, 1987.

[12] J. Dennis and R. Schnabel. Least change secant updates for quasi-newton methods. SIAM, 21:443–459, October 1979.

[13] Electric Power Research Institute. Methodology for predicting the demand for new electricity-using goods. Final Report EA-593, Project 488-1, Electric Power Research Institute, Palo Alto, California, USA, 1977.

[14] Paul Kang Hoh Phua. Eigenvalues and switching algorithms for quasi-newton updates. Optimization, 42(3):185–217, 1997.

[15] R. Leadbetter, G. Lindgren, and H. Rootzén. Extremes and related properties of random sequences and processes. Springer series in statistics. Springer-Verlag, 1983. ISBN 9780387907314.

[16] R. Duncan Luce. Individual Choice Behavior: A Theoretical analysis. Wiley, New York, NY, USA, 1959.

[17] Daniel L. McFadden. Statistical Tools for Economists. Lecture Notes, Department of Economics, University of California, Berkeley, USA, 2000.

[18] Daniel L. McFadden and Kenneth Train. Consumers’ evaluation of new products: learning from self and others. Journal of Political Economy, 104(4):683–703, 1996.

[19] Daniel L. McFadden and Kenneth Train. Mixed MNL models for discrete response. Journal of Applied Econometrics, 15(5):447–270, 2000.

[20] Jorge J. Moré and David J. Thuente. Line search algorithms with guaranteed sufficient decrease. ACM Transactions on Mathematical Software, 20(3):286–307, 1994.

[21] David Munger, Pierre L'Ecuyer, Fabian Bastin, Cinzia Cirillo, and Bruno Tuffin. Estimation of the mixed logit likelihood function by randomized quasi-monte carlo. Transportation Research Part B, 46(2):305–320, 2012.

[22] Whitney K. Newey and Daniel McFadden. Large sample estimation and hypothesis testing. In R.F. Engle and D.L. McFadden, editors, Handbook of Econometrics, volume IV, chapter 36, pages 2111–2245. Elsevier, Amsterdam, The Netherlands, 1986.

[23] Jorge Nocedal and Stephen J. Wright. Numerical Optimization. Springer, New York, NY, USA, 1999.

[24] P.K.H. Phua and R. Setiono. Combined quasi-newton updates for unconstrained optimization. (no 41), 1992. URL `http://books.google.ca/books?id=yf7ZpwAACAAJ`.

[25] Alexander Shapiro. Stochastic programming by Monte Carlo simulation methods. SPEPS, 2000.

[26] Yosef Sheffi. Urban Transportation Networks. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1985.

[27] Kenneth Train. Discrete Choice Methods with Simulation. Cambridge University Press, New York, NY, USA, 2003.

[28] Chengxian Xu and Jianzhong Zhang. A survey of quasi-newton equations and quasi-newton methods for optimization. Annals of Operations Research, 103:213–234, 2001. ISSN 0254-5330.