**N° d'ordre : 41979**

**Université Lille 1 – Sciences et Technologies**

**Université Catholique de l'Afrique de l'Ouest - Unité Universitaire du Togo (UCAO-UUT)**

**THESE EN COTUTELLE**

Pour obtenir le grade de
**Docteur de l'Université Lille 1**

| | |
|---|---|
| Discipline : | Mathématiques Appliquées |
| Ecole doctorale : | Sciences Pour l'Ingénieur |

# On the consistency of some constrained maximum likelihood estimator used in crash data modelling

Présentée par

**Issa Cherif Geraldo**

soutenue le 15 décembre 2015 devant le jury composé de :

| | | |
|---|---|---|
| M. Célestin Kokonendji | Pr., Université de Franche-Comté | Président |
| M. Sergio Alvarez-Andrade | MdC, HDR, Université de Compiègne | Examinateur |
| M. Joseph Ngatchou-Wandji | Pr., Université Nancy 1 | Rapporteur |
| M. Abdallah Mkhadri | Pr., Université Cadi Ayyad (Maroc) | Rapporteur |
| M. Assi N'Guessan | MdC, HDR, Université Lille 1 | Directeur |
| M. Kossi Essona Gneyou | MdC, HDR, UCAO-UUT | Co-Directeur |

# Remerciements

Mes premiers remerciements iront à mes directeurs de thèse, Messieurs Assi N'Guessan et Kossi Gneyou. Malgré leurs nombreuses occupations, ils ont toujours su trouver le temps pour lire mes écrits. La rigueur scientifique dont ils ont fait preuve et le regard critique qu'ils ont toujours su porter sur mes travaux m'ont considérablement aidé tout au long de ces trois années de parcours. Je tiens à adresser une mention spéciale à Monsieur N'Guessan qui s'est démené pour me trouver les financements pour ma thèse.

Je remercie Messieurs Célestin Kokonendji et Sergio Alvarez-Andrade pour l'honneur qu'ils me font en acceptant le rôle d'examinateurs de cette thèse. Mes remerciements vont également à l'endroit de Messieurs Abdallah Mkhadri et Joseph Ngatchou-Wandji pour avoir accepté la mission de rapporteurs de cette thèse.

Je tiens à remercier Messieurs Komi Midzodzi Pekpe, Toyo Koffi Edarh-Bossou et François Messanvi Gbeassor pour toutes les aides qu'ils m'ont apportées tout au long de mon travail.

Je n'oublie pas mes parents et mes petits frères pour tout le soutien qu'ils m'ont apporté ainsi que l'amour dont ils m'ont entouré pendant toutes ces années.

Je tiens également à remercier Monsieur Marius Aduayi et son épouse, Monsieur Clément Hlomaschi et son épouse pour m'avoir hébergé lors de mes séjours en France. J'adresse ma gratitude à Monsieur François Recher pour tous ses conseils. Je n'oublie pas mes amis Djidula, Julien, Karim, Quentin 1, Quentin 2, Yinouss, Yasser pour tous les bons moments partagés lors de mes séjours à Lille.

Je remercie l'UCAO-UUT, le Laboratoire Paul Painlevé, l'Université Lille 1, le Laboratoire d'Accidentologie et de Biomécanique (Renault/PSA) et tous ceux qui ont contribué à la réalisation de ce travail. A tous et à chacun, je dis un sincère Merci.

# Résumé

Dans cette thèse, nous étudions la convergence de certains estimateurs appartenant partiellement au simplexe multidimensionnel et proposés dans la littérature pour combiner des données d'accidents de la route. Les estimateurs proposés sont issus de modèles probabilistes discrets ayant pour but de modéliser à la fois les risques d'accidents et l'effet de modifications des conditions de la route. Les composantes des estimateurs étant fonctionnellement dépendantes, il est d'usage d'utiliser les méthodes itératives classiques sous contraintes pour obtenir des valeurs numériques de ces estimateurs. Cependant, ces méthodes s'avèrent difficiles d'accès du fait du choix de points initiaux, de l'inversion des matrices d'information et de la convergence des solutions. Certains auteurs ont proposé un algorithme cyclique (CA) et itératif qui contourne ces difficultés mais les propriétés numériques et la convergence presque sûre de cet algorithme n'ont pas été abordées en profondeur. Cette thèse étudie, en profondeur, les propriétés de convergence numérique et théorique de cet algorithme. Sur le plan numérique, nous analysons les aspects les plus importants et les comparons à une large gamme d'algorithmes disponibles sous R et Matlab. Sur le plan théorique, nous démontrons la convergence presque sûre des estimateurs lorsque le nombre de données d'accidents tend vers l'infini. Des simulations de données d'accidents basées sur la loi uniforme, normale et de Dirichlet viennent illustrer les résultats. Enfin, nous étudions une extension de l'algorithme CA et obtenons des résultats théoriques de la convergence lorsque la même modification des conditions de la route est appliquée à plusieurs sites chacun comportant plusieurs types d'accidents.

**Mots clés.** Optimisation sous contraintes – Modèles statistiques discrets – Maximum de vraisemblance - Complément de Schur – Convergence – Accident de la route – Sécurité Routière.

# Abstract

In this thesis, we study the convergence of some estimators which partially belong to a multidimensional simplex and which can be found in literature to combine crash data. The proposed estimators are derived from discrete probabilistic models aiming at modelling both road accident risks and the effect of changes in the road conditions. The estimators' components being functionally dependent, classic iterative constrained methods are usually chosen to obtain numerical values of these estimators. However, these methods turn out to be difficult to use because of the choice of starting points, of the convergence of solutions and of the information matrix inversion. Some authors have suggested a cyclic iterative algorithm (CA) by-passing these difficulties but the numerical properties and the almost sure convergence of this algorithm have not been examined in depth. The theoretical and numerical convergence properties of this algorithm are thoroughly studied in this thesis. On the numerical level, we analyse the most important aspects and compare them to a wide range of available algorithms in R or Matlab. On the theoretical level, we prove the estimators' almost sure convergence when the number of crash data approaches infinity. Road accident data simulations based on the uniform, Gaussian and Dirichlet distributions illustrate the results. Finally, we study an extension of the CA algorithm and we get theoretical convergence results when the same road condition modification is applied to several areas, each with several types of accidents.

**Keywords** Constrained optimization – Discrete statistical models – Maximum likelihood - Schur complement – Convergence – Road accident – Road Safety.

# Contents

# Main abbreviations and notations

## Main abbreviations

| | |
|---|---|
| a.s. | almost sure / almost surely |
| BFGS | Broyden-Fletcher-Goldfarb-Shanno algorithm |
| CA | Cyclic Algorithm |
| EM | Expectation-Maximization algorithm |
| IP | Interior Point algorithm |
| MLE | Maximum Likelihood Estimator |
| MM | Majorization-Minimization or Minorization-Maximization (depending on the context) |
| MSE | Mean squared error |
| NR | Newton-Raphson's algorithm |
| NM | Nelder-Mead algorithm |

## Main notations

The multidimensional objects (vectors and matrices) are denoted by bold symbols while unidimensional objects are denoted by thin symbols.

### Important integers

| | |
|---|---|
| $s$ | Number of experimental sites |
| $r$ | Number of accidents types |
| $n$ | Total number of accidents at all sites |
| $n_k$ | Number of accidents at site $k$ (when $s > 1$) |

### Parameters

| | |
|---|---|
| $\boldsymbol{\beta}$ | Parameter vector of a statistical model |
| $\theta$ | Mean effect of a road safety measure at all sites |
| $\boldsymbol{\phi}$ | Vector of $rs$ components |
| $\phi_j$ | $j-$th component of $\boldsymbol{\phi}$ when $s = 1$ |
| $\boldsymbol{\phi}_k$ | Sub-vector of $\boldsymbol{\phi}$ with $r$ components when $s > 1$ |

| | |
|---|---|
| $\boldsymbol{\beta}^0$ | True value of parameter $\boldsymbol{\beta}$ in simulation studies |
| $\boldsymbol{\beta}^{(0)}$ | Starting value for parameter $\boldsymbol{\beta}$ in simulation studies |
| $\hat{\boldsymbol{\beta}}$ | MLE of $\boldsymbol{\beta}$ |
| $\boldsymbol{\pi}$, $\boldsymbol{\pi}_k$ | Vectors of class probabilities |

## Variables and observations

| | |
|---|---|
| $\mathbf{X}$ | Vector of $2rs$ random variables |
| $X_{ij}$ | Component of $\mathbf{X}$ when $s = 1$, $i = 1, 2$ and $j = 1, \ldots, r$ |
| $\mathbf{X}_k$ | Sub-vector of $\mathbf{X}$ with $2r$ random variables, crash counts per period and per type of accidents at site $k$ |
| $X_{ijk}$ | Component of $\mathbf{X}_k$ when $s > 1$, $i = 1, 2$ and $j = 1, \ldots, r$ |
| $\mathbf{x}$, $\mathbf{x}_k$ | Vectors of observed values of $\mathbf{X}$ and $\mathbf{X}_k$ |
| $x_{ij}$, $x_{ijk}$ | Observed values of $X_{ij}$ and $X_{ijk}$ |
| $\mathbf{Z}$ | Vector with $r$ control coefficients (when $s = 1$) |
| $\mathbf{Z}_k$ | Vector with $r$ control coefficients linked to site $k$ (when $s > 1$) |

## Classical distributions

| | |
|---|---|
| $\mathcal{U}()$ | Uniform distribution |
| $\mathcal{N}()$ | Gaussian distribution |
| $\mathcal{M}()$ | Multinomial distribution |
| $\mathcal{P}()$ | Poisson distribution |
| Dir() | Dirichlet distribution |

# General introduction

Most of statistical methods used for data modelling require very often the estimation of an unknown parameter vector $\boldsymbol{\beta} \in \mathbb{R}^d$ where $d$ is a positive integer. To the best of our knowledge, Maximum Likelihood Estimation (MLE) [1, 88] is the most widely used method for such an estimation. If $\mathbf{X}$ denotes a random vector that describes the data, it consists in two steps described as follows: first, a probability distribution $\mathrm{P}_{\boldsymbol{\beta}}$ depending on the parameter vector $\boldsymbol{\beta}$ is assigned to $\mathbf{X}$ and secondly ones calculates the log-likelihood function

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log \mathrm{P}_{\boldsymbol{\beta}}(\mathbf{X}_i = \mathbf{x}_i)$$

where $\mathbf{X}_1, \dots, \mathbf{X}_n$ is an independent identically distributed (i.i.d) sample from $\mathrm{P}_{\boldsymbol{\beta}}$. The Maximum Likelihood Estimator of $\boldsymbol{\beta}$, denoted $\hat{\boldsymbol{\beta}}$, is obtained as a solution of the maximization problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{S}}{\operatorname{argmax}}\, \ell(\boldsymbol{\beta})$$

where $\mathbb{S} \subset \mathbb{R}^d$ is the parameter space, i.e. the set of possible values for $\boldsymbol{\beta}$. This set may be defined by some constraints of the form

$$\boldsymbol{\beta} \in \mathbb{S}, \quad g(\boldsymbol{\beta}) = 0 \quad \text{or} \quad h(\boldsymbol{\beta}) \geqslant 0.$$

The resolution of this maximization problem can be done by solving the likelihood equations

$$\nabla \ell(\hat{\boldsymbol{\beta}}) = \mathbf{0}$$

where $\nabla \ell(\hat{\boldsymbol{\beta}})$ is the gradient of $\ell$ evaluated at $\hat{\boldsymbol{\beta}}$. However, in most practical problems, it is almost impossible to find closed-form expressions to the solutions of the likelihood equations and one then uses an iterative method that computes successive approximations of the solution using the scheme

$$\boldsymbol{\beta}^{(k+1)} = A(\boldsymbol{\beta}^{(k)})$$

where $A$ is a mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$ and $\boldsymbol{\beta}^{(0)}$ is a given starting point. Convergence of the iterative method is declared using a proper criterion such as the fact that two successive values of the objective function $\ell$ are close enough i.e. $|\ell(\boldsymbol{\beta}^{(k+1)}) - \ell(\boldsymbol{\beta}^{(k)})| < \epsilon$ for $\epsilon > 0$.

The most widely used iterative method is certainly the Newton-Raphson's method that computes successive approximations using the formula

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \nabla^2 \ell(\boldsymbol{\beta}^{(k)})^{-1} \nabla \ell(\boldsymbol{\beta}^{(k)})$$

where $\nabla \ell(\boldsymbol{\beta}^{(k)})$ is the gradient of the log-likelihood (also called score) and $\nabla^2 \ell(\boldsymbol{\beta}^{(k)})$ is the Hessian matrix of the log-likelihood. The Newton-Raphson's method is known to converge quickly to the true unknown value if the starting point $\boldsymbol{\beta}^{(0)}$ is close to the true value. In addition to the difficulty to choose the starting point, several other complications can compromise the performance of this algorithm: (a) the function $\ell(\boldsymbol{\beta})$ can have a complex form, (b) it can be costly to evaluate the Hessian matrix, (c) numerical inversion can be complicated for an ill-conditioned Hessian matrix, (d) in high dimensions, it is costly to invert the Hessian matrix and (e) existence of constraints on the parameters can introduce some further complications. Moreover, the Newton's method can diverge violently when the starting point $\boldsymbol{\beta}^{(0)}$ is far from the true parameter value.

In order to cope with all these defects, other iterative procedures such as Fisher's scoring, quasi-Newton or Derivative-Free Optimization (DFO) algorithms have been proposed. The last decades have also seen the development of new classes of algorithms such that MM [36, 43, 44] and EM [20] algorithms. Actually, MM and EM are rather principles for constructing optimization algorithms. Since they don't need any matrix inversion, they are considered as relevant for high dimension problems and for some discrete multivariate distributions [108].

Recently, N'Guessan and Truffier [78] and N'Guessan [71] proposed a cyclic iterative algorithm (CA) for constrained maximum likelihood estimation of a discrete multivariate distribution and they applied it to the statistical analysis of accidents data. However, the properties of the CA have not been studied have not been examined in depth. This works aims two main objectives: (a) study convergence properties of the cyclic algorithm on both numerical and theoretical levels and (b) generalize this latter algorithm to more general models like that of N'Guessan et al. [73] and compare the performance of the generalized CA to those of its competitors (Newton, MM, quasi-Newton, etc. . . ).

This manuscript is organized in four (04) chapters.

Chapter 1 is divided into three main parts. In the first part, we present the general principles of unconstrained optimization. We remind necessary and sufficient conditions for existence of an optimum and review some of the best unconstrained optimization algorithms. The classical Newton's algorithm and some of its modifications such as the Fisher's scoring and the quasi-Newton algorithms are presented. We also describe the MM and EM algorithms that are principles for constructing optimization algorithms. We also mention derivative-free algorithms such as the Nelder-Mead's algorithm as well as some special algorithms designed for least-squares estimation. The second part of the chapter is devoted to constrained optimization and how are constraints dealt with in numerical procedures.

In the last part of the chapter, we review some statistical models for crash data and we present the specific model that serves as basis for the constrained optimization algorithm studied in the second chapter of this thesis.

The second chapter concerns the Cyclic Algorithm constructed by N'Guessan [71] in the framework of maximum likelihood estimation applied to statistical analysis of crash data. After a bibliographical review of the algorithm, we study its convergence properties on both numerical and theoretical levels. On the numerical level, we investigate some important features of the CA such as the number of iterations, the influence of the initial guess, the computation time and we compare it with the most performing algorithms such as MM algorithms, Newton-Raphsons's algorithms and other algorithms available on R and MATLAB software. On the theoretical level, we investigate the convergence of the CA to a stationary point of the log-likelihood and its ascent property (i.e. the fact that the log-likelihood increases at each iteration of the algorithm).

In Chapter 3, after an overview of the consistency of the MLE including basic definitions, we give a theoretical proof of the strong consistency of the MLE in the crash control model. The chapter ends with some numerical illustrations of our main results.

In Chapter 4, we construct a cyclic algorithm to estimate the parameters of a model that is a generalization of the one considered in Chapter 2. We prove that the generalized CA has the properties of the algorithm presented in Chapter 2, that is, the generalized CA is an ascent algorithm and it converges to the MLE from any starting point. We finally report on some numerical convergence properties of the CA. We also compare the performance of the generalized CA with that of some algorithms like MM algorithm, quasi-Newton BFGS algorithm, Nelder-Mead's algorithm and Newton's method.

# Chapter 1

# On optimization algorithms and crash data models

## 1.1 Introduction

This review chapter is motivated by the general observation that most problems in parametric statistics involve optimization of a function. In general, after the modelling step, the statistician is faced with the delicate task of estimating the underlying parameters of his model. This can be done using maximum likelihood or least squares estimation that are considered as the dominant forms of estimation in statistics. Except from few problems designed for illustration purpose and that can be solved analytically, most practical maximum likelihood and least squares estimation problems must be solved numerically and this led to the greater and greater importance of computational statistics. A numerical algorithm for estimating a parameter $\boldsymbol{\beta} \in \mathbb{R}^d$ consists in an iterative scheme of the form

$$\boldsymbol{\beta}^{(k+1)} = A(\boldsymbol{\beta}^{(k)}), \qquad k = 0, 1, \ldots$$

where $A$ is a mapping from $\mathbb{R}^d$ into itself and $\boldsymbol{\beta}^{(0)}$ is set by the user as the starting point. The iterations continue until a stopping criterion is satisfied.

The very first optimization algorithm that comes to mind is the Newton-Raphson's algorithm (NR). It is an efficient algorithm that converges quickly to the desired value when the starting point is close to true unknown parameter. But when it is started far from the true parameter, the NR can simply fail to converge. Another of its drawbacks is that NR requires computation and inversion of second derivatives matrices and this can become very complicated in high dimension or in ill-conditioned problems. For all these reasons, scientific researches have developed a plethora of remedies either by modifying the NR approach or by considering new ones. But all this comes at the cost of greater implementation complexity and no class of algorithms should be neglected a priori when dealing with a new problem. When one fails, it can be replaced by a more efficient one.

Optimization theory distinguishes unconstrained optimization from constrained optimization. And this chapter does not escape to this rule. Our review of optimization algorithms is then divided into two parts: the first one presents the general principles of unconstrained optimization and the classical algorithms (Newton, quasi-Newton, MM and EM algorithms, derivative free algorithms, ... ) and the second part deals with generalities on constrained optimization and its implementation in numerical procedures. For more details, refer to [2, 5, 21, 29, 43, 44, 45, 66, 79].

Since the optimization algorithm considered in the remainder of this thesis is inspired from the maximum likelihood estimation of the parameters of the model proposed by N'Guessan et al. [73] for the analysis of the effect of a road safety measure, a section of this review chapter is also dedicated to the review of some statistical models for crash data.

In all the sequel, we adopt the following notations. For a continuously differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, the column vector of the first partial derivatives of $f$ with respect to the $d$ variables also called the gradient of $f$ at $\boldsymbol{\beta}$ is denoted by $\nabla f(\boldsymbol{\beta})$. The matrix of second partial derivatives (the Hessian matrix) is denoted by $\nabla^2 f(\boldsymbol{\beta})$. For a continuously differentiable function $F : \mathbb{R}^d \to \mathbb{R}^d$, the matrix of first partial derivatives called the Jacobian matrix of $F$ is denoted by $\mathbf{J}F(\boldsymbol{\beta})$.

## 1.2 Numerical algorithms for unconstrained optimization

### 1.2.1 Fundamentals of unconstrained optimization

In unconstrained optimization, one seeks to minimize (or maximize) an objective function $f$ that depends on a real vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$, $d \geqslant 1$, with no restrictions at the values of the components of $\boldsymbol{\beta}$. Since maximizing $f$ is equivalent to minimizing $-f$, we consider without loss of generality the mathematical formulation

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} f(\boldsymbol{\beta}). \tag{1.1}$$

Before more details on unconstrained optimization, let us give the following well known example.

**Example 1.2.1** (Maximum likelihood estimation)**.** *Let $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ be i.i.d observations from one of a family of distributions indexed by a $d-$dimensional parameter $\boldsymbol{\beta}$. The likelihood function of the sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ is the product of the densities $f(\cdot \mid \boldsymbol{\beta})$ evaluated at the observations, treated as a function of the parameter $\boldsymbol{\beta}$:*

$$L_n(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(\mathbf{X}_i \mid \boldsymbol{\beta});$$

*the (natural) logarithm of the likelihood function, called the log-likelihood function, is denoted by*

$$\ell_n(\boldsymbol{\beta}) = \log L_n(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log f(\mathbf{X}_i \mid \boldsymbol{\beta}).$$

*If it exists, any vector $\hat{\boldsymbol{\beta}}$ at which $\ell_n(\boldsymbol{\beta})$ attains its maximum is called Maximum Likelihood Estimator (MLE) of $\boldsymbol{\beta}$.*

There are two possible types of solutions to problem (1.1).

**Definition 1.2.1.** A solution $\boldsymbol{\beta}^*$ of problem (1.1) is

– a local minimizer (resp. strict local minimizer) if there is a neighbourhood $\mathbb{V}$ of $\boldsymbol{\beta}^*$ such that $f(\boldsymbol{\beta}^*) \leqslant f(\boldsymbol{\beta})$ for all $\boldsymbol{\beta} \in \mathbb{V}$ (resp. $f(\boldsymbol{\beta}^*) < f(\boldsymbol{\beta})$ for all $\boldsymbol{\beta} \in \mathbb{V}$ and $\boldsymbol{\beta} \neq \boldsymbol{\beta}^*$);

– a global minimizer if $f(\boldsymbol{\beta}^*) \leqslant f(\boldsymbol{\beta})$ for all $\boldsymbol{\beta} \in \mathbb{R}^d$.

**Remark 1.2.1.** Global minimizers are certainly desirable but can be difficult to find. This is especially the case when $f$ is a function with many local minimizers. So, most algorithms are able to find only a local minimizer [79].

The following theorem ([21, p. 81] or [79, p. 14]) gives a necessary condition for a point $\boldsymbol{\beta}$ to be a local minimizer.

**Theorem 1.2.1** (First-Order Necessary Conditions). *If $\boldsymbol{\beta}$ is a local minimizer and $f$ is continuously differentiable in a neighbourhood of $\boldsymbol{\beta}$, then $\nabla f(\boldsymbol{\beta}) = \mathbf{0}$.*

A point $\boldsymbol{\beta}^*$ such that $\nabla f(\boldsymbol{\beta}^*) = 0$ is called *stationary point* of $f$. So according to the theorem, to find a local minimizer of $f$, one must look for the stationary points (if they exist) of the function $f$. However, since the condition is not sufficient, a stationary point is not always a minimizer. There exists a second necessary condition given by the following theorem [79, p. 14].

**Theorem 1.2.2** (Second-Order Necessary Conditions). *If $\boldsymbol{\beta}^*$ is a local minimizer of $f$ and if the Hessian matrix $\nabla^2 f$ exists and is continuous in a neighbourhood of $\boldsymbol{\beta}^*$ then $\nabla f(\boldsymbol{\beta}^*) = \mathbf{0}$ and the Hessian matrix $\nabla^2 f(\boldsymbol{\beta}^*)$ is positive semi-definite*[1].

A slight strengthening of the second-order condition gives a sufficient condition for a stationary point to be a local minimum [43, p. 160].

**Theorem 1.2.3** (Second-order sufficient condition). *If $\boldsymbol{\beta}^*$ is a stationary point and if $\nabla^2 f(\boldsymbol{\beta}^*)$ is positive definite then $\boldsymbol{\beta}^*$ is a local minimum.*

**Remark 1.2.2.** Note that the second-order sufficient conditions are not necessary. That is, a point $\boldsymbol{\beta}^*$ may be a local minimizer, but may fail to satisfy the sufficient conditions. A simple example is given by the function $f(\beta_1, \beta_2) = (\beta_1)^4 + (\beta_2)^4$ for which the point $(0, 0)$ is a local minimizer at which the Hessian matrix vanishes (and is therefore not positive definite).

**Remark 1.2.3.** The results presented above provide the foundations for unconstrained optimization algorithms since all algorithms seek a point where $\nabla f$ vanishes (equals zero).

### Convex optimization

When the objective function $f$ is convex, local and global minimizers are confounded [79, p. 16].

[1]Recall that a matrix $\mathbf{B} \in \mathbb{R}^{d\times d}$ is said to be positive definite (resp. semi-definite) if for all $\mathbf{x} \neq 0$, $\mathbf{x}^T\mathbf{B}\mathbf{x} > 0$ (resp. $\mathbf{x}^T\mathbf{B}\mathbf{x} \geqslant 0$).

### Rate of convergence

The rate of convergence is one of the key measures of performance of an algorithm. We remind the different types of convergence of an iterative algorithm. We refer the reader to [79, p. 619] for more details.

**Definition 1.2.2.** Let $(\boldsymbol{\beta}^{(k)})$ be a sequence of vectors in $\mathbb{R}^d$ converging to $\boldsymbol{\beta}^*$. The convergence is said to be

– *Q-linear*[2] if there is a real $r \in ]0, 1[$ such that

$$\frac{\|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^*\|}{\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^*\|} \leqslant r, \quad k \to +\infty.$$

– *Q-superlinear* if

$$\lim_{k\to+\infty} \frac{\|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^*\|}{\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^*\|} = 0,$$

– *Q-quadratic* if there exists a positive constant $M$ such that

$$\frac{\|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^*\|}{\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^*\|^2} \leqslant M.$$

## 1.2.2 Newton's algorithm and Fisher scoring

Newton's algorithm (also called Newton-Raphson's algorithm) is by far the most important and the most used algorithm. Since the minimum points belong to the set of stationary points, we first present Newton's method for solving a non-linear equation of the form $F(\boldsymbol{\beta}) = \mathbf{0}$ and then we present how it can be applied to solve optimization problems.

### 1.2.2.1 Newton's method for solving $F(\boldsymbol{\beta}) = \mathbf{0}$

Let us consider the equation

$$F(\boldsymbol{\beta}) = \mathbf{0} \tag{1.2}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)^T \in \mathbb{R}^d$ and $F$ is a function from $\mathbb{R}^d$ to $\mathbb{R}^d$ defined by $F(\boldsymbol{\beta}) = (F_1(\boldsymbol{\beta}), \ldots, F_d(\boldsymbol{\beta}))^T$. Note that both $\boldsymbol{\beta}$ and $F(\boldsymbol{\beta})$ are vectors of the same length $d$. Also note that if $F(\boldsymbol{\beta})$ is a linear function and if the system is non-singular, it is possible to find a solution by using the classical methods for solving linear systems of equations. So the presentation below focuses on the non-linear case.

Except for a few isolated special cases, it is not possible to guarantee that a solution can be found to Equation (1.2), nor is it possible to give analytic (closed-form) expression of the solution. Newton's method for solving Equation (1.2) is an iterative method. Given an estimate of the solution $\boldsymbol{\beta}^{(k)}$, the next iterate $\boldsymbol{\beta}^{(k+1)}$ is computed by approximating the

[2]The prefix Q indicates that this definition involves a quotient.

function $F$ by the linear function consisting of the first two terms of the Taylor series for the function $F$ at the point $\boldsymbol{\beta}^{(k)}$:

$$F(\boldsymbol{\beta}^{(k)} + \mathbf{p}) = F(\boldsymbol{\beta}^{(k)}) + \mathbf{J}F(\boldsymbol{\beta}^{(k)})\ \mathbf{p} + \mathrm{o}(\|\mathbf{p}\|)$$

where $\mathbf{J}F$ is the Jacobian matrix of $F$. By neglecting the term $\mathrm{o}(\|\mathbf{p}\|)$, we can make the approximation

$$F(\boldsymbol{\beta}^{(k)} + \mathbf{p}) \simeq F(\boldsymbol{\beta}^{(k)}) + \mathbf{J}F(\boldsymbol{\beta}^{(k)})\ \mathbf{p}.$$

If we want to have $F(\boldsymbol{\beta}^{(k)}+\mathbf{p}) = \mathbf{0}$, it is sufficient to take the step $\mathbf{p}$, whenever it is possible, as

$$\mathbf{p} = -\left(\mathbf{J}F(\boldsymbol{\beta}^{(k)})\right)^{-1}\ F(\boldsymbol{\beta}^{(k)})$$

when the matrix $\mathbf{J}F(\boldsymbol{\beta}^{(k)})$ is invertible. The Newton's iterative algorithm for solving (1.2) is given then by

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \left(\mathbf{J}F(\boldsymbol{\beta}^{(k)})\right)^{-1}\ F(\boldsymbol{\beta}^{(k)}). \tag{1.3}$$

**Remark 1.2.4.** It is usual to replace formula (1.3) by the formulas

$$\begin{aligned} \mathbf{J}F(\boldsymbol{\beta}^{(k)})\ \mathbf{p} &= -F(\boldsymbol{\beta}^{(k)}) \\ \boldsymbol{\beta}^{(k+1)} &= \boldsymbol{\beta}^{(k)} + \mathbf{p} \end{aligned} \tag{1.4}$$

but this does not fundamentally change the problem. Indeed, the reliability of the value of $\mathbf{p}$ depends on the regularity of the matrix $\mathbf{J}F(\boldsymbol{\beta}^{(k)})$.

#### 1.2.2.2 Newton's method for optimization

In the case of an optimization problem, we seek to solve the equation $\nabla f = 0$ where the Jacobian matrix $\mathbf{J}(\nabla f)$ is equal to $\nabla^2 f$, the Hessian matrix of $f$. So Newton's optimization method is the iterative scheme

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \left(\nabla^2 f(\boldsymbol{\beta}^{(k)})\right)^{-1}\ \nabla f(\boldsymbol{\beta}^{(k)}). \tag{1.5}$$

The fact that Newton's method depends on the Taylor series for $f$ suggests that it is a method particularly relevant in a neighbourhood of a solution $\boldsymbol{\beta}^*$. And this is its main advantage: if the starting point $\boldsymbol{\beta}^{(0)}$ is close enough to a solution $\boldsymbol{\beta}^*$, and $\nabla^2 f(\boldsymbol{\beta}^*)$ is non-singular, then the sequence $(\boldsymbol{\beta}^{(k)})$ obtained from the Newton's algorithm (1.5) converges $Q-$quadratically to $\boldsymbol{\beta}^*$. This is more formally stated by the following theorem:

**Theorem 1.2.4** (Local convergence of Newton's method [29])**.** *Assume that $f \in \mathcal{C}^3(\mathbb{R}^d, \mathbb{R})$, that is, the real function $f$ admits three continuous derivatives. Assume that $\boldsymbol{\beta}^*$ satisfies $\nabla f(\boldsymbol{\beta}^*) = \mathbf{0}$ with $\nabla^2 f(\boldsymbol{\beta}^*)$ non-singular. If $\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\|$ is sufficiently small, then the sequence defined by*

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \left(\nabla^2 f(\boldsymbol{\beta}^{(k)})\right)^{-1}\ \nabla f(\boldsymbol{\beta}^{(k)})$$

*converges quadratically to $\boldsymbol{\beta}^*$.*

The Newton's method has several drawbacks. The first is that it can diverge violently when the starting point $\boldsymbol{\beta}^{(0)}$ is far from the true value. This is illustrated in the case of multidimensional non-linear equations by the following example.

**Example 1.2.2** (Divergence of Newton's method [21]). *Let us consider $\boldsymbol{\beta} = (\beta_1, \beta_2)^T$ and*

$$F(\boldsymbol{\beta}) = \begin{pmatrix} \mathrm{e}^{\beta_1} - 1 \\ \mathrm{e}^{\beta_2} - 1 \end{pmatrix}.$$

*The unique root of $F$ is $\boldsymbol{\beta}^* = (0,0)^T$. If one starts the Newton's algorithm from $\boldsymbol{\beta}^{(0)} = (-10,-10)^T$ then $\boldsymbol{\beta}^{(1)} = (2.2 \times 10^4, 2.2 \times 10^4)^T$. The Newton's algorithm diverges because the Jacobian matrix*

$$\mathbf{J}F(\boldsymbol{\beta}^{(1)}) = \begin{pmatrix} \mathrm{e}^{2.2\times10^4} & 0 \\ 0 & \mathrm{e}^{2.2\times10^4} \end{pmatrix}$$

*has its diagonal elements close to infinity and can not be numerically evaluated.*

The second major drawback is that it requires at each iteration the evaluation and the inversion of the Hessian matrix. This is heavy not only because the evaluation of second derivatives can introduce some complicated intermediate calculations but also because numerical inversion of the Hessian matrix $\nabla^2 f(\boldsymbol{\beta}^{(k)})$ can be unreliable when this latter is ill-conditioned.

A third major drawback is that this method is not a descent method i.e. nothing guarantees that we always have $f(\boldsymbol{\beta}^{(k+1)}) \leqslant f(\boldsymbol{\beta}^{(k)})$ in minimization problems. The first-order approximation

$$\begin{aligned} f(\boldsymbol{\beta}^{(k+1)}) &= f\left(\boldsymbol{\beta}^{(k)} - [\nabla^2 f(\boldsymbol{\beta}^{(k)})]^{-1} \, \nabla f(\boldsymbol{\beta}^{(k)})\right) \\ &\simeq f(\boldsymbol{\beta}^{(k)}) - [\nabla f(\boldsymbol{\beta}^{(k)})]^T \, [\nabla^2 f(\boldsymbol{\beta}^{(k)})]^{-1} \, \nabla f(\boldsymbol{\beta}^{(k)}) \end{aligned}$$

shows that the descent property $f(\boldsymbol{\beta}^{(k+1)}) \leqslant f(\boldsymbol{\beta}^{(k)})$ holds only if the Hessian matrix $\nabla^2 f(\boldsymbol{\beta}^{(k)})$ is definite positive.

#### 1.2.2.3 Fisher's scoring

There exist many algorithms that have been proposed as remedies to the Newton's algorithm when it fails. One of them is the Fisher's scoring algorithm [83] that replaces the Hessian matrix by its expected value. Since Fisher's scoring is a purely statistical approach, let us first remind some of the standard vocabulary related to the likelihood function $\ell(\boldsymbol{\beta})$. In statistics, the gradient of the log-likelihood $\nabla\ell(\boldsymbol{\beta})$ is called the *score* and the negative of its Hessian matrix $-\nabla^2\ell(\boldsymbol{\beta})$ is called the *observed information matrix.* The Fisher's scoring for maximization of the log-likelihood consists in replacing the observed information matrix by the expected information matrix $\mathbf{J}(\boldsymbol{\beta}) = \mathrm{E}[-\nabla^2\ell(\boldsymbol{\beta})]$. It corresponds to the following iterative scheme:

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{J}(\boldsymbol{\beta}^{(k)})^{-1} \, \nabla\ell(\boldsymbol{\beta}^{(k)}). \tag{1.6}$$

It can be shown (see [88]) that $\mathrm{E}(\nabla\ell(\boldsymbol{\beta})) = \mathbf{0}$ and

$$\mathbf{J}(\boldsymbol{\beta}) = \mathrm{E}\Big(\nabla\ell(\boldsymbol{\beta})\ (\nabla\ell(\boldsymbol{\beta}))^T\Big). \tag{1.7}$$

Thus, $\mathbf{J}(\boldsymbol{\beta}) = \mathrm{var}(\nabla\ell(\boldsymbol{\beta}))$ is the covariance matrix of the score and therefore positive semi-definite. The fact that a covariance matrix is always positive semi-definite is a classical result. Indeed if $\boldsymbol{\Gamma}$ is the covariance matrix of a random vector $\mathbf{T} = (T_1, \dots, T_d)$ then for all vector $\mathbf{x} = (x_1, \dots, x_d)^T$, we have

$$\mathbf{x}^T\boldsymbol{\Gamma}\mathbf{x} = \mathrm{var}\Big(\sum_{i=1}^{d} x_i T_i\Big) \geqslant 0.$$

From the first-order approximation of $\ell$ in $\boldsymbol{\beta}^{(k+1)}$ we deduce that

$$\begin{aligned} \ell(\boldsymbol{\beta}^{(k+1)}) - \ell(\boldsymbol{\beta}^{(k)}) &\approx \nabla\ell(\boldsymbol{\beta}^{k})^T(\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}) \\ &\approx \nabla\ell(\boldsymbol{\beta}^{k})^T\mathbf{J}(\boldsymbol{\beta}^{(k)})^{-1}\ \nabla\ell(\boldsymbol{\beta}^{(k)}) \quad \geqslant 0 \end{aligned} \tag{1.8}$$

and it is clear that Fisher's scoring is an ascent algorithm (for maximization of a likelihood).

From the representation (1.7), it is easily seen that scoring requires only first derivatives information for its implementation. Another benefit of scoring is that the inverse matrix $\mathbf{J}(\boldsymbol{\beta})^{-1}$ immediately supplies the asymptotic variances and covariances of the maximum likelihood estimate $\hat{\boldsymbol{\beta}}$ [88]. Fisher scoring is linearly convergent, at a rate which depends on the relative difference between observed and expected information [95]. However it requires the computation of expectations and this could be difficult in some cases. Fortunately, most of the distribution families commonly encountered in statistics are exponential families and the results needed are known exactly [43].

### 1.2.3 Quasi-Newton algorithms

Unlike the Fisher's scoring that is restricted to maximization of a likelihood function, quasi-Newton algorithms can be applied not only to maximum likelihood estimation but also to any optimization problem in statistics or in another domain. Their main characteristic is that they compute an approximation $\mathbf{B}_k$ of the Hessian matrix $\nabla^2\ell(\boldsymbol{\beta}^{(k)})$ at the iteration $k$. By doing so, they avoid the computation and the inversion of the Hessian matrix at each iteration. Note that quasi-Newton algorithms can be used both for optimization and resolution of non-linear systems of equation. The presentation below focuses on the case of optimization.

Quasi-Newton algorithms use the first-order Taylor approximation

$$\nabla\ell(\boldsymbol{\beta}^{(k)}) - \nabla\ell(\boldsymbol{\beta}^{k+1}) \approx \nabla^2\ell(\boldsymbol{\beta}^{(k+1)})(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(k+1)}). \tag{1.9}$$

Before going further we define the gradient and argument differences that will appear repeatedly in the remainder of this section:

$$\begin{aligned} \mathbf{s}_k &= \boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)} \\ \mathbf{g}_k &= \nabla\ell(\boldsymbol{\beta}^{(k+1)}) - \nabla\ell(\boldsymbol{\beta}^{(k)}). \end{aligned}$$

Then from the approximation (1.9), the approximation $\mathbf{B}_{k+1}$ of the Hessian matrix $\nabla^2\ell(\boldsymbol{\beta}^{(k+1)})$ should satisfy the equation

$$\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{g}_k \tag{1.10}$$

called *secant equation.*

**Remark 1.2.5.** Most of the quasi-Newton methods have the form

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \mathbf{M}_k$$

where the matrix $\mathbf{M}_k$ represents an *update* to the old approximation $\mathbf{B}_k$ and so a quasi-Newton approximation is often referred to as an *update formula.*

If $\boldsymbol{\beta}$ is a vector of $d$ components, then the secant equation represents a system of $d$ equations with $d^2$ unknowns that are the components of $\mathbf{B}_{k+1}$. Except in the case $d = 1$, this condition by itself is insufficient to define $\mathbf{B}_{k+1}$ uniquely and additional conditions are needed. These conditions are usually properties of the Hessian matrix that we would like the approximation to share. One of them is symmetry (motivated by symmetry of the exact Hessian) and another is the requirement that the differences between successive approximations $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$ have low rank. Davidon [15] proved that the only rank-one update formula that preserves symmetry is

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{g}_k - \mathbf{B}_k\mathbf{s}_k)(\mathbf{g}_k - \mathbf{B}_k\mathbf{s}_k)^T}{(\mathbf{g}_k - \mathbf{B}_k\mathbf{s}_k)^T\mathbf{s}_k}.$$

This formula is called the *symmetric rank-one update formula* since the update term is a matrix of rank one (its row are proportional).

In addition to symmetry, it seems reasonable to ask that the matrices $\mathbf{B}_{k+1}$ be positive definite as well. This will also guarantee the descent property of the quasi-Newton method. A last condition that we found in [79, chap. 6] is that $\mathbf{B}_{k+1}$ must be the closest possible to the matrix $\mathbf{B}_k$ i.e. $\mathbf{B}_{k+1}$ minimizes the distance $\|\mathbf{B} - \mathbf{B}_k\|$. There is no rank-one update formula that maintains both symmetry and positive definiteness of the Hessian approximations [29]. However, there are many rank-two update formulas that do. The two most popular rank-two updates are the Broyden-Fletcher-Goldfarb-Shanno (BFGS) and Davidon-Fletcher-Powell (DFP) updates formulas. In the DFP formula, the matrix $\mathbf{B}_{k+1}$ is given by

$$\mathbf{B}_{k+1} = (\mathbf{I} - \rho_k\mathbf{g}_k\mathbf{s}_k^T)\mathbf{B}_k(\mathbf{I} - \rho_k\mathbf{s}_k\mathbf{g}_k^T) + \rho_k\mathbf{g}_k\mathbf{g}_k^T, \tag{1.11}$$

where

$$\rho_k = \frac{1}{\mathbf{g}_k^T\mathbf{s}_k}.$$

In practice, one directly updates the inverse $\mathbf{H}_k = \mathbf{B}_k^{-1}$ using the formula (see [79] for details)

$$\mathbf{H}_{k+1} = \mathbf{H}_k - \frac{\mathbf{H}_k\mathbf{g}_k\mathbf{g}_k^T\mathbf{H}_k}{\mathbf{g}_k^T\mathbf{H}_k\mathbf{g}_k} + \frac{\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{g}_k^T\mathbf{s}_k}. \tag{1.12}$$

This formula is considered as a rank-two perturbation of the matrix $\mathbf{H}_k$ since the last two terms are rank-one matrices. The DFP updating formula is quite effective but it only ranks second behind the BFGS update formula that is considered as the most effective quasi-Newton formula [45, 79]. Its consists in updating the matrix $\mathbf{B}_{k+1}$ using the formula

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{g}_k \mathbf{g}_k^T}{\mathbf{g}_k^T \mathbf{s}_k} \tag{1.13}$$

or directly updating its inverse $\mathbf{H}_{k+1} = \mathbf{B}_{k+1}^{-1}$ by

$$\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{g}_k^T)\mathbf{H}_k(\mathbf{I} - \rho_k \mathbf{g}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T \tag{1.14}$$

where $\rho_k$ is given by (1.12).

Quasi-Newton methods have several advantages over the Newton's method. They require only the gradient of the objective function. The second advantage is that the approximation of the Hessian matrix can be positive definite in order to guarantee the descent property in minimization problems. As far as the convergence rate is concerned, Nocedal and Wright [79] show under some regularity conditions on the objective function $f$, that BFGS converges superlinearly.

Quasi-Newton methods are considered as effective in solving a wide variety of small to mid-size problems. In cases when the number of variables is large, other methods like the limited-memory quasi-Newton methods [29, chap. 13] may be preferred.

### 1.2.4 Line search methods

Newton's method and its modifications belong to the general class of *line search algorithms*. The iterations of a line search method are given by

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \alpha_k \mathbf{p}^{(k)} \tag{1.15}$$

where $\alpha_k > 0$ (called step length) and $\mathbf{p}^{(k)}$ (called the descent direction) is chosen in order to have $(\mathbf{p}^{(k)})^T \nabla f(\boldsymbol{\beta}^{(k)}) < 0$ (this property ensures that $f(\boldsymbol{\beta}^{(k+1)}) < f(\boldsymbol{\beta}^{(k)})$ in minimization problems). Generally one takes

$$\mathbf{p}^{(k)} = -\mathbf{B}_k^{-1} \nabla f(\boldsymbol{\beta}^{(k)})$$

where $\mathbf{B}_k$ is a symmetric non-singular matrix. The case $\mathbf{B}_k = \mathbf{I}$ corresponds to the steepest descent while for $\mathbf{B}_k = \nabla^2 f(\boldsymbol{\beta}^{(k)})$, we have the Newton's method.

### 1.2.5 MM and EM algorithms

#### 1.2.5.1 MM algorithms

The acronym MM proposed by Hunter and Lange [35] does double duty. In minimization problems, the first M stands for majorize and the second M for minimize. In maximization problems, the first M stands for minorize and the second M for maximize. Before going further, let us remind the definition of the terms "majorize" and "minorize".

**Definition 1.2.3.** Let $\boldsymbol{\beta}^{(m)}$ a given point in $\mathbb{R}^d$. A function $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ is said to

(a) majorize a function $f(\boldsymbol{\beta})$ at $\boldsymbol{\beta}^{(m)}$ if the following conditions are satisfied:

$$f(\boldsymbol{\beta}^{(m)}) = g(\boldsymbol{\beta}^{(m)} \mid \boldsymbol{\beta}^{(m)}) \tag{1.16}$$

and

$$f(\boldsymbol{\beta}) \leqslant g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)}), \quad \boldsymbol{\beta} \neq \boldsymbol{\beta}^{(m)}. \tag{1.17}$$

(b) minorize a function $f(\boldsymbol{\beta})$ at $\boldsymbol{\beta}^{(m)}$ if $-g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ majorizes $-f(\boldsymbol{\beta})$ at $\boldsymbol{\beta}^{(m)}$.

Actually, MM is not an algorithm but rather a tool (or principle) for constructing optimization algorithms. According to the literature [18, 36, 45], the general principle behind MM algorithms was first enunciated by Ortega and Rheinbolt [82] in the context of line search methods but majorization algorithms were first used systematically in statistics in the area of multidimensional scaling by de Leeuw [16].

The MM principle for constructing an optimization algorithm consists of two steps. Let $\boldsymbol{\beta}^{(m)}$ be the iterate after $m$ iterations. In a minimization problem where one wants to minimize a function $f(\boldsymbol{\beta})$, the first M step consists in defining a majorizing function $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ and the second M step consists in minimizing the surrogate function $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ with respect to $\boldsymbol{\beta}$ rather than the function $f$ itself and the next iterate $\boldsymbol{\beta}^{(m+1)}$ is obtained as the value in which $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ attains its minimum. In maximization problems, the first M step consists in minorizing the objective function by a surrogate function $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ and the second M step consists in maximizing $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ with respect to $\boldsymbol{\beta}$ to produce the next iterate $\boldsymbol{\beta}^{(m+1)}$.

Without loss of generality, we now focus on minimization problems. Then one can note that

$$\begin{aligned} f(\boldsymbol{\beta}^{(m+1)}) &\leqslant g(\boldsymbol{\beta}^{(m+1)} \mid \boldsymbol{\beta}^{(m)}) && \text{by (1.17)} \\ &\leqslant g(\boldsymbol{\beta}^{(m)} \mid \boldsymbol{\beta}^{(m)}) && \text{by definition of } \boldsymbol{\beta}^{(m+1)} \\ &= f(\boldsymbol{\beta}^{(m)}) && \text{by (1.16)} \end{aligned}$$

and therefore the MM iterates generate a descent algorithm.

The presentation of the MM principle has assumed so far that the next iterate $\boldsymbol{\beta}^{(m+1)}$ is obtained through minimization of the surrogate function $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$. If the minimum of the surrogate function cannot be found exactly, it is sufficient to find a value $\boldsymbol{\beta}^{(m+1)}$ decreasing $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ i.e. a value $\boldsymbol{\beta}^{(m+1)}$ such that

$$g(\boldsymbol{\beta}^{(m+1)} \mid \boldsymbol{\beta}^{(m)}) \leqslant g(\boldsymbol{\beta}^{(m)} \mid \boldsymbol{\beta}^{(m)}).$$

If it is impossible to optimize the surrogate function explicitly, one can use Newton's method or some form of block relaxation.

Concerning the convergence of the MM algorithm, it is proven in [43, Proposition 15.3.2] that its converges at a linear rate. And in Proposition 15.4.3 of the same book, the author proves under some conditions that any sequence of iterates generated by the MM algorithm possesses a limit, and that limit is a stationary point of $f$.

The difficulty in constructing MM algorithms lies in choosing a surrogate function $g(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$. There are many rules that allow one to majorize complicated objective functions. One of them is to use the fact that the majorization relation between functions is preserved by the formation of sums, non-negative products, limits and composition with an increasing function. Skill in dealing with inequalities is also crucial in constructing majorizations. And finally, classical inequalities such as Jensen's inequality, the arithmetic-geometric mean inequality and the Cauchy-Schwartz's inequality prove useful in many problems.

#### 1.2.5.2 EM algorithms

Expectation-Maximization (EM) algorithm [20, 60] is a special case of the MM algorithm applicable to the iterative computation of maximum likelihood (ML) estimates. On each iteration of the EM algorithm, there are two steps called the Expectation step or the E step and the Maximization step or the M step. The EM algorithm is a popular tool in statistical estimation problems involving incomplete (or missing) data where ML estimation is made difficult by the absence of some part of data. The basic idea of the EM algorithm is to associate with the given incomplete data problem, a complete data problem for which ML estimation is computationally more tractable. The methodology of the EM algorithm can be defined as follows.

Let $\mathbf{X}$ be a random vector with probability density function $f(x \mid \boldsymbol{\beta})$ depending on a vector parameter $\boldsymbol{\beta} \in \mathbb{R}^d$. Consider that we can write $\mathbf{X} = (\mathbf{Y}, \mathbf{Z})$ where $\mathbf{Y}$ is the observed incomplete data and $\mathbf{Z}$ is the missing data.

**Algorithm 1.1** The EM algorithm

Starting from an initial point $\boldsymbol{\beta}^{(0)}$, the iterate $\boldsymbol{\beta}^{(m+1)}$ is generated in the following manner.

– **E step**: calculate the conditional expectation

$$Q(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)}) = \mathrm{E}\Big[\log f(\mathbf{X} \mid \boldsymbol{\beta}) \mid \mathbf{Y} = \mathbf{y}, \boldsymbol{\beta}^{(m)}\Big]. \tag{1.18}$$

– **M step**:

$$\boldsymbol{\beta}^{(m+1)} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\, Q(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)}). \tag{1.19}$$

**Convergence of the EM algorithm**

These convergence properties are studied by Wu [105] and McLachlan and Krishnan [60]. Wu [105] gives some few conditions under which the EM algorithm converges to local optima. But in general, if the log-likelihood has several (local or global) maxima, then the convergence of the EM algorithm depends on the starting point and Wu [105] recommends to try several EM iterations with different starting points. It is also proved that the EM algorithm enjoys the ascent property

$$\log g(\mathbf{y} \mid \boldsymbol{\beta}^{(m+1)}) \geqslant \log g(\mathbf{y} \mid \boldsymbol{\beta}^{(m)})$$

where $\log g(\mathbf{y} \mid \boldsymbol{\beta})$ is the log-likelihood of the observed data.

The EM algorithm shares some of the negative features of the more general MM algorithm. For example, the EM algorithm often converges at a slow rate and this rate often depends on the amount of missing data. In the absence of concavity, there is also no guarantee that the EM algorithm will converge to the global maximum. These drawbacks have resulted in the development of modifications and extensions of the algorithm. The first extension that we can mention is the Generalized EM (GEM) algorithm. In the GEM algorithm, the E step is unchanged and the M step of the EM algorithm is transformed into a GM step. The main difference between M and GM steps is that instead of generating the next iterate $\boldsymbol{\beta}^{(m+1)}$ by maximizing the conditional expectation $Q(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ with respect to $\boldsymbol{\beta}$, one only seeks a value $\boldsymbol{\beta}^{(m+1)}$ that increases $Q(\boldsymbol{\beta} \mid \boldsymbol{\beta}^{(m)})$ i.e. a value $\boldsymbol{\beta}^{(m+1)}$ such that

$$Q(\boldsymbol{\beta}^{(m+1)} \mid \boldsymbol{\beta}^{(m)}) \geqslant Q(\boldsymbol{\beta}^{(m)} \mid \boldsymbol{\beta}^{(m)}).$$

Among other extensions, we can mention the expectation-conditional maximization (ECM) algorithm [62] and Alternating ECM (AECM) algorithm [63]. These extensions allow to speed up convergence of the EM algorithm while keeping the monotonic convergence of the likelihood values.

### 1.2.6 Block-relaxation Algorithms

Block relaxation [17, 43, 45] (specifically called block descent in minimization problems and block ascent in maximization problems) consists in dividing the parameters into disjoint blocks and cycles through the blocks, updating only the parameters within the pertinent block at each stage of a cycle. When each block consists of a single parameter, block relaxation is called *cyclic coordinate descent* or *cyclic coordinate ascent*.

#### Description of the algorithm

Without loss of generality, we focus on block descent. Let us consider the general case where we want to minimize a real-valued function $f(\boldsymbol{\beta})$ defined on the product set $\mathbb{A} = \mathbb{A}_1 \times \mathbb{A}_2 \times \cdots \times \mathbb{A}_p$ where $\mathbb{A} \subset \mathbb{R}^d$, $\mathbb{A}_i \subset \mathbb{R}^{d_i}$, $d_i \geqslant 1$ for $i = 1, \ldots, p$ and $\sum_{i=1}^{p} d_i = d$. Let us also suppose that the vector $\boldsymbol{\beta}$ is partitioned into $p$ blocks

$$\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_p) \in \mathbb{A}_1 \times \mathbb{A}_2 \times \cdots \times \mathbb{A}_p.$$

Then the block descent algorithm for minimization of $f$ over $\mathbb{A}$ is given by Algorithm 1.2.

Note that in Algorithm 1.2, it is assumed the minima in the sub-steps exist although they need not to be unique. de Leeuw [17] also notes that if there are more than two blocks,

**Algorithm 1.2** Block descent algorithm [17]

```
Given $\boldsymbol{\beta}^{(0)} \in \mathbb{A}$, the $(k+1)-$iterate is computed through the following steps:
    Step 1 :  $\boldsymbol{\beta}_1^{(k+1)} = \underset{\boldsymbol{\beta}_1 \in \mathbb{A}_1}{\operatorname{argmin}} f(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2^{(k)}, \dots, \boldsymbol{\beta}_p^{(k)})$.
    Step 2 :  $\boldsymbol{\beta}_2^{(k+1)} = \underset{\boldsymbol{\beta}_2 \in \mathbb{A}_2}{\operatorname{argmin}} f(\boldsymbol{\beta}_1^{(k+1)}, \boldsymbol{\beta}_2, \boldsymbol{\beta}_3^{(k)}, \dots, \boldsymbol{\beta}_p^{(k)})$.
    ...       ...
    Step $p$ :  $\boldsymbol{\beta}_p^{(k+1)} = \underset{\boldsymbol{\beta}_p \in \mathbb{A}_p}{\operatorname{argmin}} f(\boldsymbol{\beta}_1^{(k+1)}, \dots, \boldsymbol{\beta}_{p-1}^{(k+1)}, \boldsymbol{\beta}_p)$.
```

one can move through them in various ways. For example, one can select the block that seems most in need for improvement or even choose blocks in random order.

de Leeuw [17] gives general results on the convergence of block-relaxation algorithms. One can note that block descent algorithms enjoy the descent property since

$$\begin{aligned} f(\boldsymbol{\beta}^{(k+1)}) &= f(\boldsymbol{\beta}_1^{(k+1)}, \dots, \boldsymbol{\beta}_p^{(k+1)}) \\ &\leqslant f(\boldsymbol{\beta}_1^{(k+1)}, \dots, \boldsymbol{\beta}_{p-1}^{(k+1)}, \boldsymbol{\beta}_p^{(k)}) \\ &\leqslant f(\boldsymbol{\beta}_1^{(k+1)}, \dots, \boldsymbol{\beta}_{p-1}^{(k)}, \boldsymbol{\beta}_p^{(k)}) \\ &\dots \\ &\leqslant f(\boldsymbol{\beta}_1^{(k)}, \dots, \boldsymbol{\beta}_p^{(k)}) \\ &= f(\boldsymbol{\beta}^{(k)}). \end{aligned}$$

Generally block descent is considered to be best suited to unconstrained problems where the domain of the objective function reduces to a Cartesian product of the sub-domains associated with the different blocks. Non-separable constraints can present insuperable barriers to coordinate descent and in some problems, it could be advantageous to consider overlapping blocks [45].

### 1.2.7 Trust region algorithms

At each iteration of a trust region (TR) algorithm, the objective function $f(\boldsymbol{\beta})$ is approximated by the model function

$$m_k(\mathbf{p}) = f(\boldsymbol{\beta}^{(k)}) + \nabla f(\boldsymbol{\beta}^{(k)})^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \mathbf{B}_k \mathbf{p} \tag{1.20}$$

within the region

$$\|\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}\| < \Delta_k$$

called *trust region of radius* $\Delta_k > 0$. The matrix $\mathbf{B}_k$ is a symmetric matrix approximating the Hessian matrix $\nabla^2 f(\boldsymbol{\beta}^{(k)})$. TR algorithms compute the next iterate using the formula

$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{p}^{(k)}$ where the step $\mathbf{p}^{(k)}$ is solution to the sub-problem

$$\mathbf{p}^{(k)} = \operatorname*{argmin}_{\mathbf{p}\in\mathbb{R}^d} m_k(\mathbf{p}) \quad \text{under the constraint} \quad \|\mathbf{p}\| \leqslant \Delta_k.$$

The size of the trust region is critical to the effectiveness of the TR algorithm and is updated at each iteration using the ratio

$$\rho_k = \frac{f(\boldsymbol{\beta}^{(k)}) - f(\boldsymbol{\beta}^{(k)} + \mathbf{p}^{(k)})}{f(\boldsymbol{\beta}^{(k)}) - m_k(\mathbf{p}^{(k)})}$$

that is the ratio between the actual reduction and the reduction predicted by the model. In a minimization problem if $\rho_k$ is close to 1, the actual reduction is in agreement with the predicted reduction and not only the new point $\boldsymbol{\beta}^{(k+1)}$ is accepted but also the radius of the trust region is unchanged or increased. But if the ratio is close to 0 then the radius is reduced and the new point is rejected i.e. $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)}$.

#### Global convergence of TR algorithms

Trust region algorithms are globally convergent, that is, they converge to a stationary point from any starting point $\boldsymbol{\beta}^{(0)}$ (see Nocedal and Wright [79] or Griva et al. [29, Theorem 11.11]).

### 1.2.8 Special algorithms for least-squares optimization

Along with maximum likelihood estimation, least-squares estimation is one of the most common problem in statistics. Least-squares estimation problems generally have the form

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} f(\boldsymbol{\beta}) \qquad \text{where} \qquad f(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\boldsymbol{\beta}) \tag{1.21}$$

and $f$, $r_1, \ldots, r_n$ are all functions defined from $\mathbb{R}^d$ to $\mathbb{R}$. The functions $r_i$, $i = 1, \ldots, n$ are generally called residuals. Here is an example of application of least squares estimation to data fitting.

**Example 1.2.3** (Application of least-squares to data fitting)**.** *Least-squares are generally used in data fitting where one wants to fit data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ observed from $n$ individuals with a model $g(\boldsymbol{\beta}, \mathbf{x}_i)$ depending on a vector parameter $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)$. The vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$ contains the values of $p$ independent random variables $X_1, \ldots, X_p$ observed for the individual $i$ while $y_i$ is the value of the response variable $Y$ for individual $i$. The residuals $r_i$, $i = 1, \ldots, n$ represent the difference between the observed value $y_i$ and the value predicted by the model $g(\boldsymbol{\beta}, \mathbf{x}_i)$; in other words*

$$r_i(\boldsymbol{\beta}) = y_i - g(\boldsymbol{\beta}, \mathbf{x}_i).$$

*The least-squares problem consists of choosing the vector parameter $\boldsymbol{\beta}$ so that the fit is as close as possible in the sense that the sum of the squares of the residuals is minimized i.e.*

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - g(\boldsymbol{\beta}, \mathbf{x}_i)\right)^2. \tag{1.22}$$

*One talks about linear least-squares if the model function $g(\boldsymbol{\beta}, \mathbf{x}_i)$ is linear with respect to $\boldsymbol{\beta}$ and non-linear least-squares otherwise.*

**Remark 1.2.6.** In linear least-squares, the residuals have the form

$$r_i(\boldsymbol{\beta}) = y_i - \sum_{j=1}^{p} \beta_j x_{ij} \tag{1.23}$$

and one shows [43] that the least-squares estimator $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{1.24}$$

where $\mathbf{X}$ is a $n \times d$ matrix whose elements are $x_{ij}$, $i = 1, \ldots, n$, $j = 1, \ldots, d$ and $\mathbf{y}$ is a $n \times 1$ vector whose components are $y_i$, $i = 1, \ldots, n$.

Since the closed-form expression of the estimator $\hat{\boldsymbol{\beta}}$ is known in linear least-squares, we focus on non-linear least-squares in the reminder of this sub-section.

Let $r$ be the function from $\mathbb{R}^d$ to $\mathbb{R}^d$ defined by $r(\boldsymbol{\beta}) = (r_1(\boldsymbol{\beta}), \ldots r_n(\boldsymbol{\beta}))^T$. If $\mathbf{J}$ denotes the Jacobian matrix of $r$, then one shows (see Nocedal and Wright [79]) that

$$\mathbf{J}(\boldsymbol{\beta}) = \left(\frac{\partial r_i}{\partial \boldsymbol{\beta}_j}\right) = \begin{pmatrix} \nabla r_1(\boldsymbol{\beta})^T \\ \vdots \\ \nabla r_n(\boldsymbol{\beta})^T \end{pmatrix}$$

and that the gradient and Hessian matrix of $f$ can then be expressed as follows:

$$\begin{aligned} \nabla f(\boldsymbol{\beta}) &= \sum_{j=1}^{n} \nabla r_j(\boldsymbol{\beta})\, r_j(\boldsymbol{\beta}) = \mathbf{J}(\boldsymbol{\beta})^T r(\boldsymbol{\beta}) \\ \nabla^2 f(\boldsymbol{\beta}) &= \mathbf{J}^T(\boldsymbol{\beta})\nabla r_j(\boldsymbol{\beta})\nabla r_j(\boldsymbol{\beta})^T + \sum_{j=1}^{n} r_j(\boldsymbol{\beta})\nabla^2 r(\boldsymbol{\beta}) \\ &= \mathbf{J}^T(\boldsymbol{\beta})J(\boldsymbol{\beta}) + \sum_{j=1}^{n} r_j(\boldsymbol{\beta})\nabla^2 r(\boldsymbol{\beta}). \end{aligned} \tag{1.25}$$

### 1.2.8.1 Gauss-Newton algorithm

It is a modification of Newton's method in the specific case of non-linear least-squares. At the iteration $k$, instead of looking for the step $\mathbf{p}_k$ such that $\nabla^2 f(\boldsymbol{\beta}^{(k)})\mathbf{p}_k = -\nabla f(\boldsymbol{\beta}^{(k)})$ as in Newton's method, one uses the approximation

$$\nabla^2 f(\boldsymbol{\beta}^{(k)}) \approx \mathbf{J}^T(\boldsymbol{\beta}^{(k)})\mathbf{J}(\boldsymbol{\beta}^{(k)}) \tag{1.26}$$

and then looks for $\mathbf{p}_k^{\mathrm{GN}}$ as the solution to the linear system

$$\mathbf{J}^T(x_k)J(x_k)\mathbf{p}_k^{\mathrm{GN}} = -\mathbf{J}(\boldsymbol{\beta}^{(k)})^T r(\boldsymbol{\beta}^{(k)}). \tag{1.27}$$

The Gauss-Newton method has several advantages over the Newton's method. We can mention the fact that it spares us the calculus of the Hessian matrices of the functions $r_j$. Moreover, in practice, many least-squares problems have small residuals at the solution so that the approximation (1.26) holds.

The Gauss-Newton method is globally convergent provided some conditions on the Jacobian $\mathbf{J}(\boldsymbol{\beta})$ and the residual functions $r_j$, $j = 1, \dots, n$ are fulfilled [79].

#### 1.2.8.2 Levenberg-Marquardt algorithm

The Levenberg-Marquardt method is based on a trust region strategy. At each iteration, its consists in minimizing the model function

$$\begin{aligned} m_k(\mathbf{p}) &= \frac{1}{2}\|\mathbf{J}(\boldsymbol{\beta}^{(k)})\mathbf{p} + r(\boldsymbol{\beta}^{(k)})\|^2 \\ &= \frac{1}{2}\|r(\boldsymbol{\beta}^{(k)})\|^2 + \mathbf{p}^T\mathbf{J}(\boldsymbol{\beta}^{(k)})r(\boldsymbol{\beta}^{(k)}) + \frac{1}{2}\mathbf{p}^T\mathbf{J}(\boldsymbol{\beta}^{(k)})^T\mathbf{J}(\boldsymbol{\beta}^{(k)})\mathbf{p} \end{aligned}$$

subject to the condition $\|\mathbf{p}\| \leqslant \Delta_k$ with $\Delta_k$ the trust region radius. One shows (see Nocedal and Wright [79]) that the resolution of this problem is possible if and only if there exists a scalar $\lambda \geqslant 0$ such that

$$(\mathbf{J}^T(\boldsymbol{\beta}^{(k)})\mathbf{J}(\boldsymbol{\beta}^{(k)}) + \lambda\mathbf{I})\mathbf{p}_k = -\mathbf{J}(\boldsymbol{\beta}^{(k)})^T r(\boldsymbol{\beta}^{(k)})$$

and

$$\lambda(\Delta_k - \|p_k\|) = 0.$$

In practice, the scalar $\lambda$ is updated at each iteration. The Levenberg-Marquardt method is also shown to be globally convergent under some conditions [79].

### 1.2.9 Derivative Free Optimization (DFO): Nelder-Mead's algorithm

In Derivative Free Optimization (DFO), no derivative of the objective function is needed. The successive iterates are computed from the values of the objective function on a finite set of points. DFO algorithms are of interest when derivative information is unavailable or impractical to obtain, for instance when the objective function is expensive to evaluate or non-differentiable which renders most methods based on derivatives of little or no use. These methods remain popular due to their simplicity, flexibility and reliability [89].

There exist many classifications of DFO algorithms depending on the method used to generate the successive iterates. In [89] for example, DFO algorithms are classified as *direct* and *model-based*. Direct algorithms determine the next iterate by computing values of the objective function $f$ directly, whereas model-based algorithms construct and utilize a surrogate model of $f$ to guide the search process. They can also be classified as *stochastic* or *deterministic*, depending upon whether they require random search steps or not. For more further details on DFO, we refer the reader to review books and papers such as Conn et al. [13] and Rios and Sahinidis [89]. In this work, we consider the Nelder-Mead's algorithm that is a direct search and deterministic algorithm.

### The Nelder-Mead's algorithm

It is an iterative heuristic algorithm (that quickly provides a feasible solution not necessarily optimal or exact solution) proposed by Nelder and Mead [67] in 1965. The description given below is taken from Lagarias et al. [42] since the original version of the algorithm proposed by Nelder and Mead [67] contains some ambiguities about strictness of inequalities that have led to differences in its interpretation.

Let us consider the minimization of a function $f(\boldsymbol{\beta})$ where $\boldsymbol{\beta} \in \mathbb{R}^d$. Each iteration of the Nelder-Mead's (NM) algorithm is based on a simplex (a geometric figure of $\mathbb{R}^d$ of non-zero volume that is composed of $d+1$ vertices) in $\mathbb{R}^d$ whose vertices can be denoted $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{d+1}$ and are sorted by increasing order of $f$, that is $f(\boldsymbol{\beta}_1) \leqslant \cdots \leqslant f(\boldsymbol{\beta}_{d+1})$. Let us denote

$$\boldsymbol{\beta}_g = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{\beta}_i$$

the centroid of the $d$ best points (all vertices except for $\boldsymbol{\beta}_{d+1}$). Since we considered a minimization problem, the point $\boldsymbol{\beta}_1$ is considered as the best point and $\boldsymbol{\beta}_{d+1}$ as the worst point. At each iteration, the worst point is replaced by a new point

$$\boldsymbol{\beta}_{\text{new}} = \boldsymbol{\beta}_g + \delta(\boldsymbol{\beta}_g - \boldsymbol{\beta}_{d+1}), \quad \delta \in \mathbb{R}.$$

that belongs to the straight line $(\boldsymbol{\beta}_g, \boldsymbol{\beta}_{d+1})$. Comparing the value of the function at this new point to the values at the other points leads to different operations corresponding to four types of $\delta$. An iteration of the NM algorithm is described as follows.

1. *reflection*: Compute the reflection point

$$\boldsymbol{\beta}_r = (1+\alpha)\boldsymbol{\beta}_g - \alpha\boldsymbol{\beta}_{d+1}$$

   where the coefficient $\alpha$ is a positive constant called *reflection coefficient.* If $f(\boldsymbol{\beta}_1) \leqslant f(\boldsymbol{\beta}_r) < f(\boldsymbol{\beta}_d)$ (i.e. the new point is a definite improvement) then we accept $\boldsymbol{\beta}_{\text{new}} = \boldsymbol{\beta}_r$ and we can go the next iteration.

2. *expansion*: If $f(\boldsymbol{\beta}_r) < f(\boldsymbol{\beta}_1)$ (i.e. the new point is the best) then we compute

$$\boldsymbol{\beta}_e = \gamma\boldsymbol{\beta}_r + (1-\gamma)\boldsymbol{\beta}_g$$

   with the purpose to try to expand the simplex by moving further away from the worst point. The coefficient $\gamma$ is the *expansion coefficient.* If $f(\boldsymbol{\beta}_e) < f(\boldsymbol{\beta}_r)$ then we set $\boldsymbol{\beta}_{\text{new}} = \boldsymbol{\beta}_e$ and go to the next iteration; otherwise we set $\boldsymbol{\beta}_{\text{new}} = \boldsymbol{\beta}_r$ and go to the next iteration.

3. *contraction*: if $f(\boldsymbol{\beta}_r) \geqslant f(\boldsymbol{\beta}_d)$ (i.e. the new point is only a marginal improvement so that it would be the worst point in the new simplex), then the simplex is contracted as follows:

(a) *outside contraction*: if $f(\boldsymbol{\beta}_d) \leqslant f(\boldsymbol{\beta}_r) < f(\boldsymbol{\beta}_{d+1})$ then compute

$$\boldsymbol{\beta}_{oc} = \boldsymbol{\beta}_g + \rho(\boldsymbol{\beta}_g - \boldsymbol{\beta}_{d+1}).$$

If $f(\boldsymbol{\beta}_{oc}) \leqslant f(\boldsymbol{\beta}_r)$ then $\boldsymbol{\beta}_{\text{new}} = \boldsymbol{\beta}_{oc}$. Otherwise, go to step 4.

(b) *inside contraction*: if $f(\boldsymbol{\beta}_r) \geqslant f(\boldsymbol{\beta}_{d+1})$,

$$\boldsymbol{\beta}_{ic} = \boldsymbol{\beta}_g - \rho(\boldsymbol{\beta}_g - \boldsymbol{\beta}_{d+1}).$$

If $f(\boldsymbol{\beta}_{ic}) < f(\boldsymbol{\beta}_{d+1})$ then $\boldsymbol{\beta}_{\text{new}} = \boldsymbol{\beta}_{ic}$. Otherwise, go to step 4.

4. *shrinkage*: we keep $\boldsymbol{\beta}_1$ and we replace each of the other points $\boldsymbol{\beta}_i$ by $\boldsymbol{\beta}_1 + \sigma(\boldsymbol{\beta}_i - \boldsymbol{\beta}_1)$.

**Remark 1.2.7.** The usual values of the different coefficients are

$$\alpha = 1, \qquad \gamma = 2, \qquad \rho = \frac{1}{2} \qquad \text{and} \qquad \sigma = \frac{1}{2}. \tag{1.28}$$

They can be varied but they must always satisfy the following conditions:

$$\alpha > 0, \quad \gamma > 1, \quad \gamma > \alpha, \quad 0 < \rho < 1 \quad \text{and} \quad 0 < \sigma < 1. \tag{1.29}$$

**Remark 1.2.8.** The motivation for shrinkage is that if any step away from the current simplex is going up, perhaps the steps are too big and so the simplex is too big. Shrinkage allows to bring the other points closer to the current best point.

The NM algorithm is very popular because of its advantages: it is simple to program and it can be applied when the objective function is non-differentiable or when the closed-form expressions of the derivatives are not available. It has the reputation for being rather robust, especially with respect to starting values [66]. It nevertheless presents some major drawbacks. First, even if the average value $(d+1)^{-1}\sum_{i=1}^{d+1} f(\boldsymbol{\beta}_i)$ decreases with each iteration, this does not guarantee that the function value at the best point will reduce with each iteration [37]. Secondly, using minimal assumptions about the function also means that the NM algorithm will be very slow compared to any Newton-like method. McKinnon [59] established analytically that convergence of the Nelder–Mead's algorithm can occur to a point where the gradient of the objective function is non-zero, even when the function is convex and twice continuously differentiable. There are several modifications of the algorithm to make it converging to stationary points [13, 37, 98]. However, the NM algorithm and its modifications can be inefficient for problems of dimension greater than 5 [21]. Monahan [66] advises that Nelder-Mead should not be the first tool to be used on an optimization problem, but one to be utilized when other methods have failed.

## 1.3 Numerical algorithms for constrained optimization

Many optimization problems impose constraints on parameters. In statistics for example, parameters can be required to be positive and if one works with some special distributions such as the multinomial distribution, the sum of the parameters must be required to equal 1. In this section, we remind the fundamentals of constrained optimization afterwards we present some numerical algorithms that are often used to deal with constrained optimization.

### 1.3.1 Fundamentals of constrained optimization

Without loss of generality, we consider minimization problems. One talks about constrained minimization when in addition to the minimization of a function $f(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta} \in \mathbb{R}^d$, the solution of the problem must satisfy the conditions

$$g_i(\boldsymbol{\beta}) = 0, \qquad i = 1, \dots, p \tag{1.30}$$

$$h_j(\boldsymbol{\beta}) \geqslant 0, \qquad j = 1, \dots, q \tag{1.31}$$

where all the functions are real-valued functions defined on a subset of $\mathbb{R}^d$. The conditions (1.30) are the equality constraints and the conditions (1.31) are the inequality constraints. A point satisfying all the constraints is called a feasible point. An inequality constraint $h_j(\boldsymbol{\beta})$ is said to be active at the feasible point $\boldsymbol{\beta}$ if $h_j(\boldsymbol{\beta}) = 0$ and it is inactive if $h_j(\boldsymbol{\beta}) > 0$. Note that an optimization problem may also have equality constraints without having inequality constraints or have inequality constraints without having equality constraints.

There exist both necessary and sufficient optimality conditions for a constrained optimization problem. Detailed discussions can be found in classical reference books such as Nocedal and Wright [79] and Lange [43, 44]. We remind the Karush-Kuhn-Tucker (KKT) conditions that are considered as the foundation for many constrained optimization algorithms [79].

**Theorem 1.3.1** (First-Order Necessary Conditions [79]). *Let the objective function $f$ and the constraint functions be continuously differentiable. Let $\boldsymbol{\beta}^*$ be a local minimum of $f$. Suppose that the gradients $\nabla g_i(\boldsymbol{\beta}^*)$ of the equality constraints and the gradients $\nabla h_j(\boldsymbol{\beta}^*)$ of the active inequality constraints at the point $\boldsymbol{\beta}^*$ are all linearly independent. Then there exist Lagrange multipliers $\lambda_1, \dots, \lambda_p$ and $\mu_1, \dots, \mu_q$ such that*

$$\nabla f(\boldsymbol{\beta}) - \sum_{i=1}^{p} \lambda_i \nabla g_i(\boldsymbol{\beta}) - \sum_{j=1}^{q} \mu_j \nabla h_j(\boldsymbol{\beta}) = \mathbf{0}. \tag{1.32}$$

*and*

$$\mu_j \geqslant 0, \qquad j = 1, \dots, q; \tag{1.33}$$

$$\mu_j h_j(\boldsymbol{\beta}^*) = 0, \qquad j = 1, \dots, q. \tag{1.34}$$

The conditions (1.32) - (1.34) are known as the Karush-Kuhn-Tucker (KKT) conditions. The restriction (1.34) requires that for all $j = 1, \dots, q$, $\mu_j = 0$ whenever $h_j(\boldsymbol{\beta}) > 0$ and is called *complementary slackness condition.* There is also a sufficient condition for a local minimum given by the following theorem.

**Theorem 1.3.2** (Sufficient condition [43]). *Suppose that the objective function $f$ and the constraint functions are continuously differentiable and let $\mathcal{L}(\boldsymbol{\beta})$ be the Lagrangian*

$$\mathcal{L}(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) - \sum_{i=1}^{p} \lambda_i g_i(\boldsymbol{\beta}) - \sum_{j=1}^{q} \mu_j h_j(\boldsymbol{\beta}).$$

*If the KKT conditions* (1.32) - (1.34) *are satisfied at a point* $\boldsymbol{\beta}^*$ *and if* $\mathbf{y}^T\nabla^2 f(\boldsymbol{\beta}^*)\mathbf{y} > 0$ *for every vector* $\mathbf{y}$ *satisfying* $\nabla g_i(\boldsymbol{\beta}^*)^T\mathbf{y} = 0$ *and* $\nabla h_j(\boldsymbol{\beta}^*)^T\mathbf{y} \geqslant 0$ *for all active inequality constraints, then* $\boldsymbol{\beta}^*$ *provides a local minimum of* $f(\boldsymbol{\beta})$*.*

### 1.3.2 Penalty methods

Once again, without loss of generality, we consider the minimization problems. The penalty method adds to the objective function $f(\boldsymbol{\beta})$ a continuous non-negative penalty $p(\boldsymbol{\beta})$ that is zero when the current point satisfies all the constraints and strictly positive otherwise. Most of the time, the penalty term is obtained by multiplying the constraints by a positive multiplicative coefficient $\rho_k$ called *penalty parameter* and one then optimizes the function $f(\boldsymbol{\beta}) + \rho_k p(\boldsymbol{\beta})$. By making the penalty parameter $\rho_k$ tending to $+\infty$, violations of the constraints are severely punished.

For the equality-constrained minimization problem

$$\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) \qquad \text{subject to} \qquad g_i(\boldsymbol{\beta}) = 0, \quad i = 1, \dots, p, \tag{1.35}$$

the quadratic penalization consists in minimizing the penalized function

$$f_{\rho_k}(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) + \frac{\rho_k}{2}\sum_{i=1}^{p}(g_i(\boldsymbol{\beta}))^2 \tag{1.36}$$

while the minimization of the penalized function

$$f_{\rho_k}(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) + \rho_k\sum_{i=1}^{p}|g_i(\boldsymbol{\beta})|$$

is called $\ell_1-$penalization.

In the general minimization of $f(\boldsymbol{\beta})$ subject to equality constraints (1.30) and inequality constraints (1.31), the $\ell_1-$penalty method consists in minimizing the function

$$f_{\rho}(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) + \rho\sum_{i=1}^{p}|g_i(\boldsymbol{\beta})| + \rho\sum_{j=1}^{q}\max(0, -h_j(\boldsymbol{\beta})). \tag{1.37}$$

Whatever the penalty used, penalty methods transform a constrained problem into a sequence of unconstrained problems, each depending on the positive penalty parameters $(\rho_k)$ tending to $+\infty$ and one seeks the approximate minimizer $\boldsymbol{\beta}^{(k)}$ of $f_{\rho_k}(\boldsymbol{\beta})$ for each $k$. Note that in case of exact penalty, there is a unique unconstrained problem. The classical unconstrained optimization algorithms described in Section 1.2 can be used to solve these unconstrained problems. However one must pay attention to the fact that the function (1.37) is not differentiable at some points because of the presence of the absolute value and the maximum function.

There exist convergence results for penalty methods. In quadratic penalization for example, if each $\boldsymbol{\beta}^{(k)}$ is the exact global minimizer of $f_{\rho_k}$ defined by (1.36) and if the sequence $\rho_k \to +\infty$ then every limit point $\boldsymbol{\beta}^*$ of the sequence $\boldsymbol{\beta}^{(k)}$ is a global solution of the constrained minimization problem [79]. Lange [44] proved in Proposition 16.3.1 that under some assumptions, a constrained local minimum of $f(\boldsymbol{\beta})$ is an unconstrained local minimum of $f_\rho$ provided

$$\rho > \max\{|\lambda_1|, \dots, |\lambda_p|, \mu_1, \dots, \mu_q\}$$

where the Lagrange multipliers $|\lambda_1|, \dots, |\lambda_p|, \mu_1, \dots, \mu_q$ are defined in (1.32).

### 1.3.3 Barrier methods

Unlike penalty methods, barrier method work from the inside of the feasible region outward. They add to the objective function $f(\boldsymbol{\beta})$ a continuous barrier function $b(\boldsymbol{\beta})$ that is finite in the interior of the feasible region (the set of the points that satisfy all the constraints) and infinite on its boundary. In other words barrier methods always start from a point in the interior of the feasible region and punishes the points that reach its boundary.

For the inequality-constrained minimization problem

$$\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) \qquad \text{subject to} \qquad h_j(\boldsymbol{\beta}) \geqslant 0,\ j = 1, \dots, q, \tag{1.38}$$

one of the most widely used barrier functions is the logarithmic barrier function

$$f_\rho(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) + \rho \sum_{j=1}^{q} \log\left(h_j(\boldsymbol{\beta})\right)$$

where $\rho$ is a real number called the barrier parameter.

Barrier methods minimize the sequence of functions $f_{\rho_k}(\boldsymbol{\beta})$ for a decreasing sequence of tuning constants $\rho_k$ tending to 0.

Lange [44] proved under some conditions on the objective function and the barrier function, that: (a) if the sequence of barrier parameters $\rho_k$ decreases to 0, then the functions $f_{\rho_k}(\boldsymbol{\beta})$ attain their minima at a sequence of points $\boldsymbol{\beta}^{(k)}$ satisfying the descent property $f(\boldsymbol{\beta}^{(k+1)}) \leqslant f(\boldsymbol{\beta}^{(k)})$; (b) if the minimum point of $f(\boldsymbol{\beta})$ in some closed set $\mathbb{V}$ is unique, then the sequence $\boldsymbol{\beta}^{(k)}$ converges to that point.

### 1.3.4 Augmented Lagrangian method

The augmented Lagrangian method is a combination of the Lagrangian function and the quadratic penalty function. It is mainly designed for equality-constrained optimization of the form

$$\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) \qquad \text{subject to} \qquad g_i(\boldsymbol{\beta}) = 0, \quad i = 1, \dots, p, \tag{1.39}$$

but can be extended to inequality-constrained problems [79]. It consists in optimizing with respect to $\boldsymbol{\beta}$, the function

$$\mathcal{L}_{\boldsymbol{\lambda}^{(k)},\rho_k}(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) - \sum_{i=1}^{p} \lambda_i^{(k)} g_i(\boldsymbol{\beta}) + \frac{\rho_k}{2} \sum_{i=1}^{p} (g_i(\boldsymbol{\beta}))^2 \tag{1.40}$$

for a sequence $\rho_k$ tending to $+\infty$. In (1.40), the vector $\boldsymbol{\lambda}^{(k)} = (\lambda_1^{(k)}, \dots, \lambda_p^{(k)})$ is the vector of Lagrange multipliers and is updated in the hope of matching the true Lagrange multiplier vector $\boldsymbol{\lambda}^*$. The stationary point of the augmented lagrangian $\boldsymbol{\beta}^{(k)}$ satisfies the equation

$$\begin{aligned}\mathbf{0} &= \nabla f(\boldsymbol{\beta}^{(k)}) - \sum_{i=1}^{p} \lambda_i^{(k)} \nabla g_i(\boldsymbol{\beta}^{(k)}) + \rho_k \sum_{i=1}^{p} g_i(\boldsymbol{\beta}^{(k)}) \nabla g_i(\boldsymbol{\beta}^{(k)}) \\ &= \nabla f(\boldsymbol{\beta}^{(k)}) - \sum_{i=1}^{p} \left(\lambda_i^{(k)} - \rho_k g_i(\boldsymbol{\beta}^{(k)})\right) \nabla g_i(\boldsymbol{\beta}^{(k)}).\end{aligned}$$

A comparison with the KKT condition (1.32) suggests the update

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} - \rho_k g_i(\boldsymbol{\beta}^{(k)}).$$

Nocedal and Wright [79] proved that a good estimate of a local minimizer $\boldsymbol{\beta}^*$ of $f(\boldsymbol{\beta})$ can be obtained by minimizing $\mathcal{L}_{\boldsymbol{\lambda}^{(k)},\rho_k}(\boldsymbol{\beta})$ even when $\rho_k$ is not particularly large, provided that $\boldsymbol{\lambda}^{(k)}$ is a good estimate of the true Lagrange multiplier vector $\boldsymbol{\lambda}^*$.

### 1.3.5 Interior point methods

Interior-point methods are considered as one of the most powerful algorithms for large-scale non-linear optimization. They associate to the minimization of a function $f(\boldsymbol{\beta})$ under the equality constraints (1.30) and the inequality constraints (1.31), the equality-constrained problem

$$\min_{\boldsymbol{\beta},\mathbf{s}} \quad f(\boldsymbol{\beta}) - \rho \sum_{j=1}^{q} \log s_j \tag{1.41}$$

subject to

$$g(\boldsymbol{\beta}) = \mathbf{0} \tag{1.42}$$

$$h(\boldsymbol{\beta}) - \mathbf{s} = \mathbf{0} \tag{1.43}$$

where $g(\boldsymbol{\beta}) = (g_1(\boldsymbol{\beta}), \dots, g_p(\boldsymbol{\beta}))^T$ and $h(\boldsymbol{\beta}) = (h_1(\boldsymbol{\beta}), \dots, h_q(\boldsymbol{\beta}))^T$ are vector-valued functions, $\rho$ is a positive barrier parameter and $\mathbf{s} = (s_1, \dots, s_q)$ is a vector of slack variables introduced in order to transform inequality constraints into equality constraints. The constraint $\mathbf{s} \geqslant \mathbf{0}$ is omitted because minimization of the barrier term $-\mu \sum_{j=1}^{q} \log s_j$ prevents the components of $\mathbf{s}$ from becoming too close to zero. One shows [79] that the KKT conditions are equivalent to the system of equations

$$\begin{aligned}\nabla f(\boldsymbol{\beta}) - \mathbf{J}g(\boldsymbol{\beta})^T \boldsymbol{\lambda} - \mathbf{J}h(\boldsymbol{\beta})^T \boldsymbol{\mu} &= \mathbf{0} \\ -\rho \mathbf{S}^{-1} \mathbf{1}_q + \boldsymbol{\mu} &= \mathbf{0} \\ g(\boldsymbol{\beta}) &= \mathbf{0} \\ h(\boldsymbol{\beta}) - \mathbf{s} &= \mathbf{0}.\end{aligned} \tag{1.44}$$

where $\mathbf{J}g(\boldsymbol{\beta})$ and $\mathbf{J}h(\boldsymbol{\beta})$ are the Jacobian matrices of the functions $g$ and $h$ respectively, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p)^T$ and $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_q)^T$ are the vectors of Lagrange multipliers associated respectively to the equality constraints (1.42) and (1.43), $\mathbf{S} = \text{diag}(s_1, \ldots, s_q)$ and $\mathbf{1}_q = (1, \ldots, 1)^T$. The system of equations (1.44) has the vector $(\boldsymbol{\beta}, \mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\mu})^T$ as vector of unknowns and can be solved by Newton's method for example.

Nocedal and Wright [79] proved under few conditions that for a sequence $\rho_k$ tending to 0, the interior point algorithm generates a sequence $\boldsymbol{\beta}^{(k)}$ which limit point is feasible and satisfies the first-order optimality conditions of the problem (1.41) under the constraints (1.42)-(1.43).

## 1.4 Statistical models for crash data: state of the art

Road accidents are a major cause of death and studies on the factors that influence the probability of crashes are very useful in that they will help decision makers to take road safety measures in order to reduce the number of crashes.

There exists in the literature a plethora of statistical models for the combination of crash data. The excellent works of Lord and Mannering [52] and, more recently, Mannering and Bhat [57] provide a comprehensive review of contemporary thinking in the crash frequency-analysis field and show how methodological approaches have evolved over the years. The authors give tables summarizing the different approaches as well as the advantages and disadvantages of each approach. In general, the proposed models strongly depend not only on the available data but also on the aim of the study. In this section, we review some of the statistical models usually encountered in the literature. We refer the reader to [31, 52, 57] for more details.

### 1.4.1 Poisson regression model

It is considered as one of the most basic model [52]. The number of crashes at the roadway entity (segment, intersection, etc...) $i$ per some time period is assumed to have the following Poisson distribution:

$$Y_i \sim \mathcal{P}(\lambda_i)$$

where $\lambda_i$ is the Poisson parameter for roadway entity $i$, which is equal to expected number of crashes denoted by $\text{E}(Y_i)$. In a Poisson regression model, $\lambda_i$ is commonly written as a function of explanatory variables:

$$\lambda_i = \exp(\boldsymbol{\gamma}^T \mathbf{T}_i) = \exp\left(\sum_{j=1}^{m} \gamma_j T_{ij}\right)$$

where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_m)$ is a vector of parameters and $\mathbf{T}_i = (T_{i1}, \ldots, T_{im})$ is a vector of explanatory variables.

One of the main drawbacks of the Poisson model is that it restricts the mean and variance to be equal and therefore, it cannot handle over-dispersion (the variance exceeds the mean of the crash counts) nor under-dispersion (the mean of the crash counts is greater than the variance). For example, in presence of over-dispersion, estimating a Poisson model can result in biased and inconsistent parameter estimates which could lead to erroneous conclusions [52].

### 1.4.2 Negative binomial regression model

The negative binomial model [53, 61, 80, 106] (also called Poisson-Gamma model) is an extension of the Poisson model commonly applied to deal with over-dispersed data. It is derived by rewriting the Poisson parameter for each observation $i$ as

$$\lambda_i = \exp(\boldsymbol{\gamma}^T \mathbf{T}_i + \varepsilon_i) = \exp\left(\sum_{j=1}^{m} \gamma_j T_{ij} + \varepsilon_i\right)$$

where $\exp(\varepsilon_i)$ (an error term) has a gamma distribution with mean 1 and variance $\alpha$, $\alpha > 0$ being the over-dispersion parameter. The negative binomial model verifies the relationship (see [52])

$$\operatorname{var}(Y_i) = \mathrm{E}(Y_i) + \alpha \mathrm{E}(Y_i)^2$$

which allows the variance to be greater than the mean since $\alpha > 0$.

Lord et al. [52, 53] claimed that the negative binomial model is the most common distribution used for modelling crash data. However, it suffers from some drawbacks such that its inability to handle under-dispersed data.

### 1.4.3 Poisson-lognormal and Poisson-Weibull models

They have been proposed as alternatives to the negative-binomial model for modelling crash data. The main difference between these models and the negative-binomial model is the distribution of $\exp(\varepsilon_i)$. In the Poisson-lognormal model [41, 53], $\exp(\varepsilon_i)$ is assumed to have a log-normal distribution while in the Poisson-Weibull [10, 41], it is assumed to have a Weibull distribution.

### 1.4.4 Bivariate and multivariate models

Bivariate/multivariate models become necessary in crash data modelling when one wishes to model different types of crashes (for example, crashes resulting in fatalities, injuries, etc.). To model the number of accidents for different types of crashes, one cannot use independent count models because the counts of the different crash types are not independent [52]. In this subsection, a particular emphasis is put on the generalizations of Tanner [97]'s model since one of them serves for the estimation algorithms developed and studied in the next chapters. We end this subsection by reviewing some other bivariate/multivariate models used in crash data modelling.

#### 1.4.4.1 Tanner's model

The model proposed by Tanner [97] is designed for the estimation of the mean effect of a road safety measure (transformation of intersections into roundabouts, installation of roundabouts, modification of the ground marking, etc...) that was simultaneously applied at $s$ ($s > 0$) experimental sites. Each experimental site is linked to a control site which is an area near the experimental site where the measure was not directly applied and from which trends due to external factors can be reliably assessed. If $B_i$ (resp. $A_i$), $i = 1, \dots, s$, is the random variable representing the number of crashes at the site $i$ in the "before" (resp. "after") period, $c_i$ is the ratio of accidents after to before in the control area for site $i$ and $n_i$ the total number of accidents at the site $i$, Tanner [97] assumes that the random variables $B_i$ and $A_i$ have the binomial distributions

$$B_i \sim \mathcal{B}\left(n_i, \frac{1}{1+\theta_i c_i}\right) \quad ; \quad A_i \sim \mathcal{B}\left(n_i, \frac{\theta_i c_i}{1+\theta_i c_i}\right) \tag{1.45}$$

where the parameters $\theta_1, \dots, \theta_s$ are positive real numbers representing the unknown effects of the measure at each site. He proposed a chi-square test for the null hypothesis

$$H_0 : \theta_1 = \dots = \theta_s = \theta.$$

The interpretation of the mean effect $\theta$ can be done by comparing it to 1. When the effects at all sites are assumed to be equal to a common effect $\theta$, Tanner [97] proposed to use a chi-square test with null hypothesis $(H_0) : \theta = 1$. If $\theta$ is found to be significantly greater than 1, then it could be concluded that on average the introduction of the measure has helped to reduce significantly the number of accidents. It can be said that the measure had no effect if $\theta = 1$ and a negative effect if $\theta > 1$. When the hypothesis of equality of the effects at all the sites is rejected, he obtains the mean effect at a given site as the sum of the common mean effect and a small variation.

It should be noted that the chi square test proposed by Tanner have been discussed by some authors [9, 47]. For example, when $s = 1$, they claimed that the asymptotic distribution of the test statistic might be much larger than that of a chi-square distribution if the ratio

$$\tau = \frac{\text{number of crashes at the experimental site}}{\text{number of crashes at the control site}}$$

is not close to zero.

#### 1.4.4.2 Generalizations of Tanner's model

They are proposed by N'Guessan et al. [69, 72, 73, 74, 77] who extended Tanner's binomial model to multinomial models by taking into consideration the different accident types.

More precisely, consider that a road safety measure has been applied to $s$ sites and that the accidents occurring at these sites can be classified into $r$ ($r > 1$) mutually exclusive accidents types. In order to assess the mean effect $\theta$ of the measure, one considers the random vectors

$$\mathbf{X}_k = (X_{11k}, \dots, X_{1rk}, X_{21k}, \dots, X_{2rk})^T, \quad k = 1, \dots, s$$

where $X_{1jk}$ (resp. $X_{2jk}$), $j = 1, \dots, r$, represents the number of crashes of type $j$ occurred in the "before" (resp. "after") period at the site $k$. In order to take into account some external factors (traffic flow, speed limit variation, weather conditions, ...), each site is linked with a control area where the safety measure was not directly applied. This control zone is described by a vector $\mathbf{Z}_k = (z_{1k}, \dots, z_{rk})^T$ such that $z_{jk}$ denotes the ratio of the number of accidents of type $j$ for the period "after" to the period "before" in the control area linked with site $k$ over the same time period. Let the vector

$$\mathbf{x}_k = (x_{11k}, \dots, x_{1rk}, x_{21k}, \dots, x_{2rk})^T$$

be the observed value of $\mathbf{X}_k$, $k = 1, \dots, s$. The total number of accidents observed at the site $k$ where the measure was applied is a fixed constant denoted by $n_k$ i.e.

$$\sum_{i=1}^{2} \sum_{j=1}^{r} x_{ijk} = n_k$$

and the control coefficients $z_{1k}, \dots, z_{rk}$, $k = 1, \dots, s$, are known and non-random. It is assumed that the measure has a multiplicative effect on expected numbers of accidents in the "after" period whatever the type of accident may be [81]. If the experimental site behaves like its control site, the number of accidents of type $j$ that one should expect to observe after the application of the road safety measure would be

$$x^*_{2jk} = \theta^* z_{jk} x_{1jk}$$

where

$$\theta^* = \frac{\sum_{k=1}^{s} \sum_{j=1}^{r} x_{2jk}}{\sum_{k=1}^{s} \sum_{j=1}^{r} z_{jk} x_{1jk}}$$

is the apparent average effect of the safety measure (i.e. the ratio of the total number of accidents recorded in the after period to the total number of accidents that would be expected in that period if the measure had no effect).

Under these assumptions, the total number of accidents that one might expect to observe at the site $k$ in both periods is

$$n^*_k = \sum_{j=1}^{r} x_{1jk} + \theta^* \sum_{j=1}^{r} z_{jk} x_{1jk}.$$

The proportions of accidents on the experimental site during the "before" and "after" periods might then be estimated by

$$\frac{x_{11k}}{n^*_k}, \quad \dots, \quad \frac{x_{1rk}}{n^*_k}, \quad \frac{\theta^* z_{1k} x_{11k}}{n^*_k}, \quad \dots, \quad \frac{\theta^* z_{rk} x_{1rk}}{n^*_k}. \tag{1.46}$$

If we set

$$\phi^*_{jk} = \frac{x_{1jk}}{\sum_{m=1}^{r} x_{1mk}} \quad \text{and} \quad \overline{z}^*_k = \sum_{m=1}^{r} z_{mk}\phi^*_{mk}$$

then the quantities defined by (1.46) can be rewritten

$$\frac{\phi^*_{1k}}{1+\theta^*\overline{z}^*_k}, \quad \ldots, \quad \frac{\phi^*_{rk}}{1+\theta^*\overline{z}^*_k}, \quad \frac{\theta^* z_{1k}\phi^*_{1k}}{1+\theta^*\overline{z}^*_k}, \quad \ldots, \quad \frac{\theta^* z_{rk}\phi^*_{rk}}{1+\theta^*\overline{z}^*_k}.$$

If we set

$$\pi^*_{1jk} = \frac{\phi^*_{jk}}{1+\theta^*\overline{z}^*_k}, \qquad \pi^*_{2jk} = \frac{\theta^* z_{jk}\phi^*_{jk}}{1+\theta^*\overline{z}^*_k}, \qquad j = 1, \ldots, r, \tag{1.47}$$

then the quantities $\pi^*_{11k}, \ldots, \pi^*_{1rk}$ and $\pi^*_{21k}, \ldots, \pi^*_{2rk}$ represent a repartition of the $n_k$ accidents into $2r$ mutually exclusive classes within the "before" and "after" periods.

Drawing their inspiration from the structure of the proportions $\pi^*_{ijk}$ above, N'Guessan et al. [73] then assume that for a fixed site $k$ the probability of the random vector $\mathbf{X}_k$ is the multinomial distribution

$$\mathbf{X}_k \sim \mathcal{M}(n_k; \boldsymbol{\pi}_{1k}, \boldsymbol{\pi}_{2k})$$

where $\boldsymbol{\pi}_{ik} = (\pi_{i1k}, \ldots, \pi_{irk})^T$, $\quad i = 1, 2$ and

$$\pi_{ijk} = \begin{cases} \dfrac{\phi_{jk}}{1+\theta\sum_{m=1}^{r} z_{mk}\phi_{mk}} & i = 1; \quad j = 1, \ldots, r, \\[2ex] \dfrac{\theta z_{jk}\phi_{jk}}{1+\theta\sum_{m=1}^{r} z_{mk}\phi_{mk}}, & i = 2; \quad j = 1, \ldots, r. \end{cases} \tag{1.48}$$

The parameter vector of the model thus constructed, denoted $\boldsymbol{\beta}$, has the form

$$\boldsymbol{\beta} = (\theta, \boldsymbol{\phi}_1^T, \ldots, \boldsymbol{\phi}_s^T)^T \in \mathbb{R}^{1+sr}$$

where $\theta > 0$ represents the interest parameter (or the mean effect of the measure) and the vectors $\boldsymbol{\phi}_k = (\phi_{1k}, \ldots, \phi_{rk})^T$, $k = 1, \ldots, s$, belong each to the simplex

$$\mathbb{S}_{r-1} = \Big\{(p_1, \ldots, p_r) \in \mathbb{R}^r \mid p_i > 0, \quad 1 \leqslant i \leqslant r, \quad \sum_{i=1}^{r} p_i = 1\Big\}. \tag{1.49}$$

**Remark 1.4.1.** A test of the hypothesis $\theta = 1$ is proposed by N'Guessan and Bellavance [72]. For a given $\alpha \in [0, 1]$, a confidence interval for $\theta$ is constructed and the latter authors proposed to reject the hypothesis $\theta = 1$ when the confidence interval do not contain the value 1.

**Remark 1.4.2.** An alternative representation of the mean effect $\theta$ is proposed by N'Guessan and Bellavance [72] who claimed that, in practice it is recommended to use the logarithm of the effect rather than the effect itself. They then proposed to modify the model (1.48) by introducing the parametrization $\theta = \exp(\alpha)$.

Another generalization proposed by N'Guessan et al. [74] consists in modifying the model (1.48) in the following way:

$$\pi_{2jk} = \frac{\theta \overline{z}_k \phi_{jk}}{1 + \theta \overline{z}_k}, \quad \overline{z}_k = \sum_{m=1}^{r} z_{mk}\phi_{mk} \quad j = 1, \ldots, r. \tag{1.50}$$

They motivated their choice to use $\overline{z}_k$ instead of $z_{jk}$ in the after period by the fact that the individual control coefficients are usually unstable and not free from error because calculated from ratio of accidents in control areas whereas mean values $\overline{z}_k$ are more stable, free from error and constant.

The last generalizations that can be mentioned have been proposed by N'Guessan and Langrand [77]. The latter authors replaced the real parameter $\theta$ by a vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_s)$ and they proposed a statistical test for the hypothesis $\theta_1 = \cdots = \theta_s = \theta$.

#### 1.4.4.3 Other bivariate/multivariate models

Among other bivariate/multivariate models, we can mention the multivariate Poisson model [55], the multivariate Poisson-lognormal model [56, 85], multinomial-generalized Poisson model [11]. We refer the reader to [57] for more details.

## 1.5 Conclusion

In this review chapter, we presented the main optimization algorithms used in statistics. In most of the optimization problems encountered in practice, it is not generally possible to find analytic solutions and one has to use an iterative method. The very first iterative optimization algorithm that comes to mind is the well-known Newton's method also called Newton-Raphson's method. It is very efficient and converges quickly to a local optimizer when the starting point of the iterative scheme is close to the true unknown value. However, it has several major drawbacks amongst which the fact that it requires calculation and inversion of the second derivatives matrix (the Hessian matrix) at each iteration. This major drawback makes the Newton's method impractical when the Hessian matrix is singular in a neighbourhood of the solution. Moreover, if the initial point is far from the true solution, the Newton's method can diverge violently. There exist many remedies to the Newton's method such as the Fisher's scoring that replaces the Hessian matrix by its expectation and the quasi-Newton methods that compute approximations of the Hessian matrix. While the use of Fisher's scoring is limited to log-likelihood maximization, quasi-Newton methods can be used in every optimization problem. The last decades have seen the development and the popularization of EM and MM algorithms that are principles for constructing optimization algorithms rather than simple optimization algorithms. They are considered as effective for maximum likelihood estimation because they consistently drive the likelihood uphill by maximizing a simple surrogate function for the log-likelihood. In problems where the derivatives are difficult to evaluate for example, derivative-free optimization algorithms must be given full consideration.

In constrained problems, the Karush-Kuhn-Tucker conditions are probably the most widely used and represent the basis for numerical algorithms for constrained optimization such as penalty, barrier, augmented lagrangian and interior points methods.

We also presented some of the most common models for modelling crash data such as Poisson regression, negative binomial and multivariate models. The multivariate model constructed by N'Guessan et al. [73] is of particular interest for us since it will be used in the remainder of this thesis.

In the next chapter, we present and study the convergence properties of the algorithm proposed by N'Guessan [71] for constrained maximum likelihood estimation in the framework of statistical analysis of accidents data. We investigate the convergence of the CA to a stationary point and its ascent property. We also compare it with the most performing algorithms such as those presented in this chapter.

# Chapter 2

# A cyclic algorithm for constrained maximum likelihood estimation

This chapter is an extended version of the article [76] and a part of the results presented was the subject of the conference article [75].

## 2.1 Introduction

The maximum likelihood (ML) method very often quoted and used in statistics, is a statistical estimation method enabling, according to the problem data, to estimate the unknown parameters linked to a probability function. And this is typically done by maximizing the *likelihood* function that is often obtained under the form of a product of probabilities. So it is equivalent to maximise its logarithm and one then talks about the constrained log-likelihood method.

One of the most popular and used probability functions is the so called multinomial law or distribution. The basic principle of this distribution consists in distributing a finite number, say $n$ ($n > 0$), of items in a finite number, say $d$ ($d > 0$), of categories or classes. The probability $\pi_j$, $0 < \pi_j < 1, j = 1, \dots, d$, for an object to fall in the class $j$ is called class probability with the sum of all class probabilities equal to 1, that is

$$\sum_{j=1}^{d} \pi_j = 1.$$

The random vector $\mathbf{X} = (X_1, \dots, X_d)$ where $X_i$ represents the number of items assigned to the class $i$, , $i = 1, \dots, d$, is said to have the multinomial distribution of parameters $n$ and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_d)$ denoted $\mathcal{M}(n; \boldsymbol{\pi})$.

Given a sample $\mathbf{x} = (x_1, \dots, x_d)$ from $\mathcal{M}(n; \boldsymbol{\pi})$ such that $\sum_{i=1}^{d} x_i = n$, one can show that the maximum likelihood estimator (MLE) of the class probability $\pi_j$ is obtained as $\hat{\pi}_j = x_i/n$. Unfortunately in practice, the class probabilities can depend on unknown auxiliary parameters which are very often under constraints and it is generally difficult to obtain closed-form expression of the MLE. One then uses iterative methods.

A review of optimization methods is given in Chapter 1 and more details can be found in classical books and papers such as Dennis and Schnabel [21], Nocedal and Wright [79], Lange [43, 44] and Lange et al. [45].

In spite of those significant contributions, the practical case studies still remain very sensitive and several complications may compromise the performance of these traditional algorithms especially in the case of multivariate discrete data: (a) the Hessian matrix or an approximation can be costly in terms of calculation, (b) it may not be positively definite, i.e. the inversion is not possible, (c) for data of important dimensions, inverting the Hessian matrix (or solving a linear system in order to invert it) can be costly, (d) if there exist constraints on the parameters, then the update itself needs adapted modifications and (e) the choice of an initial solution vector enabling a rapid convergence remains an important key for all the iterative methods.

The recent decades have seen the popularization of the Expectation Maximization (EM) algorithm [20] which is a special case of the more general class of MM (Minorization-Majorization) optimization algorithms [36, 65, 108]. Generally, MM algorithms are considered as effective algorithms for maximum likelihood estimation because they consistently drive the likelihood uphill by maximizing a simple surrogate function for the log-likelihood. However, because of their simplification of the original problem, MM algorithms can be slow to convergence.

Despite the many remedies and guarantees brought by scientific results, one is still facing the greater and greater complexity of numerical algorithms and the fact that they are not accessible to non-specialists. More generally, iterative algorithms need good starting values, otherwise they may converge slowly or even fail to converge.

In this particular context, N'Guessan and Truffier [78] and N'Guessan [71] have replaced the constrained maximum likelihood problem by the one of solving a constrained non-linear system of equations. Using the partition of the vector parameter into two subsets of parameters, these authors proved the existence of solutions by transforming the non-linear system into a linear sub-system and they proposed an estimation algorithm that cycles through the components which they called Cyclic iterative Algorithm (CA).

In this chapter, we study the convergence properties of the CA on both numerical and theoretical points of view. On the numerical point of view, we investigate some important features of the CA such as the number of iterations, the influence of the initial guess, the computation time and we compare it with the most performing algorithms such as MM algorithms, Newton-Raphsons's algorithms and other algorithms available on R and MATLAB software. On the theoretical point of view, we investigate the convergence of the CA and its ascent property (i.e. the fact that the log-likelihood increases at each iteration of the algorithm).

The chapter is organized as follows. In Section 2.2, we remind some important results concerning the Schur complement that will be needed in the sequel of this chapter. Section 2.3 provides a detailed state of the art of the cyclic algorithm. In section 2.4, we prove some convergence theorems about the CA. More precisely, we prove that it converges to the MLE from every starting point and that the CA enjoys the ascent property. In section 2.5, we give the main results of our numerical experiments with different initial vectors and different values of the sample size. The mean squared error, the number of iterations and the computation time are then used as a comparison criterion between the cyclic algorithm and other classic methods using or not the Hessian matrix. We finish the chapter with a conclusion.

## 2.2 Schur complement and block matrix inversion

**Definition 2.2.1.** Let $\mathbf{M}$ be a $m \times m$ matrix partitioned as follows:

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \tag{2.1}$$

where $\mathbf{A}$ is a $p \times p$ matrix, $\mathbf{D}$ a $q \times q$ matrix, $\mathbf{B}$ a $p \times q$ matrix and $\mathbf{C}$ a $q \times p$ matrix such that $p + q = m$. If $\mathbf{A}$ and $\mathbf{D}$ are non-singular, the matrix $\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ (resp. $\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}$) is called the *Schur complement* of $\mathbf{A}$ (resp. $\mathbf{D}$) in $\mathbf{M}$.

**Notation 1.** Throughout this document, the Schur complement of $\mathbf{A}$ (resp. $\mathbf{D}$) in $\mathbf{M}$ will be denoted $(\mathbf{M}/\mathbf{A})$ (resp. $(\mathbf{M}/\mathbf{D})$) as in [71] and [78].

The Schur complement [6, 34, 84, 107] is used in several areas of mathematics through three important formulas: the Banachiewicz inversion formula (Lemma 2.2.2), the Duncan inversion formula (Lemma 2.2.1) and the Schur determinant formula (Theorem 2.2.1). It is an important tool for the statistician. Indeed, most of multidimensional parametric estimation methods used in statistics involve matrix inversion not only for the estimation of the parameters themselves (Newton-Raphson's and Fisher scoring algorithms) but also that of standard errors (variances or standard deviations of parameters) through the inversion of the Fisher information matrix. The size of the latter increasing very quickly when the dimension of the parameter space is large, a numerical inversion would be very costly in terms of calculations, hence the interest of inverting matrix by use of the Schur complement method which allows not only to analytically verify the invertibility of the information matrix but also to have the exact analytical expression of its components in order to extract the standard deviations of the various parameters of the model concerned.

The following theorem [84, p. 195] expresses the link between the determinant of the matrix $\mathbf{M}$ and the Schur complements:

**Theorem 2.2.1.** *Let* $\mathbf{M}$ *be a partitioned matrix such as in* (2.1)*. If $A$ and $D$ are non-singular, then*

$$\det(\mathbf{M}) = \det(\mathbf{M}/\mathbf{A})\det(\mathbf{A}) = \det(\mathbf{M}/\mathbf{D})\det(\mathbf{D}). \tag{2.2}$$

**Proof**. [84, p. 195] The two relationships are obtained by noting that :

$$\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{pmatrix} = \begin{pmatrix}\mathbf{A} & \mathbf{0}\\ \mathbf{C} & \mathbf{I}\end{pmatrix}\begin{pmatrix}\mathbf{I} & \mathbf{A}^{-1}\mathbf{B}\\ \mathbf{0} & \mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\end{pmatrix}$$

and

$$\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{pmatrix} = \begin{pmatrix}\mathbf{I} & \mathbf{B}\\ \mathbf{0} & \mathbf{D}\end{pmatrix}\begin{pmatrix}\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C} & \mathbf{0}\\ \mathbf{D}^{-1}\mathbf{C} & \mathbf{I}\end{pmatrix}.$$

□

These formulas reflect the fact that the invertibility of matrix $\mathbf{M}$ is related to that of the sub-matrix $(\mathbf{M}/\mathbf{A})$($\mathbf{A}$ being already assumed invertible by definition of the Schur complement $(\mathbf{M}/\mathbf{A})$). This idea can be summarized using the following corollary.

**Corollary 2.2.1.** *If the matrices* $\mathbf{M}$ *and* $\mathbf{A}$ *(resp.* $\mathbf{D}$*) are invertible in* (2.1)*, then the Schur complement* $(\mathbf{M}/\mathbf{A})$ *(resp.* $(\mathbf{M}/\mathbf{D})$*) is also invertible.*

Once the invertibility of the matrix $\mathbf{M}$ is obtained, it is natural to look for the exact analytical expression of its inverse. The following formulas, known as the *Banachiewicz inversion formula* [107] give the expression of the inverse of the matrix $\mathbf{M}$ defined by the relationship (2.1).

**Lemma 2.2.1.** *If* $(\mathbf{M}/\mathbf{A})$ *exists and is invertible then* $\mathbf{M}$ *is also invertible and its inverse is given by:*

$$\mathbf{M}^{-1} = \begin{pmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1}\\ -(\mathbf{M}/\mathbf{A})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{M}/\mathbf{A})^{-1}\end{pmatrix}. \tag{2.3}$$

*If* $(\mathbf{M}/\mathbf{D})$ *exists and is invertible then* $\mathbf{M}$ *is also invertible and its inverse is:*

$$\mathbf{M}^{-1} = \begin{pmatrix}(\mathbf{M}/\mathbf{D})^{-1} & -(\mathbf{M}/\mathbf{D})^{-1}\mathbf{B}\mathbf{D}^{-1}\\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{M}/\mathbf{D})^{-1} & \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}(\mathbf{M}/\mathbf{D})^{-1}\mathbf{B}\mathbf{D}^{-1}\end{pmatrix}. \tag{2.4}$$

**Proof**. Let us consider the following linear system :

$$\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{c}\\ \mathbf{d}\end{pmatrix}$$

where $\mathbf{c}$ and $\mathbf{d}$ are two vectors of dimensions $p$ and $q$ respectively. This system is equivalent to :

$$\begin{cases}\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y} = \mathbf{c}\\ \mathbf{C}\mathbf{x}+\mathbf{D}\mathbf{y} = \mathbf{d}\end{cases} \tag{2.5}$$

If $\mathbf{A}$ is invertible, then $\mathbf{x} = \mathbf{A}^{-1}(\mathbf{c}-\mathbf{B}\mathbf{y})$ and

$$(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})\mathbf{y} = \mathbf{d}-\mathbf{C}\mathbf{A}^{-1}\mathbf{c}.$$

If $\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ is invertible then we have :

$$\begin{aligned}\mathbf{x} &= \mathbf{A}^{-1}\mathbf{c}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}\mathbf{c}-\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{d}\\ \mathbf{y} &= -(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}\mathbf{c}+(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{d}\end{aligned}$$

or equivalently

$$\begin{pmatrix}\mathbf{x}\\\mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1}\\ -(\mathbf{M}/\mathbf{A})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{M}/\mathbf{A})^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{c}\\\mathbf{d}\end{pmatrix}$$

hence the formula (2.4).

Now if $\mathbf{D}$ is invertible, we have $\mathbf{y} = \mathbf{D}^{-1}(\mathbf{d} - \mathbf{C}\mathbf{x})$ and

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})\mathbf{x} = \mathbf{c} - \mathbf{B}\mathbf{D}^{-1}\mathbf{d}.$$

If the matrix $\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}$ is invertible too then :

$$\begin{aligned}\mathbf{x} &= (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{c} - (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}\mathbf{d}\\ \mathbf{y} &= -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{c} + (\mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1})\mathbf{d}\end{aligned}$$

and

$$\begin{pmatrix}\mathbf{x}\\\mathbf{y}\end{pmatrix} = \begin{pmatrix}(\mathbf{M}/\mathbf{D})^{-1} & -(\mathbf{M}/\mathbf{D})^{-1}\mathbf{B}\mathbf{D}^{-1}\\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{M}/\mathbf{D})^{-1} & \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}(\mathbf{M}/\mathbf{D})^{-1}\mathbf{B}\mathbf{D}^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{c}\\\mathbf{d}\end{pmatrix}$$

hence the formula (2.3). □

By identifying the different blocks in formulas (2.4) and (2.3), one gets the *Duncan inversion formula* given by the following lemma.

**Lemma 2.2.2** ([107]). *Under non-singularity assumption on matrices* $\mathbf{A}$*,* $\mathbf{D}$*,* $\mathbf{M}/\mathbf{A}$ *and* $\mathbf{M}/\mathbf{D}$*, the inverses of Schur complements in the* $\mathbf{M}$ *matrix are given by:*

$$\begin{aligned}(\mathbf{M}/\mathbf{A})^{-1} &= (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} = \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}\\ (\mathbf{M}/\mathbf{D})^{-1} &= (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}.\end{aligned}$$

## 2.3 The cyclic algorithm: state of the art

The cyclic algorithm (CA) studied in this chapter is based on the works of N'Guessan and Truffier [78] and N'Guessan [71] in the specific context of maximum likelihood estimation of the parameters of the model described in [73] for the particular case $s = 1$ (only one experimental site). The fundamental problem solved by this model is the estimation of the average effect of a road safety measure on the number of accidents on one or several experimental sites. Taking into account this specific relationship between the proposed model and the cyclic algorithm, we first revisit the model [73]. Next, we present the estimation problem and the cyclic algorithm designed to solve it.

### 2.3.1 Problem setting and models

After a certain period of application of a road safety measure[1], it is natural to wonder if the latter had the desired effect or at least, if it had a statistically significant effect. In general, when the measure can take several aspects, the estimation of its effect is done by

[1]The word *measure* refers to any action or any decision taken in order to influence the accident risks.

experimentation i.e. by varying the values of the factor and comparing the results. But in the field of road safety, the actions are more targeted (transformation of intersections into roundabouts, installation of roundabouts, modification of the ground marking, etc...) and experimentation is often not feasible. The only possibility is to use observational studies i.e. analysis of already observed data.

In this chapter, we revisit the before-after model proposed by N'Guessan et al. [73] that is considered by the latter authors as a generalization of the one proposed by Tanner [97] that is, to the best of our knowledge, one of the oldest before-after models proposed for statistical analysis of crash data. Among the reasons that motivate the choice of this model, we can mention the fact that *before-after* studies allow cause-effect interpretations [32]. Moreover, this model is the basis of the cyclic algorithm studied in this chapter.

### Presentation of the selected model

Consider an experimental site where a road safety measure has been applied. It is assumed that the different accidents occurring there can be classified into $r$ ($r > 0$) mutually exclusive accidents types. In order to assess the mean effect $\theta$ of the measure, one considers the random vector

$$\mathbf{X} = (X_{11}, \dots, X_{1r}, X_{21}, \dots, X_{2r})^T$$

where $X_{1j}$ (resp. $X_{2j}$), $j = 1, \dots, r$, represents the number of crashes of type $j$ occurred in the "before" (resp. "after") period. In order to take into account some external factors (traffic flow, speed limit variation, weather conditions, ...), the experimental site is associated to a control site where the safety measure was not directly applied. This control site is described by a vector $\mathbf{Z} = (z_1, \dots, z_r)^T$ such that $z_j$ denotes the ratio of the number of accidents of type $j$ for the period "after" to the period "before" in the control site over the same time period. Let the vector

$$\mathbf{x} = (x_{11}, \dots, x_{1r}, x_{21}, \dots, x_{2r})^T$$

be the observed value of $\mathbf{X}$ and denote by $n$ the total number of accidents observed on the experimental site where the measure was applied i.e.

$$\sum_{i=1}^{2} \sum_{j=1}^{r} x_{ij} = n.$$

The following assumptions are made

(A1) The control coefficients $z_1, \dots, z_r$ are non-random.

(A2) The random vector $\mathbf{X}$ is assumed to have the following multinomial distribution:

$$\mathbf{X} \sim \mathcal{M}(n; \boldsymbol{\pi}_1(\boldsymbol{\beta}), \boldsymbol{\pi}_2(\boldsymbol{\beta}))$$

where $\boldsymbol{\pi}_i(\boldsymbol{\beta}) = (\pi_{i1}(\boldsymbol{\beta}), \ldots, \pi_{ir}(\boldsymbol{\beta}))^T, \quad \forall i = 1, 2$ and

$$\pi_{ij}(\boldsymbol{\beta}) = \begin{cases} \dfrac{\phi_j}{1 + \theta \sum_{m=1}^{r} z_m \phi_m} & i = 1; \quad j = 1, \ldots, r, \\ \\ \dfrac{\theta z_j \phi_j}{1 + \theta \sum_{m=1}^{r} z_m \phi_m}, & i = 2; \quad j = 1, \ldots, r. \end{cases} \tag{2.6}$$

The parameter vector denoted $\boldsymbol{\beta}$ then satisfies the additional assumption:

(A3) $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi}^T)^T \in \mathbb{R}^{1+r}$ where $\theta > 0$ and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_r)^T$ belongs to the simplex

$$\mathbb{S}_{r-1} = \Big\{ (\phi_1, \ldots, \phi_r) \in \mathbb{R}^r \mid \phi_i > 0, \quad 1 \leqslant i \leqslant r, \quad \sum_{i=1}^{r} \phi_i = 1 \Big\}. \tag{2.7}$$

**Remark 2.3.1.** The interpretation of the mean effect $\theta$ can be done by comparing it to 1 through a statistical test (see N'Guessan and Bellavance [72] or N'Guessan and Truffier [78] for more details) . For example, if $\theta$ is found to be significantly greater than 1, then it could be concluded that on average the introduction of the measure has helped to reduce significantly the number of accidents occurring on the experimental site. The parameter $\phi_j$ represents the probability that an accident occurring in a region having the same characteristics than the experimental site has a severity $j$.

### 2.3.2 Constrained maximum likelihood estimation of the parameters

For an observed data $\mathbf{x} = (\mathbf{x}_1^T, \mathbf{x}_2^T)^T$ with $\mathbf{x}_1 = (x_{11}, \ldots, x_{1r})^T$, $\mathbf{x}_2 = (x_{21}, \ldots, x_{2r})^T$ such that $\displaystyle\sum_{t=1}^{2}\sum_{j=1}^{r} x_{tj} = n$, the likelihood is given by:

$$L(\boldsymbol{\beta}) = \frac{n!}{\prod_{t=1}^{2}\prod_{j=1}^{r} x_{tj}!} \prod_{t=1}^{2}\prod_{j=1}^{r} \pi_{tj}^{x_{tj}}.$$

The log-likelihood that is defined up to an additive constant by:

$$\ell(\boldsymbol{\beta}) = \sum_{j=1}^{r} \left( x_{+j} \log(\phi_j) + x_{2j} \log(\theta) - x_{+j} \log(1 + \theta \sum_{m=1}^{r} z_m \phi_m) \right) \tag{2.8}$$

with $x_{+j} = x_{1j} + x_{2j}$, $j = 1, \ldots, r$.

**Example 2.3.1.** *Figure 2.1 gives an example of graphical representation of the log-likelihood for $r = 2$. In this case, the log-likelihood is considered as a function of two parameters $\theta$ and $\phi_1$ since $\phi_2 = 1 - \phi_1$.*

Provided that it exists, the Maximum Likelihood Estimator (MLE), $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})$, of $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})$ is solution to the constrained optimization problem:

$$\begin{cases} \hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}}) = \underset{(\theta,\boldsymbol{\phi})\in\mathbb{R}^{1+r}}{\operatorname{argmax}} \ \ell(\theta, \boldsymbol{\phi}) \\ \text{subject to} \\ \forall j = 1, \ldots, r, \quad \phi_j > 0, \quad \theta > 0 \quad \text{and} \quad \sum_{j=1}^{r} \phi_j = 1. \end{cases} \tag{2.9}$$

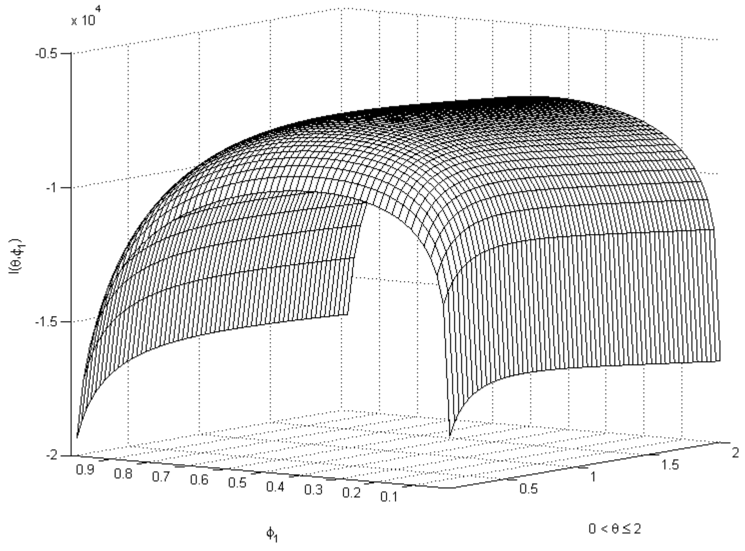

Figure 2.1. Graphical representation of the log-likelihood for $n = 5000$ and $r = 2$ when the true vector parameter is $\boldsymbol{\beta}^0 = (0.8, 0.4, 0.6)$.

This is a classic problem of Maximum Likelihood Estimation of parameters subject to constraints which has been discussed by many authors [1, 22, 51]. One can prove the following theorem.

**Theorem 2.3.1.** *Provided it exists, the MLE $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})^T$ of $\boldsymbol{\beta}$ is solution to the non-linear system of equations*

$$F(\boldsymbol{\beta}) = \mathbf{0} \tag{2.10}$$

*where $F$ is the mapping defined from $\mathbb{R}^{1+r}$ to $\mathbb{R}^{1+r}$ by $F(\boldsymbol{\beta}) = (F_0(\boldsymbol{\beta}), F_1(\boldsymbol{\beta}), \dots, F_r(\boldsymbol{\beta}))^T$ and*

$$F_0(\boldsymbol{\beta}) = \sum_{j=1}^{r} \left( x_{2j} - \frac{x_{+r}\theta \sum_{m=1}^{r} z_m\phi_m}{1 + \theta \sum_{m=1}^{r} z_m\phi_m} \right) \tag{2.11}$$

$$F_j(\boldsymbol{\beta}) = x_{+j} - \frac{n\phi_j(1 + \theta z_j)}{1 + \theta \sum_{m=1}^{r} z_m\phi_m} = 0, \qquad j = 1, \dots, r. \tag{2.12}$$

**Proof**. We only present the main steps of the proof since it is an adaptation of the general proof given by N'Guessan et al. [73] in the case $s > 1$. The introduction of one Lagrange multiplier $\lambda$ for the equality constraint enables to have the augmented log-likelihood as follows:

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda) = \ell(\boldsymbol{\beta}) - \lambda \left( 1 - \sum_{m=1}^{r} \phi_r \right).$$

The equation $F_0(\boldsymbol{\beta}) = 0$ is obtained by equalling to zero the partial derivative of $\tilde{\ell}$ with respect to (w.r.t.) $\theta$. The non-obvious part concerning the parameters $\hat{\phi}_j$, $j = 1, \dots, r$ is done by setting all the partial derivatives $\partial\tilde{\ell}/\partial\phi_j$ to 0 and summing w.r.t. $j = 1, \ldots, r$ in order to deduce the expression of $\lambda$. Indeed,

$$\frac{\partial\tilde{\ell}}{\partial\phi_j} = \frac{1}{\phi_j}\left(x_{+j} - \frac{\theta n z_j \phi_j}{1 + \theta\sum_{m=1}^{r} z_m\phi_m}\right) - \lambda. \tag{2.13}$$

Setting $\dfrac{\partial\tilde{\ell}}{\partial\phi_j} = 0$ yields:

$$\lambda\phi_j = x_{+j} - \frac{\theta n z_j \phi_j}{1 + \theta\sum_{m=1}^{r} z_m\phi_m}.$$

After summing on the index $j$, we get:

$$\lambda\sum_{j=1}^{r}\phi_j = n - \frac{\theta n\sum_{m=1}^{r} z_m\phi_m}{1 + \theta\sum_{m=1}^{r} z_m\phi_m}$$

and by taking into account the constraint $\sum_{j=1}^{r}\phi_j = 1$,

$$\lambda = \frac{n}{1 + \theta\sum_{m=1}^{r} z_m\phi_m} \tag{2.14}$$

The equations $F_j(\boldsymbol{\beta}) = 0$, $j = 1, \ldots, r$ are obtained by substituting $\lambda$ for (2.14) in (2.13) and by setting again $\frac{\partial\tilde{\ell}}{\partial\phi_j}$ to zero. $\square$

**Remark 2.3.2.** One question that arises immediately is how the constraints

$$\theta > 0 \quad \text{and} \quad 0 < \phi_j < 1, \quad j = 1, \ldots, r$$

are dealt with in the Lagrangian $\tilde{\ell}(\beta)$. First, one can note that the implication

$$\left.\begin{array}{c}\phi_j > 0, \quad j = 1, \ldots, r \\ \text{and} \\ \sum_{j=1}^{r}\phi_j = 1\end{array}\right\} \quad \Longrightarrow \quad 0 < \phi_j < 1, \quad j = 1, \ldots, r$$

holds. Thus the inequality constraints can be reduced to the following positivity constraints

$$\theta > 0 \quad \text{and} \quad \phi_j > 0, \quad j = 1, \ldots, r.$$

One of the most commonly used methods to deal with inequality constraints is the barrier method [6, 43, 44]. It consists in constructing a continuous barrier function $b(\beta)$ that is finite on the interior of the feasible region and infinite on its boundary. One then optimizes the sequence of functions $\tilde{\ell}(\beta) + \mu_k b(\beta)$ where the sequence of tuning constants $\mu_k$ tends to 0. In our case, the barrier function can take the specific form

$$b(\beta) = \log\theta + \sum_{j=1}^{r}\log\phi_j$$

called a *logarithmic barrier*. So we can say that the specific form of the log-likelihood integrates the logarithmic barrier. It diverges to negative infinity if any of the parameters $\theta$ or $\phi_j$ tends to zero.

Different iterative methods can be used to solve the non-linear equation (2.10) [21, 79]. Generally, the first that comes to mind is the Newton's method which consists in an iterative scheme of the form:

$$\mathbf{J}(\boldsymbol{\beta}^{(k)})\mathbf{p}^{(k)} = -F(\boldsymbol{\beta}^{(k)}) \tag{2.15}$$

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{p}^{(k)} \tag{2.16}$$

where $\boldsymbol{\beta}^{(0)}$ is a fixed starting guess, $\mathbf{p}^{(k)}$ is the step towards a "better" iterate and $J(\boldsymbol{\beta}^{(k)})$ is the Jacobian of $F$ applied to $\boldsymbol{\beta}^{(k)}$. Newton's method is known to be quadratically convergent from starting guesses $\boldsymbol{\beta}^{(0)}$ near the true unknown solution. However, it presents two main disadvantages. First, it requires the calculation of $\mathbf{J}(\boldsymbol{\beta}^{(k)})$ at each iteration and secondly, each iteration requires the solution of a system of linear equations that may be singular or ill-conditioned. We can notice that calculation of $\mathbf{J}(\boldsymbol{\beta}^{(k)})$ can introduce some fairly complicated intermediate calculations if the expression of $F$ is complex. These disadvantages of Newton's method presented above lead to the development of new algorithms which are generally modifications of Newton's algorithm. In applied problems such as ours, another major fact that needs to be taken into account is that the proposed algorithms must be simple to program for non-specialists. And this is exactly the main motivation of the cyclic algorithm studied in this chapter.

### 2.3.3 The cyclic algorithm (CA)

The main idea for the construction the cyclic algorithm consists in using the equation $F_0(\hat{\boldsymbol{\beta}}) = 0$ to obtain the expression of $\hat{\theta}$ from that of $\hat{\boldsymbol{\phi}}$ and using the remaining equations to get the expression of the vector $\hat{\boldsymbol{\phi}}$ w.r.t. $\hat{\theta}$. N'Guessan and Truffier [78] and N'Guessan [71] showed that the sub-system

$$F_j(\hat{\boldsymbol{\beta}}) = 0, \quad j = 1, \dots, r$$

is equivalent to a linear system of equations w.r.t. the parameters $\hat{\phi}_1$, …, $\hat{\phi}_r$ whose matrix is a function of $\hat{\theta}$ and the accident data.

The first step is completed by the following proposition.

**Lemma 2.3.1.** *The non-linear equation $F_0(\hat{\boldsymbol{\beta}}) = 0$ is equivalent to*

$$\hat{\theta} = \frac{\sum_{j=1}^{r} x_{2j}}{\left(\sum_{j=1}^{r} x_{1j}\right)\left(\sum_{j=1}^{r} z_j \hat{\phi}_j\right)}. \tag{2.17}$$

**Proof**. We have:

$$\begin{aligned}
F_0(\hat{\boldsymbol{\beta}}) = 0 &\iff \sum_{j=1}^{r} x_{2j} - \frac{n\hat{\theta}\sum_{j=1}^{r} z_j\hat{\phi}_j}{1+\hat{\theta}\sum_{j=1}^{r} z_j\hat{\phi}_j} = 0 \\
&\iff \sum_{j=1}^{r} x_{2j} - \Big(n - \frac{n}{1+\hat{\theta}\sum_{j=1}^{r} z_j\hat{\phi}_j}\Big) = 0 \\
&\iff \frac{n}{1+\hat{\theta}\sum_{j=1}^{r} z_j\hat{\phi}_j} = \sum_{j=1}^{r} x_{1j} \quad \text{since } \sum_{j=1}^{r}(x_{1j}+x_{2j}) = n \\
&\iff \hat{\theta}\sum_{j=1}^{r} z_j\hat{\phi}_j = \frac{\sum_{j=1}^{r} x_{2j}}{\sum_{j=1}^{r} x_{1j}}
\end{aligned}$$

hence the expression of $\hat{\theta}$. □

The second step is completed through the following lemmas.

**Lemma 2.3.2.** *Given $\hat{\theta}$, the sub-system $F_j(\hat{\boldsymbol{\beta}}) = 0,\ j = 1,\dots,r$ is equivalent to the linear system*

$$\mathbf{D}_{\hat{\theta},x}\hat{\boldsymbol{\phi}} = \mathbf{B}_x \tag{2.18}$$

*where*

$$\mathbf{D}_{\hat{\theta},x} = \begin{pmatrix}
1+(1-\frac{x_{+1}}{n})\hat{\theta}z_1 & -\frac{x_{+1}}{n}\hat{\theta}z_2 & \dots & -\frac{x_{+1}}{n}\hat{\theta}z_r \\
-\frac{x_{+2}}{n}\hat{\theta}z_1 & 1+(1-\frac{x_{+2}}{n})\hat{\theta}z_2 & \dots & -\frac{x_{+2}}{n}\hat{\theta}z_r \\
\vdots & \ddots & \ddots & \vdots \\
-\frac{x_{+r}}{n}\hat{\theta}z_1 & -\frac{x_{+r}}{n}\hat{\theta}z_2 & \dots & 1+(1-\frac{x_{+r}}{n})\hat{\theta}z_r
\end{pmatrix}$$

*and*

$$\mathbf{B}_x = \frac{1}{n}\left(x_{+1},\dots,x_{+r}\right)^T.$$

**Proof**. Given $\hat{\theta}$, the sub-system $F_j(\hat{\boldsymbol{\beta}}) = 0,\ j = 1,\dots,r$, is equivalent to the system of $r$ equations:

$$x_{+j}\left(1+\hat{\theta}\sum_{m=1}^{r} z_m\hat{\phi}_m\right) - n\hat{\phi}_j(1+\hat{\theta}z_j) = 0, \quad j = 1,\dots,r.$$

This system is equivalent to:

$$x_{+r} + \sum_{m\neq j}\hat{\theta}x_{+j}z_m\hat{\phi}_m + \Big((x_{+r}-n)\hat{\theta}z_j - n\Big)\hat{\phi}_j = 0, \quad j = 1,\dots,r.$$

Multiplying by $-1$ and dividing by $n$, yield:

$$-\sum_{m\neq j}\frac{x_{+j}}{n}\hat{\theta}z_m\hat{\phi}_m + \left(1+(1-\frac{x_{+j}}{n})\hat{\theta}z_j\right)\hat{\phi}_j = \frac{x_{+j}}{n}, \quad j = 1,\dots,r$$

which completes the proof. □

**Lemma 2.3.3.** *Given $\hat{\theta}$, the matrix $\mathbf{D}_{\hat{\theta},x}$ is non-singular.*

**Proof**. Set:

$$\boldsymbol{\Delta}_{\hat{\theta}} = \begin{pmatrix} 1+\hat{\theta}z_1 & 0 & \cdots & 0 \\ 0 & 1+\hat{\theta}z_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1+\hat{\theta}z_r \end{pmatrix} \quad \text{and} \quad \mathbf{M}_{\hat{\theta},x} = \begin{pmatrix} \boldsymbol{\Delta}_{\hat{\theta}} & \hat{\theta}\mathbf{B}_x \\ \mathbf{Z}^T & 1 \end{pmatrix}.$$

The matrix $\mathbf{D}_{\hat{\theta},x}$ can be written as

$$\mathbf{D}_{\hat{\theta},x} = \boldsymbol{\Delta}_{\hat{\theta}} - \hat{\theta}\mathbf{B}_x\mathbf{Z}^T$$

and then, be identified as $\mathbf{M}_{\hat{\theta},x}/1$ the Schur complement of 1 in $\mathbf{M}_{\hat{\theta},x}$ (see Definition 2.2.1). By Theorem 2.2.1,

$$\det(\mathbf{M}_{\hat{\theta},x}) = \det(\mathbf{M}_{\hat{\theta},x}/1) \times 1 = \det(\mathbf{M}_{\hat{\theta},x}/\Delta_{\hat{\theta}}) \times \det(\boldsymbol{\Delta}_{\hat{\theta}})$$

and it is easy to check that:

$$(\mathbf{M}_{\hat{\theta},x}/\boldsymbol{\Delta}_{\hat{\theta}}) = 1 - \frac{\hat{\theta}}{n}\begin{pmatrix} z_1 & \cdots & z_r \end{pmatrix} \begin{pmatrix} \frac{1}{1+\hat{\theta}z_1} & & \\ & \ddots & \\ & & \frac{1}{1+\hat{\theta}z_r} \end{pmatrix} \begin{pmatrix} x_{+1} \\ \vdots \\ x_{+r} \end{pmatrix}.$$

That is

$$\begin{aligned} (\mathbf{M}_{\hat{\theta},x}/\boldsymbol{\Delta}_{\hat{\theta}}) &= 1 - \frac{1}{n}\sum_{m=1}^{r} \frac{\hat{\theta}x_{+m}z_m}{1+\hat{\theta}z_m} = 1 - \frac{1}{n}\sum_{m=1}^{r}\left(x_{+m} - \frac{x_{+m}}{1+\hat{\theta}z_m}\right) \\ &= \frac{1}{n}\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat{\theta}z_m} > 0 \end{aligned} \tag{2.19}$$

since $\sum_{m=1}^{r} x_{+m} = n$. It follows from (2.19) that $(\mathbf{M}_{\hat{\theta},x}/\Delta_{\hat{\theta}})$ is a strictly positive number. We also have $\det(\boldsymbol{\Delta}_{\hat{\theta}}) > 0$ so that $\det(\mathbf{M}_{\hat{\theta},x}) = \det(\mathbf{M}_{\hat{\theta},x}/1) \times 1 > 0$ which means that the matrix $(\mathbf{M}_{\hat{\theta},x}/1) = \mathbf{D}_{\hat{\theta},x}$ is non-singular. □

**Lemma 2.3.4.** *Given $\hat{\theta}$, the sub-system $F_j(\hat{\boldsymbol{\beta}}) = 0$, $j = 1, \ldots, r$, accepts a solution $\hat{\boldsymbol{\phi}}$ whose components are:*

$$\hat{\phi}_j = \left(\frac{x_{+j}}{1+\hat{\theta}z_j}\right) \Big/ \left(\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat{\theta}z_m}\right), \quad j = 1, \ldots, r. \tag{2.20}$$

**Proof**. The matrix $\mathbf{D}_{\hat{\theta},x}$ being non-singular, its inverse is given by Lemma 2.2.2:

$$\mathbf{D}_{\hat{\theta},x}^{-1} = (\mathbf{M}_{\hat{\theta},x}/1)^{-1} = \boldsymbol{\Delta}_{\hat{\theta}}^{-1} + \hat{\theta}\boldsymbol{\Delta}_{\hat{\theta}}^{-1}\mathbf{B}_x(\mathbf{M}_{\hat{\theta},x}/\boldsymbol{\Delta}_{\hat{\theta}})^{-1}\mathbf{Z}^T\boldsymbol{\Delta}_{\hat{\theta}}^{-1}$$

where by (2.19),

$$(\mathbf{M}_{\hat\theta,x}/\mathbf{\Delta}_{\hat\theta}) = 1 - \hat\theta \mathbf{Z}^T \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x \quad \in \quad \mathbb{R}_+^*.$$

By (2.18), we have:

$$\begin{aligned}
\hat{\boldsymbol\phi} &= \mathbf{D}_{\hat\theta,x}^{-1}\mathbf{B}_x = \left(\mathbf{\Delta}_{\hat\theta}^{-1} + \hat\theta\mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x(\mathbf{M}_{\hat\theta}/\mathbf{\Delta}_{\hat\theta})^{-1}\mathbf{Z}^T\mathbf{\Delta}_{\hat\theta}^{-1}\right)\mathbf{B_x} \\
&= \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x + \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x(\mathbf{M}_{\hat\theta,x}/\mathbf{\Delta}_{\theta})^{-1}(\hat\theta\mathbf{Z}^T\mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x) \\
&= \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x + \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x(\mathbf{M}_{\hat\theta,x}/\mathbf{\Delta}_{\hat\theta})^{-1}\left(1 - (\mathbf{M}_{\hat\theta,x}/\mathbf{\Delta}_{\hat\theta})\right) \\
&= \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x + \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x(\mathbf{M}_{\hat\theta,x}/\mathbf{\Delta}_{\hat\theta})^{-1} - \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x \\
&= \mathbf{\Delta}_{\hat\theta}^{-1}\mathbf{B}_x(\mathbf{M}_{\hat\theta,x}/\mathbf{\Delta}_{\hat\theta})^{-1}. \qquad (2.21)
\end{aligned}$$

It follows by (2.19) and (2.21) that

$$\hat{\boldsymbol\phi} = \frac{1}{n}\left(\frac{1}{n}\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat\theta z_m}\right)^{-1} \times \begin{pmatrix} \frac{1}{1+\hat\theta z_1} & & \\ & \ddots & \\ & & \frac{1}{1+\hat\theta z_r}\end{pmatrix}\begin{pmatrix} x_{+1} \\ \vdots \\ x_{+r}\end{pmatrix},$$

that is, by component,

$$\hat\phi_j = \left(\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat\theta z_m}\right)^{-1}\left(\frac{x_{+j}}{1+\hat\theta z_j}\right), \quad j = 1, \dots, r$$

which completes the proof. □

The following theorem thus follows.

**Theorem 2.3.2** (N'Guessan [71]). *The non-linear system* (2.10) *accepts a solution* $\hat{\boldsymbol\beta} = (\hat\theta, \hat\phi_1, \dots, \hat\phi_r)^T$ *whose components satisfy:*

$$\begin{cases} \hat\theta = \dfrac{\sum_{m=1}^{r} x_{2m}}{\left(\sum_{m=1}^{r} z_m\,\hat\phi_m\right) \times \left(\sum_{m=1}^{r} x_{1m}\right)} \\[2em] \hat\phi_j = \left(\dfrac{x_{+j}}{1+\hat\theta z_j}\right)\Big/\left(\displaystyle\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat\theta z_m}\right), \quad j = 1, \dots, r. \end{cases} \qquad (2.22)$$

**Remark 2.3.3.** One can easily check that for any $\hat\theta > 0$, the vector $\hat{\boldsymbol\phi}$ as defined by Theorem 2.3.2 satisfies the constraint $\sum_{j=1}^{r}\hat\phi_j = 1$.

It follows from Theorem 2.3.2 that the MLE $\hat{\boldsymbol\beta} = (\hat\theta, \hat{\boldsymbol\phi})$ is a fixed point of the mapping $G$ defined from $\mathbb{R}^{1+r}$ to $\mathbb{R}^{1+r}$ by

$$G(\boldsymbol\beta) = G(\theta, \phi_1, \dots, \phi_r) = (G_0(\boldsymbol\beta), G_1(\boldsymbol\beta), \dots, G_r(\boldsymbol\beta))^T$$

where

$$G_j(\boldsymbol{\beta}) = \begin{cases} \dfrac{\sum_{m=1}^r x_{2m}}{(\sum_{m=1}^r z_m\,\phi_m) \times (\sum_{m=1}^r x_{1m})} & \text{if} \quad j=0 \\ \\ \left(\frac{x_{+j}}{1+\theta z_j}\right) \Big/ \left(\sum_{m=1}^r \frac{x_{+m}}{1+\theta z_m}\right) & \text{if} \quad j=1,2,\ldots,r. \end{cases} \tag{2.23}$$

Since $G(\hat{\boldsymbol{\beta}}) = \hat{\boldsymbol{\beta}}$, the MLE $\hat{\boldsymbol{\beta}}$ could be obtained from an iterative scheme of the form

$$\boldsymbol{\beta}^{(k+1)} = G(\boldsymbol{\beta}^{(k)}), \quad k = 0, 1, \ldots$$

where the starting guess $\boldsymbol{\beta}^{(0)}$ is fixed. However, because of the link between the components $\hat{\theta}$ and $\hat{\phi}_j$ $(j = 1, \ldots, r)$, it would be hard to update $\hat{\theta}$ and $\hat{\boldsymbol{\phi}}$ simultaneously. The task is significantly reduced if we use an iterative procedure alternating between updating $\hat{\theta}$ holding $\hat{\boldsymbol{\phi}}$ fixed and vice-versa. In this case the task of finding an initial solution $\boldsymbol{\beta}^{(0)} = (\theta^{(0)}, \boldsymbol{\phi}^{(0)})$ is significantly reduced to the one of choosing only a part of $\boldsymbol{\beta}^{(0)}$. For example, if the starting value $\theta^{(0)}$ is given, the iterative procedure will compute automatically the missing sub-parameter $\boldsymbol{\phi}^{(0)} = (\phi_1^{(0)}, \ldots, \phi_r^{(0)})^T$. At the step $k+1$, $\theta^{(k+1)}$ is updated from $\boldsymbol{\phi}^{(k)}$, afterwards $\boldsymbol{\phi}^{(k+1)}$ is updated from the $\theta^{(k+1)}$, and so on, until a convergence criteria is satisfied. This strategy yields the following iterative algorithm.

**Algorithm 2.1** The cyclic algorithm [71]

$$\begin{aligned} \theta^{(k+1)} &= \frac{\sum_{m=1}^r x_{2m}}{\left(\sum_{m=1}^r z_m \phi_m^{(k)}\right) \times (\sum_{m=1}^r x_{1m})} \\ \phi_j^{(k+1)} &= \left(\frac{x_{+j}}{1+\theta^{(k+1)} z_j}\right) \Big/ \left(\sum_{m=1}^r \frac{x_{+m}}{1+\theta^{(k+1)} z_m}\right), \quad j = 1, \ldots, r. \end{aligned} \tag{2.24}$$

## 2.4 Theoretical study of some numerical properties of the Cyclic Algorithm

In this section, we aim to prove that the cyclic algorithm (2.24) verifies two main properties generally required for MLE estimation iterative algorithms. The first one is the convergence of the iterative scheme (2.24) to the MLE $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})$ from the algorithmic point of view. Indeed, there is no guarantee that an iterative algorithm always converges. Proving convergence of an iterative optimization algorithm is a delicate exercise and generally this is done by imposing some conditions [44]. Remark that the notion of convergence studied here is different from the notion of consistency that will be studied later in this thesis. The first main result proved in this section is the following theorem.

**Theorem 2.4.1.** *For all starting guess* $\boldsymbol{\beta}^{(0)} = (\theta^{(0)}, \boldsymbol{\phi}^{(0)})$*, the cyclic algorithm* (2.24) *converges to the MLE* $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})$ *of* $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})$*.*

The second property that we prove is the ascent property of the cyclic algorithm 2.24 i.e. the fact that the log-likelihood is increased monotonically by the algorithm. That is given by the following theorem.

**Theorem 2.4.2.** *The cyclic algorithm 2.24 enjoys the ascent property, that is*

$$\ell(\boldsymbol{\beta}^{(k+1)}) \geqslant \ell(\boldsymbol{\beta}^{(k)}), \quad k = 0, 1, \dots \tag{2.25}$$

For simplicity, we introduce the notation $x_{i+} = \sum_{j=1}^{r} x_{ij}$ for all $i = 1, 2$. The proof of Theorem 2.4.1 is based on the following lemmas.

**Lemma 2.4.1.** *Let $\psi$ be the mapping defined on $\mathbb{R}_+$ by*

$$\psi(u) = \sum_{m=1}^{r} \frac{x_{+m}}{(1 + uz_m)} - x_{1+}. \tag{2.26}$$

**i)** *There exists a unique real number $u > 0$, denoted $\theta^*$, that is solution to the equation $\psi(u) = 0$.*

**ii)** *Let $u \in ]0, +\infty[$. Then, $\psi(u) \geqslant 0$ if $0 < u \leqslant \theta^*$ and $\psi(u) \leqslant 0$ if $u \geqslant \theta^*$.*

**iii)** *The MLE $\hat{\theta}$ of $\theta$ is equal to the unique root $\theta^*$ of $\psi$.*

**Proof**. **i)** One can easily check that $\psi$ is continuous and its derivative $\psi'(u)$ is strictly negative for every $u > 0$ and therefore $\psi$ is bijective. Moreover,

$$\lim_{u \to 0} \psi(u) \times \lim_{u \to +\infty} \psi(u) = (x_{2+}) \times (-x_{1+}) \ < \ 0$$

hence the equation $\psi(u) = 0$ has a unique solution.

**ii)** The function $\psi$ is a strictly decreasing function. Therefore,

$$\forall u \leqslant \theta^*, \quad \psi(u) \geqslant \psi(\theta^*) = 0 \quad \text{and} \quad \forall u \geqslant \theta^*, \quad \psi(u) \leqslant \psi(\theta^*) = 0.$$

**iii)** From the equality

$$\hat{\phi}_j = \frac{x_{+j}/(1 + \hat{\theta} z_j)}{\sum_{m=1}^{r} x_{+m}/(1 + \hat{\theta} z_m)}$$

we deduce the equalities

$$\sum_{j=1}^{r} z_j \hat{\phi}_j = \sum_{j=1}^{r} \frac{z_j x_{+j}/(1 + \hat{\theta} z_j)}{\sum_{m=1}^{r} x_{+m}/(1 + \hat{\theta} z_m)} = \frac{\sum_{j=1}^{r} \Big( z_j x_{+j}/(1 + \hat{\theta} z_j) \Big)}{\sum_{m=1}^{r} \Big( x_{+m}/(1 + \hat{\theta} z_m) \Big)}$$

and

$$\hat{\theta} = \frac{x_{2+}}{x_{1+}} \frac{1}{\sum_{m=1}^{r} z_m \hat{\phi}_m} = \frac{x_{2+}}{x_{1+}} \frac{\sum_{m=1}^{r} \Big( x_{+m}/(1 + \hat{\theta} z_m) \Big)}{\sum_{j=1}^{r} \Big( z_j x_{+j}/(1 + \hat{\theta} z_j) \Big)}.$$

This is equivalent to

$$\frac{x_{2+}}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat{\theta}z_m}=\sum_{m=1}^{r}\frac{\hat{\theta}z_m x_{+m}}{1+\hat{\theta}z_m}.$$

We then deduce the following equalities:

$$\begin{aligned}
\frac{x_{2+}}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat{\theta}z_m} &= \sum_{m=1}^{r}\left(x_{+m}-\frac{x_{+m}}{1+\hat{\theta}z_m}\right)\\
\frac{x_{2+}}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat{\theta}z_m} &= n-\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat{\theta}z_m}\\
\frac{n}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+\hat{\theta}z_m} &= n \quad \text{since} \quad x_{1+}+x_{2+}=n
\end{aligned}$$

hence the equality $\psi(\hat{\theta})=0$. Since $\psi$ is bijective and $\psi(\theta^*)=0$ then $\hat{\theta}=\theta^*$. This completes the proof. □

**Lemma 2.4.2.** *Let $a_{2,1}=\left(\sum_{m=1}^{r}x_{2m}\right)/\left(\sum_{m=1}^{r}x_{1m}\right)$ and $\varphi_x$ be the function from $]0;+\infty[$ to $]0;+\infty[$ defined by:*

$$\varphi_x(\theta)=a_{2,1}\left(\sum_{m=1}^{r}\frac{x_{+m}}{1+\theta z_m}\right)\Big/\left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+\theta z_m}\right). \tag{2.27}$$

**i)** *There exists a unique point $\theta^*$ such that $\varphi_x(\theta^*)=\theta^*$.*

**ii)** *The function $\varphi_x$ is such that*

$$\forall u\leqslant\theta^*,\quad \varphi_x(u)\geqslant u \quad \textit{and} \quad \forall u\geqslant\theta^*,\quad \varphi(u)\leqslant u.$$

**iii)** *The sequence $(\theta^{(k)})$ is monotonous and its monotony depends on $\theta^{(0)}$. It is an increasing sequence if $\theta^{(0)}<\theta^*$ and decreasing if $\theta^{(0)}>\theta^*$.*

**iv)** *The sequence $(\theta^{(k)})$ is also bounded. Then it is convergent and its limit is $\theta^*$.*

**Proof.** **i)** The equation $\varphi_x(u)=u$ is equivalent to

$$\frac{x_{2+}}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+uz_m}=\sum_{m=1}^{r}\frac{uz_m x_{+m}}{1+uz_m}.$$

After straightforward calculations similar to those used in the proof of Lemma 2.4.1, one gets $\psi(u)=0$. This last equation has a unique solution that is also the unique solution of the equation $\varphi_x(u)=u$.

**ii)** Let $u \in \mathbb{R}_+^*$. Then

$$\begin{aligned}
\varphi_x(u) - u &= \left(\frac{x_{2+}}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+uz_m} - \sum_{m=1}^{r}\frac{uz_m x_{+m}}{1+uz_m}\right) \Big/ \left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+uz_m}\right)\\
&= \left(\frac{x_{2+}}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+uz_m} - \sum_{m=1}^{r}\left(x_{+m} - \frac{x_{+m}}{1+uz_m}\right)\right) \Big/ \left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+uz_m}\right)\\
&= \left(\frac{x_{2+}}{x_{1+}}\sum_{m=1}^{r}\frac{x_{+m}}{1+uz_m} - n + \sum_{m=1}^{r}\frac{x_{+m}}{1+uz_m}\right) \Big/ \left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+uz_m}\right)\\
&= \frac{n}{x_{1+}}\left(\sum_{m=1}^{r}\frac{x_{+m}}{1+uz_m} - x_{1+}\right) \Big/ \left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+uz_m}\right)\\
&= \frac{n}{x_{1+}}\left(\psi(u)\right) \Big/ \left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+uz_m}\right).
\end{aligned}$$

The sign of $\varphi_x(u) - u$ is merely obtained from that of $\psi(u)$.

**iii)** Using (2.24), one can say that the real sequence $\theta^{(k)}$ is given by :

$$\theta^{(k+1)} = a_{2,1}\left(\sum_{m=1}^{r}\frac{x_{+m}}{1+\theta^{(k)}z_m}\right) \Big/ \left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+\theta^{(k)}z_m}\right). \tag{2.28}$$

which is equivalent to

$$\theta^{(k+1)} = \varphi_x(\theta^{(k)}).$$

The function $\varphi_x$ is differentiable and its derivative has the form

$$\varphi_x'(\theta) = a_{2,1} \times \varphi_{1,x}(\theta)/\varphi_{2,x}(\theta)$$

with

$$\varphi_{2,x}(\theta) = \left(\sum_{m=1}^{r}\frac{z_m x_{+m}}{1+\theta z_m}\right)^2 > 0$$

and

$$\begin{aligned}
\varphi_{1,x}(\theta) &= -\left(\sum_{i=1}^{r}\frac{z_i x_{+i}}{(1+\theta z_i)^2}\right)\left(\sum_{j=1}^{r}\frac{z_j x_{+j}}{1+\theta z_j}\right) + \left(\sum_{i=1}^{r}\frac{(z_i)^2 x_{+i}}{(1+\theta z_i)^2}\right)\left(\sum_{j=1}^{r}\frac{x_{+j}}{1+\theta z_j}\right)\\
&= \sum_{i=1}^{r}\sum_{j=1}^{r}\frac{(z_i)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_i)^2(1+\theta z_j)}.
\end{aligned}$$

After removing the zero terms corresponding to $i = j$, we get:

$$\begin{aligned}
\varphi_{1,x}(\theta) &= \sum_{i\neq j}\frac{(z_i)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_i)^2(1+\theta z_j)}\\
&= \sum_{i<j}\frac{(z_i)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_i)^2(1+\theta z_j)} + \sum_{j<i}\frac{(z_i)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_i)^2(1+\theta z_j)}.
\end{aligned}$$

By swapping indices $i$ and $j$ in the second right-hand term, we get

$$\varphi_{1,x}(\theta) = \sum_{i<j} \frac{(z_i)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_i)^2(1+\theta z_j)} + \sum_{i<j} \frac{(z_j)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_j)^2(1+\theta z_i)}.$$

A few calculus leads to:

$$\begin{aligned}\varphi_{1,x}(\theta) &= \sum_{i<j} \left( \frac{(z_i)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_i)^2(1+\theta z_j)} + \frac{(z_j)^2 x_{+i}x_{+j} - z_i z_j x_{+i}x_{+j}}{(1+\theta z_i)(1+\theta z_j)^2} \right) \\ &= \sum_{i<j} \left( \frac{x_{+i}x_{+j}}{(1+\theta z_i)(1+\theta z_j)} \right) \left( \frac{(z_i)^2 - z_i z_j}{(1+\theta z_i)} + \frac{(z_j)^2 - z_i z_j}{(1+\theta z_j)} \right).\end{aligned}$$

Since

$$\frac{(z_i)^2 - z_i z_j}{(1+\theta z_i)} + \frac{(z_j)^2 - z_i z_j}{(1+\theta z_j)} = \frac{(z_i)^2 - 2z_i z_j + (z_j)^2}{(1+\theta z_i)(1+\theta z_j)}, \tag{2.29}$$

we get

$$\varphi_{1,x}(\theta) = \sum_{i<j} \frac{x_{+i}x_{+j}(z_i - z_j)^2}{(1+\theta z_i)^2(1+\theta z_j)^2} \geqslant 0.$$

Thus the function $\varphi_x$ is an increasing function.

If $\theta^{(0)}$ is chosen such that $\theta^{(0)} < \theta^*$, then by using the properties of $\varphi_x$ demonstrated above, we will have $\theta^{(0)} < \varphi_x(\theta^{(0)}) = \theta^{(1)}$, $\theta^{(1)} < \varphi_x(\theta^{(1)}) = \theta^{(2)}$ and then the relationship $\theta^{(k)} < \varphi_x(\theta^{(k)}) = \theta^{(k+1)}$. By a similar reasoning, One can prove that if $\theta^{(0)} > \theta^*$ then $\theta^{(k)} > \varphi_x(\theta^{(k)}) = \theta^{(k+1)}$.

**iv)** For every $k > 0$, we have :

$$0 < \theta^{(k)} < a_{2,1} \times \sup_{u>0} \varphi_x(u) = a_{2,1} \times \lim_{u \to +\infty} \varphi_x(u) = \frac{a_{2,1}}{n} \sum_{m=1}^{r} \frac{x_{+m}}{z_m}. \tag{2.30}$$

The real sequence $\theta^{(k)}$ is monotonic and bounded. Thus its converge to $\theta^*$ the only fixed point of the function $\varphi_x$ that is also equal to the MLE $\hat{\theta}$. □

**Remark 2.4.1.** Using Lemma 2.4.2, we can conclude that the sequence $(\theta^{(k)})$ converges to $\hat{\theta}$. As each $\phi_j^{(k)}$, $j = 1, \ldots, r$, is the image of $\theta^{(k)}$ by the continuous mapping $G_{\phi,j}$ : $\theta \mapsto G_j(\theta, \boldsymbol{\phi})$, the sequence $\boldsymbol{\phi}^{(k)}$ also a limit that is $G_{\phi,j}(\hat{\theta}) = \hat{\phi}_j$. Thus, the vector $\boldsymbol{\beta}^{(k)} = (\theta^{(k)}, \boldsymbol{\phi}^{(k)})$ converges to the MLE $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})$. Theorem 2.4.1 is thus proved.

We may now prove Theorem 2.4.2. Because of the partition of the parameter vector $\boldsymbol{\beta}$ into two sub-parameters $\theta$ and $\boldsymbol{\phi}$, we consider the concentrated (or profile) likelihood function that is also commonly used in maximum likelihood estimation [66]. Considering that for a given value of $\hat{\theta}$, the MLE of the sub-parameter $\boldsymbol{\phi}$ is found as a function of $\hat{\theta}$ denoted $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}(\hat{\theta})$, the likelihood function $\ell(\boldsymbol{\beta}) = \ell(\theta, \boldsymbol{\phi})$ can be re-written as a function only of $\theta$,

$$\ell_c(\theta) = \ell(\theta, \boldsymbol{\phi}(\theta)) \tag{2.31}$$

that is called the concentrated (or profile) likelihood. Note that the profile likelihood is not a true likelihood, as it is not based directly on a probability distribution.

**Lemma 2.4.3.** *The concentrated (or profile) likelihood function is defined up to an additive constant by*

$$\ell_c(\theta) = x_{2+} \log \theta - \sum_{j=1}^{r} x_{+j} \log(1 + \theta z_j). \tag{2.32}$$

**Proof**. The expression (2.8) is equivalent to

$$\ell(\boldsymbol{\beta}) = \sum_{j=1}^{r} x_{+j} \log(\phi_j) + x_{2+} \log(\theta) - n \log(1 + \theta \sum_{m=1}^{r} z_m \phi_m)$$

and the relationship (2.20) enables to write

$$\begin{aligned}\ell_c(\theta) = \sum_{j=1}^{r} x_{+j} \log\left(\frac{x_{+j}}{1+\theta z_j}\right) - \sum_{j=1}^{r} x_{+j} \log\left(\sum_{m=1}^{r} \frac{x_{+m}}{1+\theta z_m}\right) + x_{2+} \log(\theta) \\ - n \log\left(\sum_{m=1}^{r} \frac{x_{+m}}{1+\theta z_m} + \sum_{m=1}^{r} \frac{\theta z_m x_{+m}}{1+\theta z_m}\right) + n \log\left(\sum_{m=1}^{r} \frac{x_{+m}}{1+\theta z_m}\right).\end{aligned}$$

After some manipulations with the second and the fourth terms, we get:

$$\begin{aligned}\ell_c(\theta) = \sum_{j=1}^{r} x_{+j} \log\left(\frac{x_{+j}}{1+\theta z_j}\right) - n \log\left(\sum_{m=1}^{r} \frac{x_{+m}}{1+\theta z_m}\right) + x_{2+} \log(\theta) \\ - n \log n + n \log\left(\sum_{m=1}^{r} \frac{x_{+m}}{1+\theta z_m}\right).\end{aligned}$$

Removing the second and the fifth terms and the constants, one gets the expression (2.32). □

**Example 2.4.1.** *Figure 2.2 shows an example of graphical representation of the concentrated log-likelihood for* $n = 5000$, $r = 3$, $\mathbf{Z} = (0.8, 1.5, 2.5)$ *and* $\boldsymbol{\beta} = (0.8, 0.4, 0.5, 0.1)$.

We may now proceed to the proof of Theorem 2.4.2.

### Proof of Theorem 2.4.2

The profile log-likelihood $\ell_c(\theta)$ is differentiable for every $\theta > 0$ and its derivative is

$$\begin{aligned}\ell_c'(\theta) &= \frac{x_{2+}}{\theta} - \sum_{j=1}^{r} \frac{x_{+j} z_j}{1+\theta z_j} = \frac{1}{\theta}\left(x_{2+} - \sum_{j=1}^{r}\left(x_{+j} - \frac{x_{+j}}{1+\theta z_j}\right)\right) \\ &= \frac{1}{\theta}\left(x_{2+} - n + \sum_{j=1}^{r} \frac{x_{+j}}{1+\hat{\theta} z_j}\right).\end{aligned}$$

Since $x_{1+} + x_{2+} = n$, we have

$$\ell_c'(\theta) = \theta^{-1}\ \psi(\theta)$$

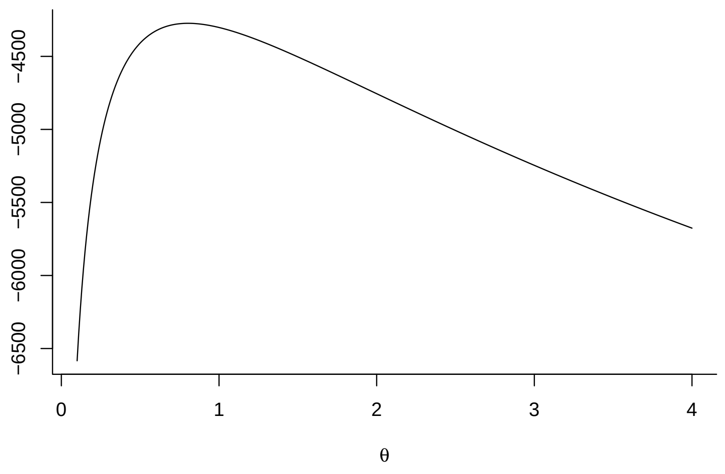

Figure 2.2. Graphical representation of the concentrated log-likelihood for $n = 5000$, $r = 3$, $\mathbf{Z} = (0.8, 1.5, 2.5)$ and $\boldsymbol{\beta} = (0.8, 0.4, 0.5, 0.1)$.

where the function $\psi$ is defined by Equation (2.26) (Lemma 2.4.1). Using this last lemma, we deduce that

$$\forall \theta \leqslant \theta^*, \quad \ell_c'(\theta) \geqslant 0 \quad \text{and} \quad \forall \theta \geqslant \theta^*, \quad \ell_c'(\theta) \leqslant 0$$

where $\theta^*$ is the MLE of $\theta$ and also the unique root of $\psi$. Hence the function $\ell_c$ is increasing on the interval $]0, \theta^*]$ and decreasing on $[\theta^*, +\infty[$. To finish the proof, we consider the two cases $\theta^{(0)} < \theta^*$ and $\theta^{(0)} > \theta^*$.

– If $\theta^{(0)} < \theta^*$, then we have proved that the sequence $\theta^{(k)}$ is increasing and still belongs to the interval $]0, \theta^*]$. Then $\theta^{(k)} \leqslant \theta^{(k+1)}$ and $\ell_c(\theta^{(k)}) \leqslant \ell_c(\theta^{(k+1)})$ because $\ell_c$ is increasing on $]0, \theta^*]$.

– If $\theta^{(0)} > \theta^*$, then the sequence $\theta^{(k)}$ is decreasing and still belongs to the interval $[\theta^*, +\infty[$. Then $\theta^{(k+1)} \leqslant \theta^{(k)}$ and $\ell_c(\theta^{(k+1)}) \geqslant \ell_c(\theta^{(k)})$ because $\ell_c$ is decreasing on $[\theta^*, +\infty[$.

In all the cases, we have

$$\ell(\boldsymbol{\beta}^{(k)}) = \ell_c(\theta^{(k)}) \leqslant \ell_c(\theta^{(k+1)}) = \ell(\boldsymbol{\beta}^{(k+1)})$$

and the proof of Theorem 2.4.2 is complete. □

## 2.5 Numerical experiments

This section focuses on the numerical study of the cyclic algorithm (CA). To our knowledge, three criteria are usually investigated on iterative algorithms: robustness (the algorithm should perform well for all reasonable choices of the initial guess), accuracy (the algorithm should be able to identify a solution near the true values with precision) and efficiency (the algorithm should not require too much computation time or storage).

If the number of iterations is approximately the same for different starting values $\hat{\Theta}^{(0)}$, then it can be said that the algorithm is robust. So in our experiments, the robustness will be checked trough the number of iterations. In order to evaluate the accuracy of the algorithm, we compute the mean squared error (MSE)

$$\text{MSE}(\hat{\boldsymbol{\beta}}) = \frac{1}{1+r}\left((\hat{\theta}-\theta^0)^2 + \sum_{j=1}^{r}(\hat{\boldsymbol{\phi}}_j - \boldsymbol{\phi}_j^0)^2\right) \tag{2.33}$$

with $\boldsymbol{\beta}^0 = (\theta^0, \boldsymbol{\phi}^0)$ the true parameter vector. Efficiency will be monitored by the central process unit (CPU) time computed in seconds.

We also compare the cyclic algorithm to some of the best available optimization algorithms on R software [87] and MATLAB. The methods selected for this comparison are the Newton-Raphson algorithm, the quasi-Newton BFGS algorithm [7, 25, 27, 93], the Nelder-Mead's algorithm [67], the Minorization-Majorization (MM) algorithm and the Interior Point algorithm (IP). The performances of the different algorithms in terms of computation time are compared trough their respective CPU time ratios calculated as the ratio between their mean duration and the mean duration of the cyclic algorithm. Thus, the CPU time ratio of the cyclic algorithm is always equal to 1.

The running times (or computation times) of an algorithm in the R software are calculated by combining the functions `Sys.time()` and `difftime`. For MATLAB software, they are obtained by using the two functions `tic` and `toc` placed either side of the algorithm whose execution time is to be measured. The BFGS and Nelder-Mead algorithm are implemented using the function `constrOptim.nl` of the `alabama` R package developed by Varadhan [100]. The Interior Point algorithm is implemented by using the function `fmincon` of the MATLAB Optimization toolbox.

### Implementation of the MM algorithm

We code the MM algorithm in R software using the following algorithm proposed by Mkhadri et al. [65].

**Remark 2.5.1.** The MM algorithm is a so-called cyclic MM algorithm that cycles through the parameters, updating one at a time instead of updating the whole vector at once. Mkhadri et al. [65] proved that at the step $k$ of the MM algorithm and for all $w > 0$, each of the updates $\theta^{(k)}$ and $\phi_j^{(k)}$, $j = 1, \dots, r$ is positive. They also proved that the condition $0 < \phi^{(k)} < 1$ implies $0 < \phi^{(k+1)} < 1$ provided the inequality

$$(x_{1j} + x_{2j}) \leqslant n + na^{(k)}\theta^{(k+1)}\left(z_j - \sum_{m=1}^{r} z_m \phi_m^{(k)}\right)$$

**Algorithm 2.2** Implementation of the MM algorithm [65]

The MM udpates are given by

$$\theta^{(k+1)} = \frac{w\theta^{(k)} + \sum_{j=1}^{r} x_{2j}}{w + na^{(k)} \sum_{j=1}^{r} z_j \phi_j^{(k)}}$$

and

$$\phi^{(k+1)} = \frac{(x_{1j} + x_{2j}) + w\phi_j^{(k)}}{w + n + na^{(k)}\theta^{(k+1)} \left( z_j - \sum_{m=1}^{r} z_m \phi_m^{(k)} \right)}, \quad j = 1, \ldots, r$$

where

$$a^{(k)} = \frac{1}{1 + \theta^{(k)} \sum_{m=1}^{r} z_m \phi_m^{(k)}}$$

and $w$ is a non-negative tuning parameter.

holds. They claim that this latter inequality condition is difficult to establish analytically, since it depends on the values of the observations $x_{ij}$ and $z_j$ for $i = 1, 2$, $j = 1, \ldots, r$. But, it seems to be often satisfied in practice.

### Implementation of the Newton's algorithm

We know that the Newton's algorithm is an algorithm with double facet. It is both an optimization algorithm and an algorithm for solving systems of equations. There exists a plethora of non-linear optimisation packages designed for R software but our search for an implementation of the algorithm of Newton-Raphson (NR) accepting both box constraints (lower/upper bounds) and equality constraints remained vain. It must be noted that most packages offering the Newton-Raphson's algorithm are either intended for unconstrained optimization or optimization with box constraints or equality constraints but not both. For example, Table 2.1 gives a few packages that provide an implementation of the NR method but with a few limitations.

Table 2.1. Some implementations of the NR algorithm in R software and their limitations

| Package | NR implementation | Limitation |
|---|---|---|
| `stats` | `nlm` | Only designed for unconstrained optimization |
| `maxLik` [33] | `maxNR` | Inequality constraints are not implemented |
| `Bhat` [54] | `newton` | Only accepts box-constraints |

We may also consider the non-linear system of equations (2.10) rather than the optimization problem (2.9). Most of the R packages designed for this purpose use the stopping criterion

$$\|F(\beta^{(k)})\| < \epsilon$$

where $\epsilon > 0$ is very small. In order to compare all the selected algorithms on identical bases, we must impose the same stopping criterion. Taking into account the specificities of the different selected algorithms, the common stopping criterion that was used is

$$|\ell(\boldsymbol{\beta}^{(k+1)}) - \ell(\boldsymbol{\beta}^{(k)})| < \epsilon.$$

We code the NR algorithm in R software so as to take into account this latter stopping criterion. In order to ensure that the positivity constraints existing on the parameters is respected, we opt for the following reparameterization:

$$\boldsymbol{\beta} = (\theta, \phi_1, \dots, \phi_r) = (\exp(\alpha), \exp(\eta_1), \dots, \exp(\eta_r)) = \Psi(\boldsymbol{\beta}_{\text{new}}) \tag{2.34}$$

where $\boldsymbol{\beta}_{\text{new}} = (\alpha, \eta_1, \dots, \eta_r)$ is the new vector parameter. This latter is no more subject to inequality constraints since its components can take any value in the set of real numbers $\mathbb{R}$. After simplification of the denominators, the non-linear system (2.10) may be rewritten

$$F(\boldsymbol{\beta}_{\text{new}}) = 0 \tag{2.35}$$

where the function $F$ is now defined from $\mathbb{R}^{r+1}$ in itself by

$$F(\boldsymbol{\beta}_{\text{new}}) = (F_0(\boldsymbol{\beta}_{\text{new}}), F_1(\boldsymbol{\beta}_{\text{new}}), \dots, F_r(\boldsymbol{\beta}_{\text{new}}))^T$$

and

$$\begin{aligned} F_0(\boldsymbol{\beta}_{\text{new}}) &= (x_{2+}) - (x_{1+})\mathrm{e}^{\alpha} \sum_{m=1}^{r} z_m \mathrm{e}^{\eta_m} \\ F_j(\boldsymbol{\beta}_{\text{new}}) &= (x_{+j}) \left(1 + \mathrm{e}^{\alpha} \sum_{m=1}^{r} z_m \mathrm{e}^{\eta_m}\right) - n\mathrm{e}^{\eta_j} \left(1 + \mathrm{e}^{\alpha} z_j\right), \quad j = 1, \dots, r. \end{aligned} \tag{2.36}$$

**Theorem 2.5.1.** *The Newton-Raphson's method applied to the estimation of* $\boldsymbol{\beta}_{\text{new}} = (\alpha, \boldsymbol{\eta})$ *corresponds to the iterative scheme*

$$-\left[\mathbf{J}(\boldsymbol{\beta}_{\text{new}}^{(k)})\right] \mathbf{p}^{(k)} = F(\boldsymbol{\beta}_{\text{new}}^{(k)}) \tag{2.37}$$

$$\boldsymbol{\beta}_{\text{new}}^{(k+1)} = \boldsymbol{\beta}_{\text{new}}^{(k)} + \mathbf{p}^{(k)} \tag{2.38}$$

*where*

$$\mathbf{J}(\boldsymbol{\beta}_{\text{new}}) = \begin{pmatrix} J_{00}(\boldsymbol{\beta}_{\text{new}}) & J_{01}(\boldsymbol{\beta}_{\text{new}}) & \cdots & J_{0r}(\boldsymbol{\beta}_{\text{new}}) \\ J_{10}(\boldsymbol{\beta}_{\text{new}}) & J_{11}(\boldsymbol{\beta}_{\text{new}}) & \cdots & J_{1r}(\boldsymbol{\beta}_{\text{new}}) \\ \vdots & \vdots & \ddots & \vdots \\ J_{r0}(\boldsymbol{\beta}_{\text{new}}) & J_{r1}(\boldsymbol{\beta}_{\text{new}}) & \cdots & J_{rr}(\boldsymbol{\beta}_{\text{new}}) \end{pmatrix},$$

$$\begin{aligned} J_{00}(\boldsymbol{\beta}_{\text{new}}) &= -x_{1+}\ \mathrm{e}^{\alpha} \sum_{m=1}^{r} z_m \mathrm{e}^{\eta_m}, \\ J_{0j}(\boldsymbol{\beta}_{\text{new}}) &= -x_{1+}\ \mathrm{e}^{\alpha} z_j \mathrm{e}^{\eta_j}, \qquad j = 1, \dots, r, \\ J_{i0}(\boldsymbol{\beta}_{\text{new}}) &= \mathrm{e}^{\alpha} \left(x_{+i} \sum_{m=1}^{r} z_m \mathrm{e}^{\eta_m} - n z_i \mathrm{e}^{\eta_i}\right), \qquad i = 1, \dots, r, \\ J_{ij}(\boldsymbol{\beta}_{\text{new}}) &= \mathrm{e}^{\eta_j} \left(x_{+i}\ \mathrm{e}^{\alpha} z_j - n(1 + \mathrm{e}^{\alpha} z_i)\ \delta_i^j\right), \qquad i, j = 1, \dots, r, \end{aligned}$$

*and* $\delta_i^j$ *represents the classical Kronecker's delta.*

**Proof**. The matrix $\mathbf{J}$ is the Jacobian matrix of $F$. Thus its components are obtained as follows:

$$J_{00}(\boldsymbol{\beta}_{\text{new}}) = \frac{\partial F_0}{\partial \alpha}, \quad J_{0j}(\boldsymbol{\beta}_{\text{new}}) = \frac{\partial F_0}{\partial \eta_j}, \quad j = 1, \ldots, r,$$

$$J_{i0}(\boldsymbol{\beta}_{\text{new}}) = \frac{\partial F_i}{\partial \alpha}, \quad i = 1, \ldots, r, \qquad J_{ij}(\boldsymbol{\beta}_{\text{new}}) = \frac{\partial F_i}{\partial \eta_j}, \quad i, j = 1, \ldots, r.$$

□

## 2.5.1 Data generation principle

Given $r$ (the number of accident types) and $n$ (the total number of accidents), we generate the components of vector $\mathbf{Z} = (z_1, \ldots, z_r)^T$ from a uniform random variable $\mathcal{U}(0.5; 2.5)$. We also generate the true value of parameter $\theta$, denoted $\theta^0$, as the mean of a uniform random variable $\mathcal{U}(0;1)$ and the true value of vector $\boldsymbol{\phi}$, denoted $\boldsymbol{\phi}^0 = (\phi_1^0, \ldots, \phi_r^0)^T$ from a Dirichlet distribution. Afterwards, we compute the true values

$$\pi_{1j}^0(\boldsymbol{\beta}) = \frac{\phi_j^0}{1 + \theta^0 \sum_{m=1}^r z_m \phi_m^0} \qquad j = 1, \ldots, r$$

and

$$\pi_{2j}^0(\boldsymbol{\beta}) = \frac{\theta^0 \, z_j \phi_j^0}{1 + \theta^0 \sum_{m=1}^r z_m \phi_m^0} \qquad j = 1, \ldots, r$$

linked to the multinomial distribution of $\mathbf{X}$. Finally, the data $\mathbf{x}$ is randomly generated from the multinomial distribution $\mathcal{M}(n; \boldsymbol{\pi}_1^0(\boldsymbol{\beta}^0), \boldsymbol{\pi}_2^0(\boldsymbol{\beta}^0))$.

## 2.5.2 Results

The results presented in this chapter correspond to $r \in \{3; 5\}$ and two values of $n$: a small value ($n = 50$) and a great value ($n = 5000$). In order to explore all the possible starting positions, we consider four different ways of setting the starting parameter vector $\boldsymbol{\beta}^{(0)} = (\theta^{(0)}, \boldsymbol{\phi}^{(0)})$. The parameter $\theta^{(0)}$ is randomly generated and the parameter vector $\boldsymbol{\phi}^{(0)}$ is randomly generated in four different ways:

$(I_1)$ Uniform: $\boldsymbol{\phi}^{(0)} = \left(\frac{1}{r}, \ldots, \frac{1}{r}\right)^T$.

$(I_2)$ Proportional I:

$$\boldsymbol{\phi}^{(0)} = \frac{1}{n}(\mathbf{x}_1 + \mathbf{x}_2) = \left(\frac{x_{11} + x_{21}}{n}, \ldots, \frac{x_{1r} + x_{2r}}{n}\right)^T.$$

$(I_3)$ Random: $\boldsymbol{\phi}^{(0)} = \mathbf{U} / \sum_{j=1}^r u_j$ where $\mathbf{U} = (u_1, \ldots, u_r)^T$ is a $r-$dimensional vector whose components are randomly generated from an uniform distribution $\mathcal{U}(0.05; 0.95)$.

$(I_4)$ Proportional II: $\boldsymbol{\phi}^{(0)} = \mathbf{x}_1 / \sum_{j=1}^r x_{1j}$.

**Remark 2.5.2.** The choice of these different initialization schemes needs some explanation. The first one is a quite logical way to initialize a vector of class probabilities whose sum is equal to 1. The second one ($I_2$) corresponds to the starting point used by N'Guessan and Truffier [78]. The fourth one corresponds to a natural initialization of $\boldsymbol{\phi}$ by construction of the model. At last the third one corresponds to the general case where the vector $\boldsymbol{\phi}^{(0)}$ is randomly chosen.

By combining these different values of $r$, $n$ and $\boldsymbol{\beta}^{(0)}$, we get 16 different scenarios. The results presented in the tables 2.4 to 2.11 correspond to the mean values obtained for 1000 replications for all the scenarios. For all the algorithms, the convergence criterion is set to $10^{-6}$, i.e. the iterations are stopped and convergence is declared if

$$|\ell(\boldsymbol{\beta}^{(k+1)}) - \ell(\boldsymbol{\beta}^{(k)})| < 10^{-6}.$$

#### 2.5.2.1 Numerical study of the cyclic algorithm

Table 2.2. Results of the cyclic algorithm for all the 16 000 simulated datasets (R software)

| | | $n = 50$ | | | $n = 5000$ | | |
|---|---|---|---|---|---|---|---|
| $r$ | Init | MSE | Iterations | CPU time | MSE | Iterations | CPU time |
| 3 | I1 | 9,7E-03 | 3,8 | 3,40E-04 | 8,28E-05 | 4,3 | 3,79E-04 |
| | I2 | 9,7E-03 | 3,5 | 3,62E-04 | 8,15E-05 | 4,0 | 3,38E-04 |
| | I3 | 8,7E-03 | 3,8 | 3,49E-04 | 8,10E-05 | 4,4 | 5,42E-04 |
| | I4 | 8,8E-03 | 3,3 | 3,45E-04 | 8,04E-05 | 3,3 | 3,12E-04 |
| 5 | I1 | 6,60E-03 | 3,9 | 3,87E-04 | 5,96E-05 | 4,5 | 5,04E-04 |
| | I2 | 7,15E-03 | 3,7 | 3,42E-04 | 6,16E-05 | 4,4 | 5,00E-04 |
| | I3 | 6,49E-03 | 3,9 | 3,82E-04 | 6,23E-05 | 4,6 | 5,45E-04 |
| | I4 | 6,68E-03 | 3,6 | 3,56E-04 | 6,10E-05 | 3,6 | 4,51E-04 |

Table 2.3. Results of the cyclic algorithm for all the 16 000 simulated datasets (MATLAB software)

| | | $n = 50$ | | | $n = 5000$ | | |
|---|---|---|---|---|---|---|---|
| $r$ | Init | MSE | Iterations | CPU time | MSE | Iterations | CPU time |
| 3 | I1 | 9,5E-03 | 3,8 | 7,50E-04 | 7,49E-05 | 4,3 | 7,57E-04 |
| | I2 | 9,0E-03 | 3,5 | 6,22E-04 | 7,86E-05 | 4,0 | 7,55E-04 |
| | I3 | 8,5E-03 | 3,8 | 7,48E-04 | 7,94E-05 | 4,4 | 8,00E-04 |
| | I4 | 8,6E-03 | 3,3 | 6,39E-04 | 7,78E-05 | 3,3 | 6,66E-04 |
| 5 | I1 | 6,51E-03 | 3,9 | 7,52E-04 | 5,95E-05 | 4,5 | 8,14E-04 |
| | I2 | 6,93E-03 | 3,8 | 7,59E-04 | 5,98E-05 | 4,4 | 7,86E-04 |
| | I3 | 7,15E-03 | 3,9 | 7,19E-04 | 5,99E-05 | 4,6 | 7,74E-04 |
| | I4 | 6,97E-03 | 3,5 | 7,16E-04 | 6,07E-05 | 3,6 | 6,61E-04 |

The overall performance of the cyclic algorithm is presented in Table 2.2 for R software and in Table 2.3 for MATLAB software. Inspection of these two tables shows that the mean number of iterations needed by the CA to achieve convergence is the same for the two software. It can also be noted that the MSE do not vary significantly from a software to the other. They are close to $10^{-2}$ for $n = 50$ and close to $10^{-4}$ for $n = 5000$.

One can understand the difference between the computation times since the CA is ran on two different software. At first glance, it can be said that these computation times are rather small but such a statement obviously remains relative. We can only really analyse them by comparing them with the computation times of other efficient algorithms. Therefore the efficiency will be studied later on when we compare the CA with the other algorithms.

As mentioned above, we use the MSE as an indicator of the accuracy of the algorithm. It is noticed that the MSE has an order of $10^{-2}$ when $n$ is small and $10^{-4}$ when $n$ is great. The results presented in Tables 2.4 to 2.11 indicate that the estimates produced by the CA are quite near to the true values and when $n$ is great, the estimates match the true values.

The results also suggest that the cyclic algorithm is robust towards the starting value (or initial guess). For each value of $r$, 8000 starting values (corresponding to 4 initialization schemes for $\boldsymbol{\phi}^{(0)}$, 2 values of $n$ and 1000 replications) are randomly selected in the parameters space. For all these values, the number of iterations lies between 3 and 4 on average. It can be noticed that for a given value of $n$, there is no significant increase of the number of iterations when the dimension of the parameters space $(r+1)$ increases. When $n$ increases, a slight increase of the mean number of iterations can be noticed. For a fixed value of $n$, 8000 starting values (corresponding to four initialization schemes for $\boldsymbol{\phi}^{(0)}$, two (02) values of $r$ and 1000 replications) are randomly selected in the parameters space and the MSE are quite the same even when $r$ varies from 3 to 5. It can also be noticed that the computation time doesn't vary too much (all the CPU times are near $10^{-3}$) and this strengthens the suggestion that the CA is robust.

#### 2.5.2.2 Comparison with other algorithms

Tables 2.4 to 2.11 report for each scenario, the value of the estimate $\hat{\boldsymbol{\beta}}$ produced by each of the compared algorithms, the minimum, maximum and mean values of the number of iterations needed to achieved convergence, the CPU time needed by each algorithm, the CPU time ratio, the final value of the log-likelihood and the MSE. In these tables, except the minimum and the maximum number of iterations, the other values presented are obtained as mean values over 1000 replications. Since the mean value does not tell everything about the variation of the values that we are studying, Figures 2.3 to 2.5 display the variation of the computation times, the MSE and the number of iterations for each of the selected algorithms.

**Analysis of the MSEs**

It can be seen that for all the scenarios, the CA, NR, MM and IP are accurate. For $n = 50$, their MSEs are near $10^{-2}$ and for $n = 5000$, they are close to $10^{-4}$. The Nelder-Mead's algorithm also is as accurate as the CA except for scenario where $r = 5$, $n = 5000$ and $\boldsymbol{\phi}^{(0)}$ is randomly chosen. This cannot be said of the BFGS algorithm. In general, the BFGS algorithm has a higher MSE than the other methods. This is seen on Figure 2.4. For example, for an initialization scheme $(I_2)$, $n = 5000$ and $r = 3$, the order of the MSE

Table 2.4. Results for scenario $r = 3$ and $\boldsymbol{\phi}^{(0)} = (1/r, \ldots, 1/r)^T$.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,530 | 0,530 | 0,530 | 0,530 | 0,530 | 0,528 | 0,528 |
| $\hat{\phi}_1$ | 0,052 | 0,052 | 0,052 | 0,052 | 0,052 | 0,052 | 0,052 |
| $\hat{\phi}_2$ | 0,499 | 0,499 | 0,499 | 0,499 | 0,499 | 0,492 | 0,492 |
| $\hat{\phi}_3$ | 0,449 | 0,449 | 0,449 | 0,449 | 0,449 | 0,456 | 0,456 |
| Min iter | 2 | 3 | 8 | 10 | 10 | 2 | 10 |
| Max iter | 5 | 6 | 44 | 11 | 12 | 6 | 22 |
| Mean iter. | 3,8 | 4,8 | 19,7 | 11,0 | 11,0 | 3,8 | 14,2 |
| CPU time | 3,40E-04 | 2,76E-03 | 2,21E-03 | 8,62E-02 | 2,50E-01 | 7,50E-04 | 2,86E-01 |
| CPU time ratio | 1 | 8 | 6 | 253 | 736 | 1 | 381 |
| Loglik | -84,4 | -84,4 | -84,4 | -84,4 | -84,4 | -84,6 | -84,6 |
| MSE | 9,67E-03 | 9,67E-03 | 9,67E-03 | 9,67E-03 | 9,67E-03 | 9,46E-03 | 9,46E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 |
| $\hat{\phi}_1$ | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 |
| $\hat{\phi}_2$ | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 |
| $\hat{\phi}_3$ | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 |
| Min iter | 3 | 4 | 11 | 15 | 13 | 3 | 8 |
| Max iter | 6 | 7 | 35 | 15 | 18 | 6 | 21 |
| Mean iter. | 4,3 | 5,2 | 24,1 | 15,0 | 15,2 | 4,3 | 14,5 |
| CPU time | 3,79E-04 | 2,99E-03 | 2,59E-03 | 2,00E-01 | 3,55E-01 | 7,57E-04 | 3,02E-01 |
| Time ratio | 1 | 8 | 7 | 527 | 937 | 1 | 399 |
| Loglik | -8096,9 | -8096,9 | -8096,9 | -8096,9 | -8096,9 | -8113,6 | -8113,6 |
| MSE | 8,28E-05 | 8,28E-05 | 8,28E-05 | 8,28E-05 | 8,28E-05 | 7,49E-05 | 7,49E-05 |

Table 2.5. Results for scenario $r = 3$ and $\boldsymbol{\phi}^{(0)} = (\mathbf{x}_1 + \mathbf{x}_2)/n$.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,536 | 0,536 | 0,536 | 0,536 | 0,536 | 0,528 | 0,528 |
| $\hat{\phi}_1$ | 0,050 | 0,050 | 0,050 | 0,050 | 0,050 | 0,052 | 0,052 |
| $\hat{\phi}_2$ | 0,498 | 0,498 | 0,498 | 0,498 | 0,498 | 0,489 | 0,489 |
| $\hat{\phi}_3$ | 0,452 | 0,452 | 0,452 | 0,452 | 0,452 | 0,459 | 0,459 |
| Min iter | 2 | 3 | 7 | 11 | 10 | 2 | 7 |
| Max iter | 5 | 5 | 33 | 11 | 12 | 5 | 22 |
| Mean iter. | 3,5 | 4,3 | 16,9 | 11,0 | 11,0 | 3,5 | 12,9 |
| CPU time | 3,62E-04 | 2,49E-03 | 1,90E-03 | 8,41E-02 | 2,44E-01 | 6,22E-04 | 2,66E-01 |
| CPU time ratio | 1 | 7 | 5 | 232 | 673 | 1 | 427 |
| Loglik | -83,8 | -83,8 | -83,8 | -83,8 | -83,8 | -84,2 | -84,2 |
| MSE | 9,67E-03 | 9,67E-03 | 9,67E-03 | 9,67E-03 | 9,67E-03 | 9,02E-03 | 9,02E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,499 | 0,499 | 0,499 | 0,611 | 0,499 | 0,500 | 0,500 |
| $\hat{\phi}_1$ | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 |
| $\hat{\phi}_2$ | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 |
| $\hat{\phi}_3$ | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 |
| Min iter | 2 | 3 | 10 | 1 | 2 | 2 | 7 |
| Max iter | 6 | 6 | 29 | 16 | 18 | 6 | 20 |
| Mean iter. | 4,0 | 4,6 | 22,2 | 14,2 | 14,5 | 4,0 | 12,9 |
| CPU time | 3,38E-04 | 2,55E-03 | 2,39E-03 | 1,86E-01 | 3,40E-01 | 7,55E-04 | 3,06E-01 |
| CPU time ratio | 1 | 8 | 7 | 551 | 1006 | 1 | 405 |
| Loglik | -8131,5 | -8131,5 | -8131,5 | -8215,2 | -8131,5 | -8124,1 | -8124,1 |
| MSE | 8,15E-05 | 8,15E-05 | 8,15E-05 | 8,02E-02 | 8,15E-05 | 7,86E-05 | 7,86E-05 |

Table 2.6. Results for scenario $r = 3$ and $\boldsymbol{\phi}^{(0)}$ randomly chosen.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,523 | 0,523 | 0,523 | 0,789 | 0,523 | 0,524 | 0,524 |
| $\hat{\phi}_1$ | 0,051 | 0,051 | 0,051 | 0,086 | 0,051 | 0,052 | 0,052 |
| $\hat{\phi}_2$ | 0,496 | 0,496 | 0,496 | 0,476 | 0,496 | 0,497 | 0,497 |
| $\hat{\phi}_3$ | 0,454 | 0,454 | 0,454 | 0,438 | 0,454 | 0,451 | 0,451 |
| Min iter | 2 | 3 | 9 | 1 | 2 | 2 | 9 |
| Max iter | 5 | 7 | 41 | 12 | 11 | 6 | 25 |
| Mean iter. | 3,8 | 4,9 | 19,6 | 9,7 | 10,1 | 3,8 | 14,4 |
| CPU time | 3,49E-04 | 2,77E-03 | 2,15E-03 | 7,46E-02 | 2,32E-01 | 7,48E-04 | 2,90E-01 |
| CPU time ratio | 1 | 8 | 6 | 214 | 663 | 1 | 387 |
| Loglik | -84,0 | -84,0 | -84,0 | -88,8 | -84,0 | -84,2 | -84,2 |
| MSE | 8,71E-03 | 8,71E-03 | 8,71E-03 | 2,13E-01 | 8,71E-03 | 8,52E-03 | 8,52E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,501 | 0,501 | 0,501 | 0,728 | 0,501 | 0,500 | 0,500 |
| $\hat{\phi}_1$ | 0,019 | 0,019 | 0,019 | 0,057 | 0,019 | 0,019 | 0,019 |
| $\hat{\phi}_2$ | 0,513 | 0,513 | 0,513 | 0,489 | 0,513 | 0,513 | 0,513 |
| $\hat{\phi}_3$ | 0,468 | 0,468 | 0,468 | 0,455 | 0,468 | 0,468 | 0,468 |
| Min iter | 3 | 3 | 13 | 1 | 2 | 3 | 9 |
| Max iter | 7 | 7 | 34 | 16 | 17 | 7 | 22 |
| Mean iter. | 4,4 | 5,2 | 24,1 | 13,2 | 13,7 | 4,4 | 14,2 |
| CPU time | 5,42E-04 | 4,41E-03 | 4,05E-03 | 2,72E-01 | 5,07E-01 | 8,00E-04 | 3,00E-01 |
| CPU time ratio | 1 | 8 | 7 | 501 | 936 | 1 | 375 |
| Loglik | -8063,2 | -8063,2 | -8063,2 | -8526,0 | -8063,2 | -8082,6 | -8082,6 |
| MSE | 8,10E-05 | 8,10E-05 | 8,10E-05 | 1,71E-01 | 8,12E-05 | 7,94E-05 | 7,94E-05 |

Table 2.7. Results for scenario $r = 3$ and $\boldsymbol{\phi}^{(0)} = \mathbf{x}_1 / \sum_{m=1}^{r} x_{1m}$.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,525 | 0,525 | 0,525 | 0,575 | 0,525 | 0,528 | 0,528 |
| $\hat{\phi}_1$ | 0,052 | 0,052 | 0,052 | 0,051 | 0,052 | 0,051 | 0,051 |
| $\hat{\phi}_2$ | 0,491 | 0,491 | 0,491 | 0,491 | 0,491 | 0,497 | 0,497 |
| $\hat{\phi}_3$ | 0,457 | 0,457 | 0,457 | 0,457 | 0,457 | 0,452 | 0,452 |
| Min iter | 2 | 3 | 6 | 1 | 2 | 2 | 8 |
| Max iter | 5 | 5 | 33 | 11 | 12 | 5 | 21 |
| Mean iter. | 3,3 | 4,2 | 17,3 | 10,8 | 10,8 | 3,3 | 12,9 |
| CPU time | 3,45E-04 | 2,35E-03 | 2,00E-03 | 8,08E-02 | 2,38E-01 | 6,39E-04 | 2,80E-01 |
| CPU time ratio | 1 | 7 | 6 | 234 | 690 | 1 | 438 |
| Loglik | -83,8 | -83,8 | -83,8 | -84,2 | -83,8 | -84,0 | -84,0 |
| MSE | 8,78E-03 | 8,78E-03 | 8,79E-03 | 4,52E-02 | 8,79E-03 | 8,61E-03 | 8,61E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,500 | 0,500 | 0,500 | 0,586 | 0,500 | 0,500 | 0,500 |
| $\hat{\phi}_1$ | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 |
| $\hat{\phi}_2$ | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 |
| $\hat{\phi}_3$ | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 |
| Min iter | 2 | 3 | 9 | 1 | 2 | 2 | 7 |
| Max iter | 5 | 5 | 26 | 16 | 18 | 5 | 19 |
| Mean iter. | 3,3 | 4,0 | 19,0 | 14,4 | 14,7 | 3,3 | 12,9 |
| CPU time | 3,12E-04 | 2,14E-03 | 1,92E-03 | 1,91E-01 | 3,39E-01 | 6,66E-04 | 3,05E-01 |
| CPU time ratio | 1 | 7 | 6 | 611 | 1087 | 1 | 459 |
| Loglik | -8095,0 | -8095,0 | -8095,0 | -8157,0 | -8095,0 | -8088,3 | -8088,3 |
| MSE | 8,04E-05 | 8,04E-05 | 8,04E-05 | 6,36E-02 | 8,04E-05 | 7,78E-05 | 7,78E-05 |

Table 2.8. Results for scenario $r = 5$ and $\boldsymbol{\phi}^{(0)} = (1/r, \ldots, 1/r)^T$.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,537 | 0,537 | 0,537 | 0,537 | 0,537 | 0,534 | 0,534 |
| $\hat{\phi}_1$ | 0,141 | 0,141 | 0,141 | 0,141 | 0,141 | 0,141 | 0,141 |
| $\hat{\phi}_2$ | 0,042 | 0,042 | 0,042 | 0,042 | 0,042 | 0,043 | 0,043 |
| $\hat{\phi}_3$ | 0,214 | 0,214 | 0,214 | 0,214 | 0,214 | 0,214 | 0,214 |
| $\hat{\phi}_4$ | 0,230 | 0,230 | 0,230 | 0,230 | 0,230 | 0,227 | 0,227 |
| $\hat{\phi}_5$ | 0,374 | 0,374 | 0,374 | 0,374 | 0,374 | 0,376 | 0,376 |
| Min iter | 2 | 3 | 8 | 11 | 10 | 2 | 13 |
| Max iter | 5 | 6 | 41 | 13 | 12 | 5 | 30 |
| Mean iter. | 3,9 | 4,7 | 19,6 | 11,0 | 11,0 | 3,9 | 18,3 |
| CPU time | 3,87E-04 | 4,15E-03 | 2,35E-03 | 1,06E-01 | 6,36E-01 | 7,52E-04 | 3,06E-01 |
| CPU time ratio | 1 | 11 | 6 | 275 | 1644 | 1 | 407 |
| Loglik | -113,1 | -113,1 | -113,1 | -113,1 | -113,1 | -113,0 | -113,0 |
| MSE | 6,60E-03 | 6,60E-03 | 6,60E-03 | 6,60E-03 | 6,60E-03 | 6,51E-03 | 6,51E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 | 0,499 |
| $\hat{\phi}_1$ | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,141 |
| $\hat{\phi}_2$ | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,010 |
| $\hat{\phi}_3$ | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,220 |
| $\hat{\phi}_4$ | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,237 |
| $\hat{\phi}_5$ | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,392 |
| Min iter | 2 | 3 | 13 | 15 | 12 | 3 | 13 |
| Max iter | 6 | 6 | 33 | 15 | 18 | 6 | 27 |
| Mean iter. | 4,5 | 5,0 | 24,0 | 15,0 | 15,0 | 4,5 | 17,5 |
| CPU time | 5,04E-04 | 5,99E-03 | 3,77E-03 | 2,69E-01 | 1,49E+00 | 8,14E-04 | 3,12E-01 |
| CPU time ratio | 1 | 12 | 7 | 533 | 2950 | 1 | 384 |
| Loglik | -10924,3 | -10924,3 | -10924,3 | -10924,3 | -10924,3 | -10954,2 | -10971,9 |
| MSE | 5,96E-05 | 5,96E-05 | 5,96E-05 | 5,96E-05 | 5,96E-05 | 5,95E-05 | 7,01E-05 |

corresponding to BFGS is $10^{-1}$ while others are $10^{-4}$. This is also seen for initialization scheme $(I_3)$.

### Analysis of the number of iterations

As far as the robustness is concerned, it can be seen that the CA and the NR almost have the same number of iterations (approximatively between 3 and 5) which are the lowest. The MM, BFGS and NM algorithms use approximatively 3 to 4 times more iterations than the CA. It can be noticed that the MM algorithm always has the higher number of iterations.

### Analysis of the computation times

The results suggest that the CA algorithm is efficient and it needs much less computation time than the other algorithms. Indeed, none of the CPU time ratios is lower than 1 which means that none of the algorithm needs less computation time than CA. Among the six algorithms selected for comparison, the CA, MM and NR are by far the most efficient. On average and in all the cases, the CA is 5 to 8 times quicker than MM algorithm and when $r$ varies from 3 to 5, no significant increase is observed in the number of iterations of the MM algorithm. Despite having their numbers of iterations very close, the CA is 7 to 12

Table 2.9. Results for scenario $r = 5$ and $\boldsymbol{\phi}^{(0)} = (\mathbf{x}_1 + \mathbf{x}_2)/n$.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,542 | 0,542 | 0,542 | 0,542 | 0,542 | 0,540 | 0,540 |
| $\hat{\phi}_1$ | 0,139 | 0,139 | 0,139 | 0,139 | 0,139 | 0,142 | 0,142 |
| $\hat{\phi}_2$ | 0,043 | 0,043 | 0,043 | 0,043 | 0,043 | 0,042 | 0,042 |
| $\hat{\phi}_3$ | 0,212 | 0,212 | 0,212 | 0,212 | 0,212 | 0,214 | 0,214 |
| $\hat{\phi}_4$ | 0,230 | 0,230 | 0,230 | 0,230 | 0,230 | 0,227 | 0,227 |
| $\hat{\phi}_5$ | 0,376 | 0,376 | 0,376 | 0,376 | 0,376 | 0,375 | 0,375 |
| Min iter | 3 | 3 | 8 | 10 | 10 | 3 | 12 |
| Max iter | 5 | 5 | 38 | 12 | 12 | 5 | 26 |
| Mean iter. | 3,7 | 4,4 | 17,8 | 11,0 | 11,0 | 3,8 | 16,5 |
| CPU time | 3,42E-04 | 3,86E-03 | 1,96E-03 | 1,00E-01 | 6,07E-01 | 7,59E-04 | 3,22E-01 |
| CPU time ratio | 1 | 11 | 6 | 293 | 1776 | 1 | 425 |
| Loglik | -113,2 | -113,2 | -113,2 | -113,2 | -113,2 | -112,9 | -112,9 |
| MSE | 7,15E-03 | 7,15E-03 | 7,15E-03 | 7,15E-03 | 7,15E-03 | 6,93E-03 | 6,93E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,500 | 0,500 | 0,500 | 0,515 | 0,500 | 0,500 | 0,499 |
| $\hat{\phi}_1$ | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,141 |
| $\hat{\phi}_2$ | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,010 |
| $\hat{\phi}_3$ | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,221 |
| $\hat{\phi}_4$ | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,236 |
| $\hat{\phi}_5$ | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,392 |
| Min iter | 3 | 4 | 14 | 1 | 3 | 3 | 12 |
| Max iter | 6 | 6 | 30 | 15 | 17 | 6 | 28 |
| Mean iter. | 4,4 | 4,8 | 23,3 | 14,9 | 14,9 | 4,4 | 17,7 |
| CPU time | 5,00E-04 | 5,52E-03 | 3,49E-03 | 2,56E-01 | 1,42E+00 | 7,86E-04 | 3,13E-01 |
| CPU time ratio | 1 | 11 | 7 | 512 | 2846 | 1 | 398 |
| Loglik | -10939,9 | -10939,9 | -10939,9 | -10952,5 | -10940,2 | -10924,6 | -10942,3 |
| MSE | 6,16E-05 | 6,16E-05 | 6,16E-05 | 7,61E-03 | 6,89E-05 | 5,98E-05 | 7,08E-05 |

times quicker than the NR. This is understandable because, at each iteration of the NR algorithm a matrix inversion is performed through the resolution of a linear system. We can notice that when the dimension of the parameters space varies from $r + 1 = 4$ to 6, there is an increase in the CPU time ratio.

The BFGS and IP algorithms are far behind the first three algorithms. The CA is approximately 214 (respectively 375) to 613 (respectively 460) times quicker than BFGS (respectively IP). The Nelder-Mead's algorithm is by far the less performing of the six algorithms. The CA is 663 to 3389 times quicker than the NM algorithm.

### 2.5.3 Illustration of Theorems 2.4.1 and 2.4.2

We run the cyclic algorithm (2.24) from four different initial values of $\theta^{(0)}$ that are 0.2, 0.4, 0.7 and 1. For each $\theta^{(0)}$ the successive values of $\theta^{(k)}$ are reported in Table 2.12 and displayed on Figure 2.6. The true parameters are $n = 1000$, $\theta^0 = 0.5$ and $\boldsymbol{\phi}^0 = (0.15, 0.25, 0.6)^T$. Note that for each of the four values of $\theta^{(0)}$ the CA took only four iterations to satisfy the convergence criterion $|\ell(\boldsymbol{\beta}^{(k+1)}) - \ell(\boldsymbol{\beta}^{(k)})| < 10^{-5}$.

It can be seen that when $\theta^{(0)}$ is lower than the true value $\theta^0 = 0.5$, the sequence $\theta^{(k)}$ is increasing while it decreases when $\theta^{(0)} > \theta^0$. The corresponding values of the log-likelihood

Table 2.10. Results for scenario $r = 5$ and $\boldsymbol{\phi}^{(0)}$ randomly chosen.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,532 | 0,532 | 0,532 | 0,792 | 0,531 | 0,540 | 0,540 |
| $\hat{\phi}_1$ | 0,142 | 0,142 | 0,142 | 0,150 | 0,142 | 0,142 | 0,142 |
| $\hat{\phi}_2$ | 0,043 | 0,043 | 0,043 | 0,062 | 0,044 | 0,042 | 0,042 |
| $\hat{\phi}_3$ | 0,210 | 0,210 | 0,210 | 0,207 | 0,212 | 0,214 | 0,214 |
| $\hat{\phi}_4$ | 0,231 | 0,231 | 0,231 | 0,229 | 0,231 | 0,226 | 0,226 |
| $\hat{\phi}_5$ | 0,375 | 0,375 | 0,375 | 0,352 | 0,371 | 0,375 | 0,375 |
| Min iter | 2 | 4 | 9 | 1 | 4 | 2 | 13 |
| Max iter | 5 | 6 | 36 | 12 | 16 | 5 | 32 |
| Mean iter. | 3,9 | 4,8 | 19,8 | 9,7 | 10,6 | 3,9 | 19,1 |
| CPU time | 3,82E-04 | 4,06E-03 | 2,20E-03 | 8,96E-02 | 6,05E-01 | 7,19E-04 | 3,11E-01 |
| CPU time ratio | 1 | 11 | 6 | 235 | 1585 | 1 | 433 |
| Loglik | -113,0 | -113,0 | -113,0 | -117,4 | -113,1 | -112,8 | -112,8 |
| MSE | 6,49E-03 | 6,49E-03 | 6,49E-03 | 1,36E-01 | 6,59E-03 | 7,15E-03 | 7,15E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,500 | 0,500 | 0,500 | 0,835 | 0,500 | 0,500 | 0,500 |
| $\hat{\phi}_1$ | 0,142 | 0,142 | 0,142 | 0,150 | 0,145 | 0,142 | 0,141 |
| $\hat{\phi}_2$ | 0,003 | 0,003 | 0,003 | 0,033 | 0,004 | 0,003 | 0,010 |
| $\hat{\phi}_3$ | 0,222 | 0,222 | 0,222 | 0,219 | 0,223 | 0,222 | 0,220 |
| $\hat{\phi}_4$ | 0,238 | 0,238 | 0,238 | 0,235 | 0,239 | 0,238 | 0,236 |
| $\hat{\phi}_5$ | 0,395 | 0,395 | 0,395 | 0,364 | 0,389 | 0,395 | 0,392 |
| Min iter | 2 | 3 | 14 | 1 | 3 | 3 | 12 |
| Max iter | 6 | 7 | 32 | 15 | 91 | 6 | 31 |
| Mean iter. | 4,6 | 5,1 | 24,2 | 12,8 | 14,1 | 4,6 | 18,2 |
| CPU time | 5,45E-04 | 6,62E-03 | 4,12E-03 | 2,45E-01 | 1,50E+00 | 7,74E-04 | 3,06E-01 |
| CPU time ratio | 1 | 12 | 8 | 449 | 2748 | 1 | 396 |
| Loglik | -10907,7 | -10907,7 | -10907,7 | -11508,4 | -10918,4 | -10929,2 | -10946,9 |
| MSE | 6,23E-05 | 6,23E-05 | 6,23E-05 | 1,77E-01 | 2,38E-04 | 5,99E-05 | 7,07E-05 |

$\ell(\boldsymbol{\beta}^{(k)})$ are reported in Table 2.13 and displayed on Figure 2.7. It can be seen that the log-likelihood is increasing over the iterations until convergence is achieved.

## 2.6 Conclusion

In this chapter, we presented some numerical convergence properties of a cyclic iterative algorithm (CA) for constrained maximum likelihood estimation of the parameters of a multinomial model applied to the modelling of the mean effect of a road safety measure on accidents risks. This algorithm is very simple to program without any matrix inversion. The results of the numerical experiments performed in this chapter suggest that this algorithm is robust towards the starting values, efficient and accurate. Moreover, the comparison of the performance of the cyclic algorithm to some of the best available optimization algorithms like MM and Newton-Raphson's algorithms suggest that it is as accurate as the others and most importantly that it is much more faster as far as the convergence is concerned.

Since this chapter has focused only on algorithmic aspects, the next chapter will concentrate on the stochastic level. More precisely we will investigate whether the maximum likelihood estimator obtained from the CA is consistent or not. A second natural exten-

Table 2.11. Results for scenario $r = 5$ and $\boldsymbol{\phi}^{(0)} = \mathbf{x}_1 / \sum_{m=1}^{r} x_{1m}$.

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | NR | MM | BFGS | NM | CA | IP |
| $n = 50$ | | | | | | | |
| $\hat{\theta}$ | 0,530 | 0,530 | 0,530 | 0,540 | 0,530 | 0,533 | 0,533 |
| $\hat{\phi}_1$ | 0,143 | 0,143 | 0,143 | 0,143 | 0,143 | 0,145 | 0,145 |
| $\hat{\phi}_2$ | 0,043 | 0,043 | 0,043 | 0,043 | 0,043 | 0,042 | 0,042 |
| $\hat{\phi}_3$ | 0,214 | 0,214 | 0,214 | 0,214 | 0,214 | 0,208 | 0,208 |
| $\hat{\phi}_4$ | 0,226 | 0,226 | 0,226 | 0,226 | 0,226 | 0,231 | 0,231 |
| $\hat{\phi}_5$ | 0,374 | 0,374 | 0,374 | 0,375 | 0,375 | 0,375 | 0,375 |
| Min iter | 2 | 3 | 9 | 1 | 5 | 2 | 12 |
| Max iter | 5 | 5 | 43 | 13 | 12 | 5 | 27 |
| Mean iter. | 3,6 | 4,3 | 18,5 | 11,0 | 11,0 | 3,5 | 16,5 |
| CPU time | 3,56E-04 | 3,83E-03 | 2,11E-03 | 1,05E-01 | 6,34E-01 | 7,16E-04 | 3,19E-01 |
| CPU time ratio | 1 | 11 | 6 | 295 | 1780 | 1 | 445 |
| Loglik | -113,0 | -113,0 | -113,0 | -113,1 | -113,0 | -113,0 | -113,0 |
| MSE | 6,68E-03 | 6,68E-03 | 6,68E-03 | 1,18E-02 | 6,69E-03 | 6,97E-03 | 6,97E-03 |
| $n = 5000$ | | | | | | | |
| $\hat{\theta}$ | 0,500 | 0,500 | 0,500 | 0,502 | 0,500 | 0,500 | 0,499 |
| $\hat{\phi}_1$ | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,141 |
| $\hat{\phi}_2$ | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,010 |
| $\hat{\phi}_3$ | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,220 |
| $\hat{\phi}_4$ | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,236 |
| $\hat{\phi}_5$ | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,392 |
| Min iter | 2 | 3 | 11 | 1 | 8 | 2 | 12 |
| Max iter | 5 | 5 | 27 | 15 | 17 | 5 | 27 |
| Mean iter. | 3,6 | 4,0 | 19,8 | 15,0 | 15,0 | 3,6 | 17,5 |
| CPU time | 4,51E-04 | 4,99E-03 | 3,27E-03 | 2,77E-01 | 1,53E+00 | 6,61E-04 | 3,04E-01 |
| CPU time ratio | 1 | 11 | 7 | 613 | 3389 | 1 | 460 |
| Loglik | -10954,7 | -10954,7 | -10954,7 | -10955,8 | -10954,7 | -10929,7 | -10947,6 |
| MSE | 6,10E-05 | 6,10E-05 | 6,10E-05 | 4,00E-04 | 6,10E-05 | 6,07E-05 | 7,13E-05 |

Table 2.12. Successive values of $\theta^{(k)}$ for different initial point $\theta^{(0)}$

| $\theta^{(0)}$ | $\theta^{(1)}$ | $\theta^{(2)}$ | $\theta^{(3)}$ | $\theta^{(4)}$ |
|---|---|---|---|---|
| 0.2 | 0.4730594 | 0.4942616 | 0.4956170 | 0.4957025 |
| 0.4 | 0.4893110 | 0.4953036 | 0.4956828 | 0.4957067 |
| 0.7 | 0.5071791 | 0.4964261 | 0.4957535 | 0.4957111 |
| 1.0 | 0.5201269 | 0.4972247 | 0.4958037 | 0.4957143 |

Table 2.13. Evolution of the log-likelihood $\ell(\boldsymbol{\beta}^{(k)})$ for different initial point $\theta^{(0)}$

| $\ell(\theta^{(0)})$ | $\ell(\theta^{(1)})$ | $\ell(\theta^{(2)})$ | $\ell(\theta^{(3)})$ | $\ell(\theta^{(4)})$ |
|---|---|---|---|---|
| -1819.227 | -1727.530 | -1727.277 | -1727.276 | -1727.276 |
| -1732.608 | -1727.296 | -1727.276 | -1727.276 | -1727.276 |
| -1741.163 | -1727.337 | -1727.276 | -1727.276 | -1727.276 |
| -1784.310 | -1727.545 | -1727.277 | -1727.276 | -1727.276 |

sion for this chapter will be the generalization of the CA to the general model designed by N'Guessan et al. [73] in the framework of the modelling of a road safety measure applied simultaneously to different experimental sites and several accident types. We will then

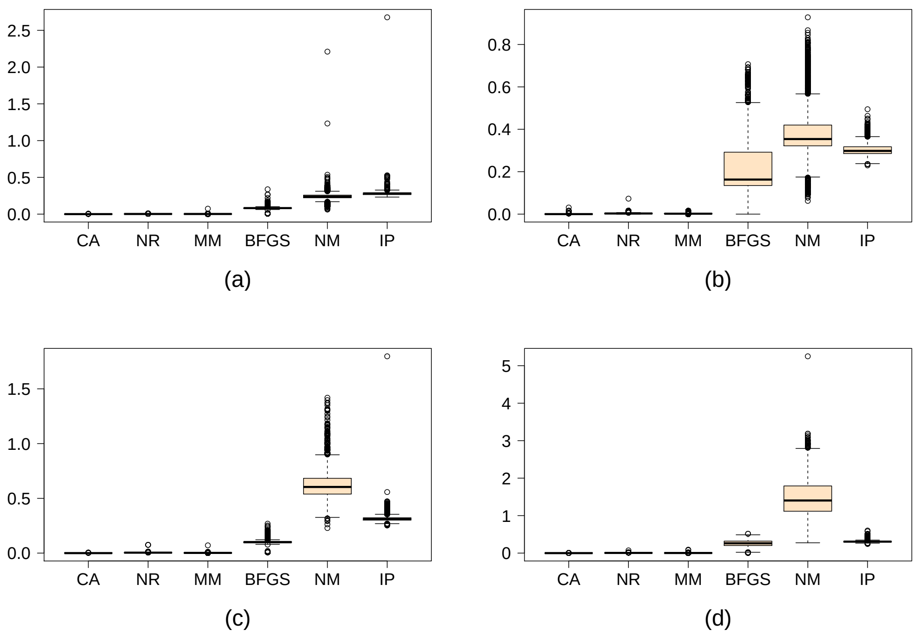

Figure 2.3. Comparison of the CPU times in seconds needed by the selected algorithms to achieve convergence : (a) $r = 3$ and $n = 50$, (b) $r = 3$ and $n = 5000$, (c) $r = 5$ and $n = 50$, (d) $r = 5$ and $n = 5000$.

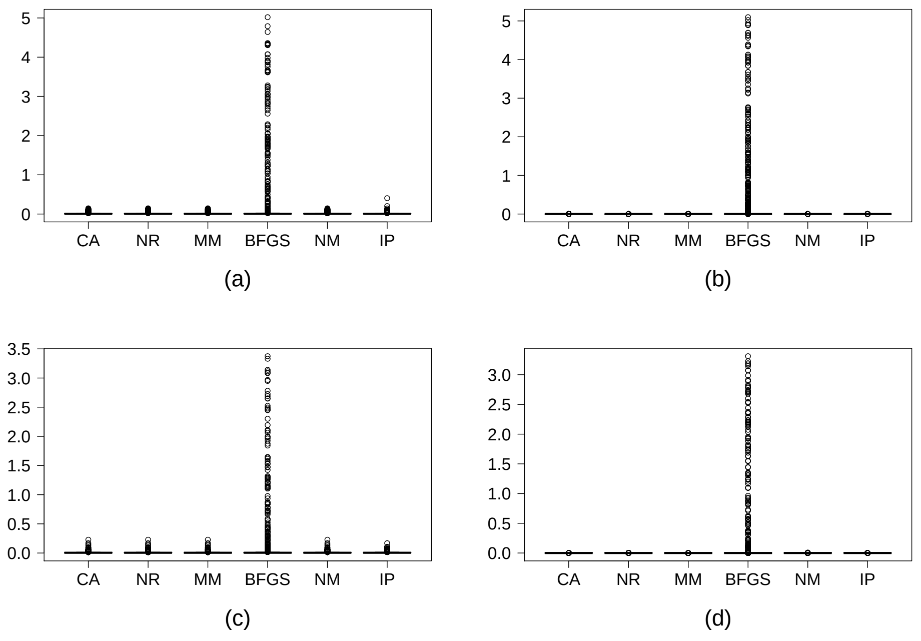

Figure 2.4. Comparison of the MSEs of the different algorithms : (a) $r = 3$ and $n = 50$, (b) $r = 3$ and $n = 5000$, (c) $r = 5$ and $n = 50$, (d) $r = 5$ and $n = 5000$.

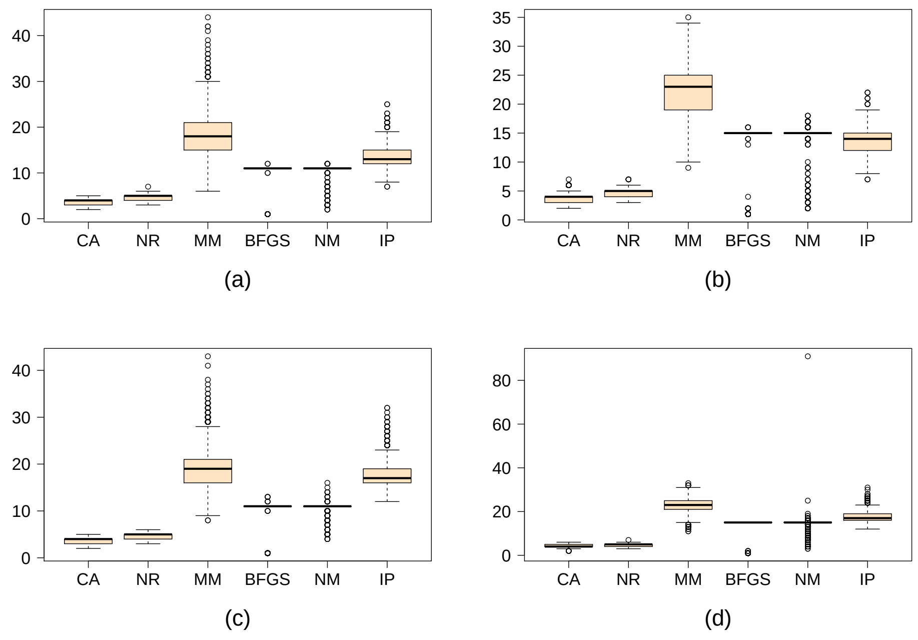

Figure 2.5. Comparison of the number of iterations needed by the selected algorithms to achieve convergence : (a) $r = 3$ and $n = 50$, (b) $r = 3$ and $n = 5000$, (c) $r = 5$ and $n = 50$, (d) $r = 5$ and $n = 5000$.

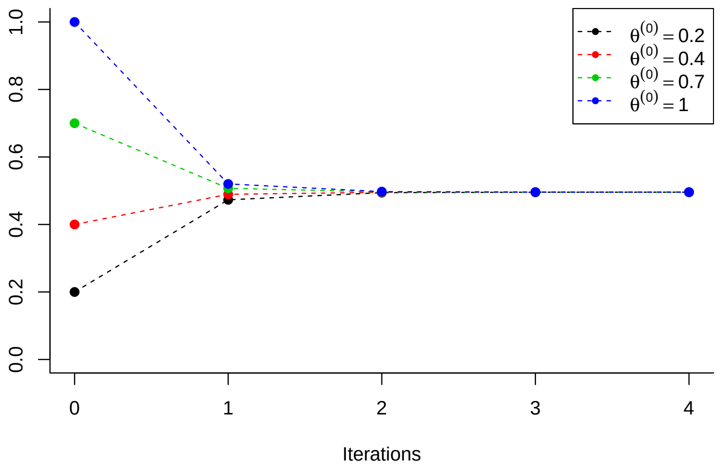

Figure 2.6. Successive values of $\theta^{(k)}$ for different values of the initial point $\theta^{(0)}$. The colors black, red, green and blue respectively correspond to $\theta^{(0)} = 0.2$, $\theta^{(0)} = 0.4$, $\theta^{(0)} = 0.7$ and $\theta^{(0)} = 1$.

have to compare the generalized version of the CA algorithm with some efficient algorithms like Newton-Raphson's and MM algorithms and also those available on R and MATLAB

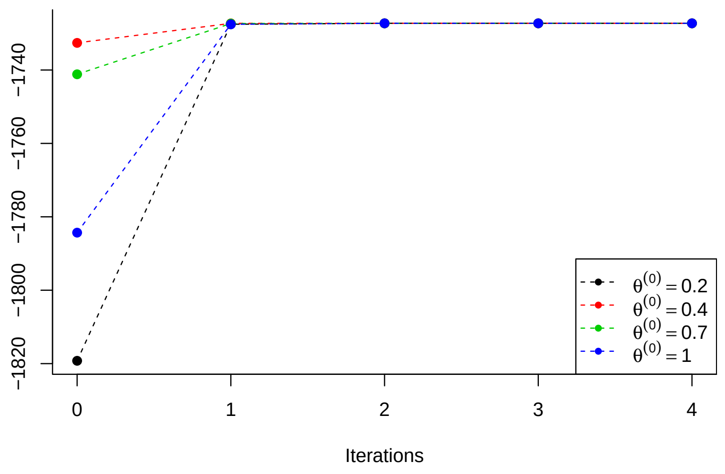

Figure 2.7. Evolution of the log-likelihood $\ell(\boldsymbol{\beta}^{(k)})$ for different values of the initial point $\theta^{(0)}$. The colors black, red, green and blue respectively correspond to $\theta^{(0)} = 0.2$, $\theta^{(0)} = 0.4$, $\theta^{(0)} = 0.7$ and $\theta^{(0)} = 1$.

software.

# Chapter 3

# Strong consistency of the cyclic maximum likelihood estimator

This chapter is an extended version of our article [26].

## 3.1 Introduction

Let $\mathbf{X}$ be a $\mathbb{R}^d$ valued random variable defined on a probability space $(\Omega, \mathcal{A}, \mathrm{P})$ with probability density function depending on a vector parameter $\boldsymbol{\beta}$. The Maximum Likelihood Estimator (MLE) $\hat{\beta}_n$ of $\beta$ can be obtained by solving the optimization problem

$$\hat{\boldsymbol{\beta}}_n = \underset{\boldsymbol{\beta}\in\mathbb{S}}{\operatorname{argmax}}\, \ell(\boldsymbol{\beta})$$

where $\ell$ is the log-likelihood function calculated on a sample of $n$ i.i.d. observations of $\mathbf{X}$ and $\mathbb{S}$ is the parameter space (the set of all possible values of $\boldsymbol{\beta}$).

One of the most desired properties of the estimator $\hat{\boldsymbol{\beta}}_n$ is its consistency i.e. its asymptotic convergence to the true value $\boldsymbol{\beta}^0$ of $\boldsymbol{\beta}$. This property has been the subject of many books and papers (see e.g. [8, 14, 23, 24, 30, 38, 39, 48, 50, 64, 66, 88, 91, 99, 102, 104]). The main result on the strong consistency was established by Wald [102] who gave regularity conditions under which the MLE is strongly consistent. However, all these conditions may be hard to check in practice if the dimension of the parameter space is large and the probability density function (or the likelihood) takes some complex forms. Nevertheless introducing modifications in Wald's work, authors among which [24, 38, 91, 99] obtained useful results on the consistency under less restrictive conditions. Van der Vaart [99] established general consistency properties of $M$-estimators presenting the MLE as a special case of $M$-estimators. But it is still possible that the MLE is not consistent even when it exists, as shown by examples given in [3, 23, 39, 48].

The present work is motivated by our need to provide a proof of the strong convergence property of the MLE of $\boldsymbol{\beta}$ proposed by N'Guessan and Truffier [78] and N'Guessan [71] for statistical analysis of accident data on an experimental site where observed accidents can be classified into $r$ mutually exclusive categories, $r \in \mathbb{N}^*$. In their before-after study in order to assess the impact of a measure on the occurrence of accidents, N'Guessan and Truffier [78] considered a random vector $\mathbf{X} = (X_{11}, \dots, X_{1r}, X_{21}, \dots, X_{2r})^T$ whose components are positive non-zero discrete random variables such that $X_{1j}$ (resp. $X_{2j}$), $j = 1, \dots, r$, represents the number of crashes of type $j$ occurred in the "before" (resp. "after") period. This model also integrates a vector of known non-random components denoted by $\mathbf{Z} = (z_1, \dots, z_r)^T$. It is assumed that $\mathbf{X}$ follows the multinomial distribution

$$\mathbf{X} \sim \mathcal{M}(n; \boldsymbol{\pi}_1(\boldsymbol{\beta}), \boldsymbol{\pi}_2(\boldsymbol{\beta}))$$

where $n$ is a positive integer representing the total number of independent accidents in both before and after periods. Here $\boldsymbol{\pi}_1(\boldsymbol{\beta}) = (\pi_{11}(\boldsymbol{\beta}), \dots, \pi_{1r}(\boldsymbol{\beta}))^T$, $\boldsymbol{\pi}_2(\boldsymbol{\beta}) = (\pi_{21}(\boldsymbol{\beta}), \dots, \pi_{2r}(\boldsymbol{\beta}))^T$ with

$$\pi_{ij}(\boldsymbol{\beta}) = \begin{cases} \dfrac{\phi_j}{1 + \theta \sum_{m=1}^r z_m \phi_m}, & i = 1; \quad j = 1, \dots, r, \\ \\ \dfrac{\theta z_j \phi_j}{1 + \theta \sum_{m=1}^r z_m \phi_m}, & i = 2; \quad j = 1, \dots, r \end{cases} \tag{3.1}$$

and

$$\sum_{i=1}^2 \sum_{j=1}^r \pi_{ij}(\boldsymbol{\beta}) = 1.$$

The random vector $\mathbf{X}$ has a probability density function depending on a multidimensional parameter $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi}^T)^T$ where $\theta \in \mathbb{R}_+^*$ and $\boldsymbol{\phi} = (\phi_1, \dots, \phi_r)^T$ belongs to the simplex of dimension $r - 1$ thus defined:

$$\mathbb{S}_{r-1} = \Big\{ (\phi_1, \dots, \phi_r) \in \mathbb{R}^r \mid \phi_j > 0, \quad 1 \leqslant i \leqslant r, \quad \sum_{j=1}^r \phi_j = 1 \Big\}.$$

The scalar $\theta$ represents the unknown average effect of the road safety measure while each $\phi_j$ $(j = 1, \dots, r)$ denotes the global accident risk of type $j$. The coefficients $z_1, \dots, z_r$ are given positive real numbers.

The existence of the constrained MLE $\hat{\boldsymbol{\beta}}_n$ of model (3.1) has been studied by N'Guessan [71] and an application is given in [78]. The numerical convergence properties of $\hat{\boldsymbol{\beta}}_n$ to the true values were recently studied by N'Guessan and Geraldo [76] using intensive simulation studies. They found that the MLE $\hat{\boldsymbol{\beta}}_n$ given by (2.22) converges numerically to the true value of the parameter whenever $n$ tends to $+\infty$.

The remainder of the chapter is organized as follows. In Section 3.2, we give an overview of the consistency of a MLE including basic definitions as well as the general results on this consistency. In Section 3.3, we give some preliminary results. The main results on the strong convergence of the MLE in crash control model are presented in Section 3.4. In Section 3.5, we give some numerical illustrations of our main results and we finish this chapter with some remarks.

## 3.2 Consistency of the MLE : an overview

**Definition 3.2.1.** Let $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n)$ be a sample of independent identically distributed (iid) random vectors according to a distribution depending on a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\mathbf{T}_n = \mathbf{T}_n(\mathbf{X})$ be an estimator of $\boldsymbol{\beta}$ for every $n$. The sequence $\mathbf{T}_n$ is said to be consistent (or weakly consistent) if, for every $\boldsymbol{\beta} \in \mathbb{S}$, $\mathbf{T}_n$ converges in probability to $\boldsymbol{\beta}$, that is

$$\forall \varepsilon > 0, \quad \lim_{n\to\infty} \mathrm{P}\left(\|\mathbf{T}_n - \boldsymbol{\beta}\| > \varepsilon\right) = 0 \tag{3.2}$$

or equivalently

$$\forall \varepsilon > 0, \quad \lim_{n\to\infty} \mathrm{P}\left(\|\mathbf{T}_n - \boldsymbol{\beta}\| < \varepsilon\right) = 1 \tag{3.3}$$

for any norm $\|\cdot\|$ on $\mathbb{R}^d$.

The notion of consistency is actually a concept relating to a sequence of estimators $(\mathbf{T}_n)_n$, but very often for simplicity reasons, one usually says "consistency of $\mathbf{T}_n$". It should be noted that the definition of consistency deals with an entire family of probability models indexed by $\boldsymbol{\beta}$. For each different value of $\boldsymbol{\beta}$, the probability model associated with the sequence $\mathbf{T}_n$ is different. And the definition says that for each value of $\boldsymbol{\beta}$, the probability model is such that the sequence converges in probability to the true value of $\boldsymbol{\beta}$.

The concept of consistency as defined in Definition 3.2.1 can be extended to other modes of convergence of random variables such as the almost sure convergence.

**Definition 3.2.2.** Let $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n)$ be a sample of independent identically distributed (iid) random vectors according to a distribution depending on a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\mathbf{T}_n = \mathbf{T}_n(\mathbf{X})$ be a point estimator of $\boldsymbol{\beta}$ for every $n$. The sequence $\mathbf{T}_n$ is said to be strongly consistent if, for every $\boldsymbol{\beta} \in \mathbb{S}$, $\mathbf{T}_n$ converges almost surely to $\boldsymbol{\beta}$, that is

$$\mathrm{P}\left(\lim_{n\to\infty} \mathbf{T}_n = \boldsymbol{\beta}\right) = 1. \tag{3.4}$$

Equivalently, the sequence $\mathbf{T}_n$ is strongly consistent if there exists a null set $N$ (i.e. $\mathrm{P}(N) = 0$) such that

$$\text{for every } \omega \in \Omega \setminus N, \quad \lim_{n\to\infty} \mathbf{T}_n(\omega) = \boldsymbol{\beta}. \tag{3.5}$$

Consistency is a very essential requirement in the sense that any inconsistent estimator should not be used. It has thus been the subject of many books and papers. To the best of our knowledge, even if the term "consistency" appears nowhere in his book, Cramer [14] is one of the very first authors to investigate the weak consistency of the MLE. If one considers the case of a single unknown parameter $\beta$, Cramer [14] proved that the likelihood equation

$$\frac{\partial \log f}{\partial \beta} = 0$$

has a solution which converges in probability to the true value of $\boldsymbol{\beta}$, as $n \to \infty$, if the following conditions are fulfilled:

(C1) for almost all $x$, the first, second and third order derivatives of $\log f$ exist for every $\beta$ belonging to an open (not necessarily finite) interval $\mathbb{A}$.

(C2) For every $\beta \in \mathbb{A}$, we have

$$\left|\frac{\partial \log f}{\partial \beta}\right| < F_1(x), \quad \left|\frac{\partial^2 \log f}{\partial \beta^2}\right| < F_2(x), \quad \left|\frac{\partial^3 \log f}{\partial \beta^3}\right| < F_3(x)$$

where the functions $F_1$ and $F_2$ are being integrable over $]-\infty, +\infty[$ while

$$\int_{-\infty}^{+\infty} F_3(x) f(x) dx \ < \ M$$

where $M$ is independent of $\beta$.

(C3) for every $\beta \in \mathbb{A}$, the integral

$$\int_{-\infty}^{+\infty} \Big(\frac{\partial \log f}{\partial \beta}\Big)^2 f(x) dx$$

is finite and positive.

**Remark 3.2.1.** Here are some interpretations of these conditions. Condition (C1) insures that the function $\partial \log f / \partial \beta$ has a Taylor expansion as a function of $\beta$. Condition (C2) allows differentiation with respect to $\beta$ under the integral sign and Condition (C3) states that the random variable $\partial \log f / \partial \beta$ has finite positive variance.

Cramer [14] notes that in the case of several unknown parameters, one has to introduce conditions which form a straightforward generalization of the conditions given in the case of a single parameter. Serfling [92] proved under the same regularity conditions as Cramer that the likelihood equations admit a sequence of solutions strongly consistent. However, as noted by Wald [102], the proof given by Cramer establishes the consistency of some root of the likelihood equations but does not necessarily establish the consistency of the MLE when the likelihood equations have several roots.

The main theorem on strong consistency of the MLE is due to Wald [102] who gave regularity conditions under which any statistic $\hat{\boldsymbol{\beta}}_n(x_1, \ldots, x_n)$ such that

$$\frac{f(x_1, \hat{\boldsymbol{\beta}}_n) \cdots f(x_n, \hat{\boldsymbol{\beta}}_n)}{f(x_1, \boldsymbol{\beta}^0) \cdots f(x_n, \boldsymbol{\beta}^0)} \geqslant c > 0 \tag{3.6}$$

also verifies

$$\mathrm{P}(\lim \hat{\boldsymbol{\beta}}_n = \boldsymbol{\beta}^0) = 1.$$

And he deduced the strong consistency of the MLE since it satisfies the relationship (3.6) with $c = 1$. The regularity conditions imposed by Wald are the following.

**Wald's conditions for strong consistency of the MLE [102]**

(W1) The cumulative distribution function $F(x, \boldsymbol{\beta})$ is either discrete for all $\boldsymbol{\beta}$ or absolutely continuous for all $\boldsymbol{\beta}$.

(W2) For $\epsilon$ sufficiently small and $a$ sufficiently large, the expected values

$$\int_{-\infty}^{+\infty} \log f^*(x, \boldsymbol{\beta}, \epsilon)\ dF(x, \boldsymbol{\beta}^0) \qquad \text{and} \qquad \int_{-\infty}^{+\infty} \log \varphi^*(x, a)\ dF(x, \boldsymbol{\beta}^0)$$

are finite with $\boldsymbol{\beta}^0$ the true value of $\boldsymbol{\beta}$ and

$$\begin{aligned} f(x, \boldsymbol{\beta}, \epsilon) &= \sup_{\boldsymbol{\beta}'/\|\boldsymbol{\beta}-\boldsymbol{\beta}'\| \leqslant \epsilon} f(x, \boldsymbol{\beta}') \\ \varphi(x, a) &= \sup_{\|\boldsymbol{\beta}\| > a} f(x, \boldsymbol{\beta}) \\ f^*(x, \boldsymbol{\beta}, \epsilon) &= \max(1, f(x, \boldsymbol{\beta}, \epsilon)) \\ \varphi^*(x, a) &= \max(1, \varphi(x, a)). \end{aligned}$$

(W3) The mapping $\boldsymbol{\beta} \mapsto f(x, \boldsymbol{\beta})$ is continuous.

(W4) The model is identifiable i.e. if $\boldsymbol{\beta} \neq \boldsymbol{\beta}'$ then $F(x, \boldsymbol{\beta}) \neq F(x, \boldsymbol{\beta}')$ for at least one value of $x$.

(W5) If $\|\boldsymbol{\beta}\|$ tends to infinity then $f(x, \boldsymbol{\beta})$ tends to 0.

(W6) The true vector parameter $\boldsymbol{\beta}^0$ verifies the relationship

$$\int_{-\infty}^{+\infty} |\log f(x, \boldsymbol{\beta}^0)|\ dF(x, \boldsymbol{\beta}^0)\ <\ \infty.$$

(W7) The parameter space is closed.

(W8) The function $f(x, \boldsymbol{\beta}, \epsilon)$ is a measurable function of $x$ for any $\boldsymbol{\beta}$ and $\epsilon$.

**Remark 3.2.2.** Even if it is claimed in [102] that the underlying assumptions are easy to check in practice, it must be noted that some of the assumptions involve the calculation of complex integrals over domains included in $\mathbb{R}^d$ with $d$ the number of parameters.

Fiorin [24] proposed another proof of the consistency of the MLE under slight modifications of Wald's conditions. His consistency result states that, with probability one, and when the number of observations is big enough, the likelihood function always admits at least one global maximizer and that, at the same time, all the possible global maximizers are strongly consistent estimates for the unknown parameter.

Van der Vaart [99] studies the weak consistency of $M-$estimators of which maximum likelihood estimators are a special case. A $M-$estimator is any estimator maximizing a function of the type

$$\boldsymbol{\beta} \longmapsto M_n(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} m_{\boldsymbol{\beta}}(X_i)$$

where $m_{\boldsymbol{\beta}}$ is a known function. The name $M-$estimator is also used for estimators satisfying systems of equations of the type

$$\Psi(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \psi_{\boldsymbol{\beta}}(X_i) = 0$$

where $\psi_{\boldsymbol{\beta}}$ is a known vector-valued mapping. Strong consistency of $M-$estimators was studied by Chafai and Concordet [8]. Seo and Lindsay [91] give some general results on the consistency of MLEs using the concept of finite entropy. They consider the more general problem of estimating

$$\hat{m}_n = \underset{m\in\mathbb{M}}{\operatorname{argmax}} \sum_{i=1}^{n} \log m(X_i)$$

where $\mathbb{M}$ is a set of probability measures. They noted that the consistency of the MLE for parametric case can be constructed by adding a series of regularity conditions.

**Remark 3.2.3.** For probability models having a complex log-likelihood like ours, some of the assumptions are not very easy to check since they imply calculation of very complicated mathematical expectations. Moreover, the presence of constraints on the parameters may further complicate the situation and may possibly require taking into account Lagrange multipliers when dealing with the likelihood equations. Another fact that must not be forgotten is that the MLE can be inconsistent. This is shown by some examples given in papers like [3, 23, 39, 48]. A disturbing example is given in [39], in which Cramer's conditions are fulfilled, the MLE exists, is unique but is not consistent. In [23], an example is given with parameter space $[0, 1]$, where the MLE is well defined but almost always converges to 1, no matter which parameter is used to generate the data.

## 3.3 Preliminary results

Throughout the sequel, the subscript $n$ is used to indicate that the estimators depend on the sample size $n$. It is proven in [73] that the log-likelihood associated to an observed data $\mathbf{x} = (x_{11}, \ldots, x_{1r}, x_{21}, \ldots, x_{2r})$ satisfying

$$\sum_{i=1}^{2}\sum_{j=1}^{r} x_{ij} = n$$

is defined up to an additive constant by

$$\ell(\boldsymbol{\beta}) = \sum_{j=1}^{r}\left( x_{+j}\log(\phi_j) + x_{2j}\log(\theta) - x_{+j}\log(1+\theta\sum_{m=1}^{r} z_m\phi_m)\right)$$

where $x_{+j} = x_{1j} + x_{2j}$, $j = 1, \ldots, r$. The MLE $\hat{\boldsymbol{\beta}}_n$ of $\boldsymbol{\beta}$ is then given by the following lemma.

**Lemma 3.3.1** ([71]). *The components $\hat{\theta}_n$ and $\hat{\phi}_n$ of the MLE $\hat{\boldsymbol{\beta}}_n$ satisfy*

$$\begin{cases} \hat{\theta}_n = \dfrac{\sum_{m=1}^r X_{2m}}{\left(\sum_{m=1}^r z_m \hat{\phi}_{n,m}\right) \times \left(\sum_{m=1}^r X_{1m}\right)} \\[3em] \hat{\phi}_{n,j} = \dfrac{1}{1 - \frac{1}{n} \displaystyle\sum_{m=1}^r \dfrac{\hat{\theta}_n z_m X_{+m}}{1 + \hat{\theta}_n z_m}} \times \dfrac{X_{+j}}{n(1+\hat{\theta}_n z_j)}, \qquad j = 1, \dots, r \end{cases} \tag{3.7}$$

*with $X_{+j} = X_{1j} + X_{2j}$, $j = 1, \dots, r$.*

Let us recall some important lemmas that will be the key for establishing our strong convergence results. The first lemma is provided by the continuous mapping theorem of [99, p. 7]. The second is due to the strong law of large numbers (SLLN).

**Lemma 3.3.2** ([99]). *Let $g : \mathbb{R}^k \to \mathbb{R}^m$ be continuous at every point of a set $A$ such that $\mathrm{P}(X \in A) = 1$. If $X_n$ converges almost surely (a.s.) to $X$ then $g(X_n)$ converges almost surely to $g(X)$.*

**Lemma 3.3.3.** *If the random vector $\mathbf{X} = (X_{11}, \dots, X_{1r}, X_{21}, \dots, X_{2r})$ has the multinomial distribution $\mathcal{M}(n; \boldsymbol{\pi})$ with $\boldsymbol{\pi} = (\pi_{11}, \dots, \pi_{1r}, \pi_{21}, \dots, \pi_{2r})$ then, as $n \to +\infty$:*

$$\frac{1}{n}\mathbf{X} \quad \xrightarrow{a.s.} \quad (\pi_{11}, \dots, \pi_{1r}, \pi_{21}, \dots, \pi_{2r}).$$

**Proof**. The proof is inspired from [101]. For the sake of notation, the components of the vectors $\mathbf{X}$ and $\boldsymbol{\pi}$ will be denoted respectively $\mathbf{X} = (X_1, \dots, X_d)$ and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_d)$. We begin by noting that the random vector $\mathbf{X} = (X_1, \dots, X_d)$ can be linked to a random experiment that consists in distributing $n$ objects into $d$ mutually exclusive classes such that each object has the same probability $\pi_k$ to be assigned to the class $k$. The random variable $X_k$ then represents the total number of objects assigned to the class $k$. The marginal distribution of $X_k$ is used to determine its expected value. The assignment of the object $i$ ($i = 1, \dots, n$) into the $d$ classes can also be considered as Bernoulli trial with two possible outcomes that are : success (if the object $i$ is assigned to the class $k$) with probability $\pi_k$ and failure (if the object $i$ is assigned to a class different from $k$) with probability $1 - \pi_k$. The random variable $X_k$ thus takes its values in the set $\{0, 1, \dots, n\}$ and represents the number of successes from $n$ independent trials of a Bernoulli trial. Its follows the binomial distribution $\mathcal{B}(n, \pi_k)$ hence $\mathrm{E}(X_k) = n\pi_k$.

To obtain the almost sure convergence result, we write the vector $\mathbf{X}$ as

$$\mathbf{X} = \sum_{i=1}^n \mathbf{Y}_i$$

where $\mathbf{Y}_i = (Y_{i1}, \dots, Y_{id})$,

$$Y_{ij} = \begin{cases} 1 & \text{if the object } i \text{ is assigned to the class } j \\ 0 & \text{otherwise.} \end{cases}$$

The random vectors $\mathbf{Y}_i$ are independent identically distributed from the multinomial distribution $\mathcal{M}(1, \boldsymbol{\pi})$. By the SLLN, the sequence of random vectors $\mathbf{Y}/n$ converges a.s. to $\mathrm{E}(\mathbf{Y}_1) = \boldsymbol{\pi}$. $\square$

The remaining lemmas are related to the uniform convergence of sequences of functions on a metric space. The first one [90, p. 148, Theorem 7.9] gives conditions under which convergence of a sequence of functions implies uniform convergence. The second [4, Theorem 2] states conditions under which convergence of a sequence of injective functions $(f_n)$ implies that of their inverses $(f_n^{-1})$.

**Lemma 3.3.4** ([90]). *Let $f_1, \dots, f_n$ be a sequence of functions on a set $E$ and $f$ a function on $E$. Let us suppose that*

$$\lim_{n\to\infty} f_n(x) = f(x), \quad \forall x \in E.$$

*Then, the sequence $(f_n)$ converges uniformly to $f$ on $E$ if and only if*

$$\sup_{x\in E} |f_n(x) - f(x)| \to 0 \quad as \quad n \to \infty.$$

**Lemma 3.3.5** ([4]). *If $(f_n)$ is a sequence of injection mappings on a metric space $E$, taking values in a locally compact metric space $G$ and converging uniformly to $f$ on $E$ and if $f^{-1}$ is a continuous mapping on $G_1 \subset G$, then $f_n^{-1}$ converges uniformly to $f^{-1}$ on every compact set contained in $\mathrm{int}(G_1) \cap (\cap_n f_n(E))$ where $\mathrm{int}(G_1)$ denotes the interior of $G_1$.*

**Lemma 3.3.6** ([96]). *Let $f_n$ be a sequence of continuous functions on a set $D$. If $f_n$ converges uniformly to $f$, then $f_n(u_n)$ converges to $f(u)$ for all sequences $u_n$ in $D$ convergent to $u \in D$.*

Our main results are presented in the next section.

## 3.4 Main results

We first establish a cyclic type convergence.

**Theorem 3.4.1.** *As $n$ tends to $+\infty$, the random variable $\hat{\theta}_n$ converges a.s. to $\theta^0$ if and only if the random vector $\hat{\boldsymbol{\phi}}_n$ converges a.s. to $\boldsymbol{\phi}^0$.*

**Proof**. We know that (see [92]) $\hat{\boldsymbol{\phi}}_n = (\hat{\phi}_{n,1}, \dots, \hat{\phi}_{n,r}) \in \mathbb{R}^r$ converges a.s. to $\boldsymbol{\phi}^0 = (\phi_1^0, \dots, \phi_r^0) \in \mathbb{R}^r$ if and only if, for all $j = 1, \dots, r$, $\hat{\phi}_{n,j} \to \phi_j^0$ a.s. Thus, it is sufficient to prove that for all $j = 1, \dots, r$,

$$\hat{\theta}_n \longrightarrow \theta^0 \ a.s. \text{ implies that } \hat{\phi}_{n,j} \longrightarrow \phi_r^0 \ a.s.$$

Now let us suppose that $\hat{\theta}_n \to \theta^0$ a.s. Observing that $\sum_{m=1}^r X_{+m} = n$, we get

$$\hat{\phi}_{n,j} = \frac{(X_{1j} + X_{2j})/(1 + \hat{\theta}_n z_j)}{\sum_{m=1}^r (X_{1m} + X_{2m})/(1 + \hat{\theta}_n z_m)}. \tag{3.8}$$

Moreover we can write

$$\hat{\phi}_{n,j} = g_j\left(\frac{X_{11}}{n}, \ldots, \frac{X_{1r}}{n}, \frac{X_{21}}{n}, \ldots, \frac{X_{2r}}{n}, \hat{\theta}_n\right)$$

where $g_j$ is the continuous function from $\mathbb{R}^{2r+1}$ to $\mathbb{R}$ defined by

$$g_j(b_1, \ldots, b_r, a_1, \ldots, a_r, \theta) \quad = \quad \frac{(b_j + a_j)/(1 + \theta z_j)}{\sum_{m=1}^{r}(b_m + a_m)/(1 + \theta z_m)}.$$

Using Lemma 3.3.3, we have almost surely

$$\left(\frac{X_{11}}{n}, \ldots, \frac{X_{1r}}{n}, \frac{X_{21}}{n}, \ldots, \frac{X_{2r}}{n}, \hat{\theta}_n\right) \to (\pi_{11}^0, \ldots, \pi_{1r}^0, \pi_{21}^0, \ldots, \pi_{2r}^0, \theta^0)$$

as $n \to \infty$. Applying the continuous mapping theorem (Lemma 3.3.2) and relations (3.1), we get as $n \to +\infty$,

$$\hat{\phi}_{n,j} \quad \to \quad \frac{\left(\pi_{1j}^0 + \pi_{2j}^0\right)/(1 + \theta^0 z_j)}{\sum_{m=1}^{r}\left(\pi_{1m}^0 + \pi_{2m}^0\right)/(1 + \theta^0 z_m)} = \phi_j^0 \quad \text{a.s.}$$

This proves that $\hat{\phi}_{n,j}$ converges to $\phi_j^0$ a.s.

Now let us assume that $\hat{\boldsymbol{\phi}}_n \longrightarrow \boldsymbol{\phi}^0$ a.s. or equivalently $\hat{\phi}_{n,j} \longrightarrow \phi_r^0$ a.s. for all $j = 1, \ldots, r$. From Lemma 3.3.1, we get

$$\begin{aligned}\hat{\theta}_n &= \frac{\sum_{m=1}^{r}(X_{2m}/n)}{\sum_{m=1}^{r}(X_{1m}/n)} \times \frac{1}{\sum_{m=1}^{r} z_m\, \hat{\phi}_{n,m}} \\ &= g\left(\frac{X_{11}}{n}, \ldots, \frac{X_{1r}}{n}, \frac{X_{21}}{n}, \ldots, \frac{X_{2r}}{n}, \hat{\phi}_{n,1}, \ldots, \hat{\phi}_{n,r}\right)\end{aligned}$$

where $g$ is the continuous function defined from $\mathbb{R}^{3r}$ to $\mathbb{R}$ by

$$g(b_1, \ldots, b_r, a_1, \ldots, a_r, \phi_1, \ldots, \phi_r) \quad = \quad \frac{\sum_{m=1}^{r} a_m}{\sum_{m=1}^{r} b_m} \times \frac{1}{\sum_{m=1}^{r} z_m\, \phi_m}.$$

We again apply Lemmas 3.3.2 and 3.3.3 and get

$$\hat{\theta}_n \quad \xrightarrow[n \to +\infty]{a.s.} \quad g(\pi_{11}^0, \ldots \pi_{1r}^0, \pi_{21}^0, \ldots, \pi_{2r}^0, \phi_1^0, \ldots, \phi_r^0) = \theta^0.$$

□

Theorem 3.4.1 shows that the almost sure convergence of $\hat{\boldsymbol{\phi}}_n$ to $\boldsymbol{\phi}^0$ is equivalent to the almost sure convergence of $\hat{\theta}_n$ to $\theta^0$. To prove that the MLE $\hat{\boldsymbol{\beta}}_n = (\hat{\theta}_n, \hat{\boldsymbol{\phi}}_n^T)^T$ converges almost surely, it is then sufficient by Theorem 3.4.1 to prove for example that $\hat{\theta}_n$ converges almost surely to $\theta^0$. With that in mind, we first prove that the a.s. limit of $\hat{\theta}_n$ exists and then show that this a.s. limit is equal to $\theta^0$.

The following result shows the almost sure convergence of $\hat{\theta}_n$.

**Theorem 3.4.2.** *There exists a constant* $\mu > 0$ *and a subset* $N \subset \Omega$ *such that* $\mathrm{P}(N) = 0$ *and*

$$\forall \omega \in \Omega \setminus N, \quad \lim_{n\to\infty} \hat{\theta}_n(\omega) = \mu. \tag{3.9}$$

**Proof**. Set

$$\varphi_n(u) = \sum_{m=1}^{r} \frac{X_{+m}/n}{1+uz_m}, \; u \in ]0,+\infty[$$

and for all $i = 1, 2$, denote $X_{i+} = \sum_{j=1}^{r} X_{ij}$.

We first show that for all $\omega \in \Omega$, the real valued function

$$\varphi_{\omega,n}(u) = \varphi_n(u)(\omega) = \sum_{m=1}^{r} \frac{X_{+m}(\omega)/n}{1+uz_m}$$

is a continuous bijective mapping from $]0,+\infty[$ to $]0,1[$ and that $\hat{\theta}_n(\omega) = \varphi_{\omega,n}^{-1}(X_{1+}(\omega)/n)$. By Equation (3.8), we have the relationship

$$\hat{\phi}_{n,j} = \frac{X_{+j}/(1+\hat{\theta}_n z_j)}{\sum_{m=1}^{r} X_{+m}/(1+\hat{\theta}_n z_m)}$$

which enables to write

$$\sum_{j=1}^{r} z_j \hat{\phi}_{n,j} = \frac{\sum_{j=1}^{r} \Big( z_j X_{+j}/(1+\hat{\theta}_n z_j) \Big)}{\sum_{m=1}^{r} \Big( X_{+m}/(1+\hat{\theta}_n z_m) \Big)}.$$

By the first line of System (2.22) in Lemma 3.3.1, we have

$$\begin{aligned} \hat{\theta}_n &= \frac{X_{2+}}{X_{1+}} \frac{1}{\sum_{j=1}^{r} z_j \hat{\phi}_{n,j}} \\ &= \frac{X_{2+}}{X_{1+}} \frac{\sum_{m=1}^{r} \Big( X_{+m}/(1+\hat{\theta}_n z_m) \Big)}{\sum_{j=1}^{r} \Big( z_j X_{+j}/(1+\hat{\theta}_n z_j) \Big)}. \end{aligned}$$

This is equivalent to

$$\frac{X_{2+}}{X_{1+}} \sum_{m=1}^{r} \frac{X_{+m}}{1+\hat{\theta}_n z_m} = \sum_{m=1}^{r} \frac{\hat{\theta}_n z_m X_{+m}}{1+\hat{\theta}_n z_m}.$$

We then deduce that

$$\sum_{m=1}^{r} \frac{X_{+m}}{1+\hat{\theta}_n z_m} = X_{1+}.$$

Divide the last equality by the sample size $n$ and get

$$\sum_{m=1}^{r} \frac{X_{+m}(\omega)/n}{1+\hat{\theta}_n(\omega) z_m} = \frac{X_{1+}(\omega)}{n}, \quad \forall \omega \in \Omega. \tag{3.10}$$

It is obvious that the random real function $\varphi_{\omega,n}(u)$ has a strictly negative derivative with respect to $u$ and satisfies for all $\omega \in \Omega$:

$$1 = \lim_{u\to 0} \varphi_{\omega,n}(u) = \sum_{m=1}^{r} X_{+m}(\omega)/n$$
$$0 = \lim_{u\to +\infty} \varphi_{\omega,n}(u).$$

Hence $\varphi_{\omega,n}(u)$ is a continuous and bijective mapping from $]0,+\infty[$ to $]0,1[$ and since $X_{1+}(w)/n \in ]0,1[$, Equation (3.10) yields

$$\hat{\theta}_n(\omega) = \varphi_{\omega,n}^{-1}\left(\frac{X_{1+}(\omega)}{n}\right).$$

Let us now prove that there exists a subset $N \subset \Omega$ with $\mathrm{P}(N) = 0$ such that for all $\omega \in \Omega \setminus N$, the sequence of real functions $\varphi_{\omega,n}(u)$ converges uniformly to some function $\varphi(u)$ on $]0,+\infty[$. The almost sure convergence of the statistic

$$\varphi_n(u) = \sum_{m=1}^{R} \frac{X_{+m}/n}{1+uz_m}$$

to $\varphi(u)$ will then follow.

For all $m = 1,\dots,r$, write

$$\frac{X_{+m}}{n} = g_m\left(\frac{X_{11}}{n},\dots,\frac{X_{1r}}{n},\frac{X_{21}}{n},\dots,\frac{X_{2r}}{n}\right)$$

where $g_m$ is the continuous mapping defined from $\mathbb{R}^{2r}$ to $\mathbb{R}$ by

$$g_m(b_1,\dots,b_r,a_1,\dots,a_r) = b_m + a_m.$$

Applying Lemmas 3.3.2 and 3.3.3, we get

$$\frac{X_{+m}}{n} \xrightarrow{a.s.} \alpha_m^0 = g_m(\pi_{11}^0,\dots,\pi_{1r}^0,\pi_{21}^0,\dots,\pi_{2r}^0) = \frac{(1+\theta^0 z_m)\phi_m^0}{1+\theta^0\sum_{k=1}^r z_k\phi_k^0}.$$

Equivalently [12, p. 68], there exists a null set $N_m$ such that

$$\forall\omega \in \Omega\setminus N_m, \quad \lim_{n\to\infty}\frac{X_{+m}(\omega)}{n} = \alpha_m^0. \tag{3.11}$$

The set $E_1 = \bigcup_{m=1}^r N_m$ satisfies $\mathrm{P}(E_1) = 0$ and

$$\forall\omega \in \Omega\setminus E_1, \quad \lim_{n\to\infty}\varphi_{\omega,n}(u) = \lim_{n\to\infty}\sum_{m=1}^{r}\frac{X_{+m}(\omega)/n}{1+uz_m} = \sum_{m=1}^{r}\frac{\alpha_m^0}{1+uz_m} = \varphi(u). \tag{3.12}$$

Thus we have proved that for all $\omega \in \Omega \setminus E_1$, the sequence of functions $\varphi_{\omega,n}$ converges simply to $\varphi$ on $]0,+\infty[$. Moreover,

$$\begin{aligned}
\sup_{u\in]0,+\infty[} \left|\varphi_{\omega,n}(u)-\varphi(u)\right| &= \sup_{u\in]0,+\infty[} \left|\sum_{m=1}^{r} \frac{X_{+m}(\omega)/n}{1+uz_m} - \sum_{m=1}^{r} \frac{\alpha_m^0}{1+uz_m}\right| \\
&\leqslant \sup_{u\in]0;+\infty[} \sum_{m=1}^{r} \left|\frac{X_{+m}(\omega)/n-\alpha_m^0}{1+uz_m}\right| \\
&\leqslant \sup_{u\in]0,+\infty[} \sum_{m=1}^{r} \frac{|X_{+m}(\omega)/n-\alpha_m^0|}{1+uz_m} \\
&\leqslant \sum_{m=1}^{r} \left|\frac{X_{+m}(\omega)}{n}-\alpha_m^0\right|
\end{aligned}$$

because

$$\forall u\in]0,+\infty[, \quad \forall z_m>0, \quad \frac{1}{1+uz_m} \leqslant 1.$$

It follows by (3.11) that

$$\sup_{u\in]0,+\infty[} \left|\varphi_{\omega,n}(u)-\varphi(u)\right| \leqslant \sum_{m=1}^{r} \left|\frac{X_{+m}(\omega)}{n}-\alpha_m^0\right| \longrightarrow 0 \text{ as } n\to+\infty.$$

This proves the uniform convergence of the sequence $\varphi_{\omega,n}$ to $\varphi$ on $]0,+\infty[$.

In summary we have proved that $\varphi_{\omega,n}$ is a sequence of bijective functions taking values in $]0,1[$ that is locally compact. Moreover, $\varphi_{\omega,n}$ converges uniformly to $\varphi$ and $\varphi^{-1}$ is continuous (as the inverse of a non zero continuous function). Thus the conditions of Lemma 3.3.5 are satisfied and hence the sequence $\varphi_{\omega,n}^{-1}$ converges uniformly to $\varphi^{-1}$. Moreover, the sequence $X_{1+}/n$ satisfies

$$\frac{X_{1+}}{n} = \tilde{g}\left(\frac{X_{11}}{n},\ldots,\frac{X_{1r}}{n},\frac{X_{21}}{n},\ldots,\frac{X_{2r}}{n}\right)$$

where $\tilde{g}$ is the continuous mapping defined from $\mathbb{R}^{2r}$ to $\mathbb{R}$ by

$$\tilde{g}(b_1,\ldots,b_r,a_1,\ldots,a_r) = \sum_{m=1}^{r} b_m.$$

Apply again Lemmas 3.3.2 and 3.3.3 and get

$$\frac{X_{1+}}{n} \xrightarrow{a.s.} \tilde{g}(\pi_{11}^0,\ldots,\pi_{1r}^0,\pi_{21}^0,\ldots,\pi_{2r}^0) = \frac{1}{1+\theta^0\sum_{k=1}^{r} z_k\phi_k^0}.$$

That is, there exists a null set $E_2$ such that

$$\forall\omega\in\Omega\setminus E_2,\ \lim_{n\to\infty}\frac{X_{1+}(\omega)}{n} = \gamma^0 = \frac{1}{1+\theta^0\sum_{k=1}^{r} z_k\phi_k^0}.$$

The set $N=E_1\cup E_2$ satisfies $\mathrm{P}(N)=0$ and for all $\omega\in\Omega\setminus N$, the sequence $X_{1+}(\omega)/n$ is convergent and hence is bounded. That is, there exists a compact set $D\subset]0,1[$ such that $X_{1+}(\omega)/n\in D, \forall n>0$. Moreover, since

$$\hat{\theta}_n(\omega) = \varphi_{\omega,n}^{-1}\left(\frac{X_{1+}(\omega)}{n}\right)$$

and $\varphi_{\omega,n}^{-1}$ converges uniformly to $\varphi$, we apply Lemma 3.3.6 to conclude that

$$\forall\omega \in \Omega \setminus N,\ \hat{\theta}_n(\omega) \longrightarrow \mu = \varphi^{-1}(\gamma^0) \text{ as } n \to \infty \tag{3.13}$$

where $\gamma^0$ is given above. This ends the proof of Theorem 3.4.2. □

**Theorem 3.4.3.** *Set $\beta^0 = \sum_{j=1}^r z_j\phi_j^0$. Let $F_{\theta^0}$ be the function from $\mathbb{R}_+$ to $\mathbb{R}$ defined by:*

$$F_{\theta^0}(u) = u\left(\sum_{m=1}^r \frac{z_m(1+\theta^0 z_m)\phi_m^0}{1+uz_m}\right) - \theta^0\beta^0\left(\sum_{m=1}^r \frac{(1+\theta^0 z_m)\phi_m^0}{1+uz_m}\right).$$

*Then,*

*i) the function $F_{\theta^0}$ has $\theta^0$ as unique root on $\mathbb{R}_+$.*

*ii) the almost sure limit $\mu$ of $\hat{\theta}_n$ is equal to the unique root $\theta^0$ of $F_{\theta^0}$ on $\mathbb{R}_+$.*

**Proof**. i) We have $F_{\theta^0}(\theta^0) = 0$ and

$$\begin{aligned} F'_{\theta^0}(u) &= \sum_m \frac{c_m(1+\theta z_m)\phi_m}{1+uz_m} - u\sum_m \frac{z_m^2(1+\theta z_m)\phi_m}{(1+uz_m)^2} + \theta\beta\sum_m \frac{z_m(1+\theta z_m)\phi_m}{(1+uz_m)^2} \\ &= \sum_m \frac{(1+\theta z_m)z_m\phi_m(1+\theta\beta)}{(1+uz_m)^2} > 0. \end{aligned}$$

So $F_{\theta^0}$ is strictly monotone and satisfies

$$\lim_{u\to 0} F_{\theta^0}(u) < 0 \quad \text{and} \quad \lim_{u\to+\infty} F_{\theta^0}(u) > 0.$$

So we get the first assertion of the theorem.

ii) Let us assume that as $n \to +\infty$, $\hat{\theta}_n \to \mu$ a.s. Let us then divide the numerator and the denominator of $\hat{\theta}_n$ given by Lemma 3.3.1 by $n^2$ and get

$$\hat{\theta}_n = \frac{\sum_{m=1}^r (X_{2m}/n) \ \times\ \sum_{m=1}^r \frac{X_{+m}/n}{1+\hat{\theta}_n z_m}}{\sum_{m=1}^r (X_{1m}/n) \ \times\ \sum_{m=1}^r \frac{z_m X_{+m}/n}{1+\hat{\theta}_n z_m}}. \tag{3.14}$$

By Lemma 3.3.3,

$$\frac{\sum_{m=1}^r X_{2m}/n}{\sum_{m=1}^r X_{1m}/n} \quad \xrightarrow[n\to+\infty]{a.s.} \quad \frac{\sum_{m=1}^r \pi_{2m}^0}{\sum_{m=1}^r \pi_{1m}^0} = \theta^0\beta^0$$

and

$$\frac{X_{+j}/n}{1+\hat{\theta}_n z_j} = \frac{X_{1j}/n + X_{2j}/n}{1+\hat{\theta}_n z_j} \quad \xrightarrow[n\to+\infty]{a.s.} \quad \frac{\pi_{1j}^0 + \pi_{2j}^0}{1+\mu z_j} = \frac{(1+\theta^0 z_j)\phi_j^0}{(1+\theta^0\sum_{m=1}^r z_m\phi_m^0)(1+\mu z_j)}.$$

Thus, as $n \to +\infty$, the first and the second hands of equation (3.14) yield, almost surely,

$$\mu = \theta^0\beta^0 \times \left(\sum_{m=1}^r \frac{(1+\theta^0 z_m)\phi_m^0}{(1+\mu z_m)}\right) \Big/ \left(\sum_{m=1}^r \frac{z_m(1+\theta^0 z_m)\phi_m^0}{(1+\mu z_m)}\right).$$

That is,

$$\mu \sum_{m=1}^{r} \frac{z_m(1+\theta^0 z_m)\phi_m^0}{(1+\mu z_m)} = \theta^0\beta^0 \times \sum_{m=1}^{r} \frac{(1+\theta^0 z_m)\phi_m^0}{(1+\mu z_m)},$$

which means that $F_{\theta^0}(\mu) = 0$ and hence $\mu = \theta^0$ by i). This completes the proof of Theorem 3.4.3. □

**Theorem 3.4.4.** *The MLE $\hat{\boldsymbol{\beta}}_n = (\hat{\theta}_n, \hat{\boldsymbol{\phi}}_n)^T$ converges a.s. to $\boldsymbol{\beta}^0 = (\theta^0, \boldsymbol{\phi}^0)^T$ with $\boldsymbol{\phi}^0 = (\phi_1^0, \ldots, \phi_r^0)^T$.*

**Proof**. This is a consequence of Theorem 3.4.1 and Theorem 3.4.3. Indeed, $\hat{\theta}_n$ converges a.s. to $\theta^0$ and since by Theorem 3.4.1, the consistency of $\hat{\theta}_n$ is equivalent to that of $\hat{\boldsymbol{\phi}}_n$, then $\hat{\boldsymbol{\phi}}_n$ converges also almost surely to $\boldsymbol{\phi}^0$. Thus the vector $\hat{\boldsymbol{\beta}}_n = (\hat{\theta}_n, \hat{\boldsymbol{\phi}}_n)^T$ converges a.s. to the vector $\boldsymbol{\beta}^0 = (\theta^0, \boldsymbol{\phi}^0)^T$. □

## 3.5 Numerical illustrations

### 3.5.1 Data generation

For given values of $r$ (the number of accident categories) and $n$ (the total number of accidents observed), the vector of accidents counts $\mathbf{x} = (x_{11}, \ldots, x_{1r}, x_{21}, \ldots, x_{2r})$ is generated as follows. Knowing the true values of the vector parameter $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi}^T)^T$ denoted

$$\boldsymbol{\beta}^0 = (\theta^0, (\boldsymbol{\phi}^0)^T)^T = (\theta^0, \phi_1^0, \ldots, \phi_r^0)^T$$

and the vector $\mathbf{Z} = (z_1, \ldots, z_r)^T$, we compute the true values $\boldsymbol{\pi}_1(\boldsymbol{\beta}^0)$, $\boldsymbol{\pi}_2(\boldsymbol{\beta}^0)$ and finally, data $\mathbf{x}$ is randomly generated from the multinomial distribution $\mathcal{M}(n; \boldsymbol{\pi}_1(\boldsymbol{\beta}^0), \boldsymbol{\pi}_2(\boldsymbol{\beta}^0))$.

### 3.5.2 Illustration of Theorem 3.4.1

We remind that a sequence of random vectors $(\mathbf{Y}_n)_n$ defined on a probability space $(\Omega, \mathcal{A}, \mathrm{P})$ converges almost surely (a.s.) to a constant vector $\mathbf{a}$ if

$$\mathrm{P}\left(\lim_{n\to\infty} \mathbf{Y}_n = \mathbf{a}\right) = 1 \qquad (3.15)$$

or equivalently if

$$\mathrm{P}\left(\left\{\omega \in \Omega \mid \forall\varepsilon > 0, \exists n_0 > 0, n \geqslant n_0 \Longrightarrow \|\mathbf{Y}_n(\omega) - \mathbf{a}\| < \varepsilon\right\}\right) = 1.$$

We propose to illustrate Theorem 3.4.1 in two steps. In the first step, we generate the estimator $\hat{\theta}_n$ so that it converges a.s. to $\theta^0$ (the true value of $\theta$) and we verify that the vector $\hat{\boldsymbol{\phi}}_n$ converges a.s. to $\boldsymbol{\phi}^0$. In the second and last step, we generate the vector $\hat{\boldsymbol{\phi}}_n$ so that it converges a.s. to $\boldsymbol{\phi}^0$ and then we verify that $\hat{\theta}_n$ converges a.s. to $\theta^0$.

In the first step, to ensure the a.s. convergence of $\hat{\theta}_n$ to $\theta^0$, we can generate $\hat{\theta}_n$ as

$$\hat{\theta}_n = \frac{1}{n}\sum_{i}^{n} Y_i$$

where the sequence $Y_1, \ldots, Y_n$ is composed of random variables independent identically distributed from a distribution having $\theta^0$ as expected value. Then, by the SLLN we will have the a.s. convergence p.s. of $\hat{\theta}_n$ to $\mathrm{E}(Y_1) = \theta^0$.

In order to diversify the simulations, three different scenarios for generating the random variables $Y_i$ have been considered:

– Scenario 1: $Y_i$ is a constant equal to $\theta^0$ i.e. $Y_i = \theta^0$.

– Scenario 2: $Y_i \sim \mathcal{U}(a, b)$ with $a = 0$ and $b = 2\theta^0$.

– Scenario 3: $Y_i \sim \mathcal{N}(\mu, \sigma)$ with $\mu = \theta^0$ and $\sigma = 0.2$. In this latter case, the standard deviation has been adjusted in order to ensure that $\hat{\theta}_n > 0$.

The results presented below correspond to values

$$\theta^0 = 0.5, \qquad \boldsymbol{\phi}^0 = (0.15, 0.25, 0.6)^T.$$

In order to verify the asymptotic behaviour of the different random variables involved in our simulation study, the values of the sample size $n$ are varied from $10^3$ to $10^5$ by an increment of $\Delta_n = 10^3$.

Figure 3.1 allows to check that for each of the three scenarios considered in this first step, the norm $|\hat{\theta}_n - \theta^0|$ tends to 0 when the sample size $n$ increases and tends to $10^5$. The values corresponding to this figure are given in Table 3.1. By varying $n$ from $10^3$ to $10^5$ by an increment of $\Delta_n = 10^3$, we get one hundred values for $n$. In order to summarize the results and save space, this table shows the evolution of the order of the error $|\hat{\theta}_n - \theta^0|^2$. The term "order" is used to refer to the closest power of 10.

Table 3.1. Evolution of the order of the error $|\hat{\theta}_n - \theta^0|^2$ when $n$ varies from $10^3$ to $10^5$.

| Values of $n$ | Scenario 1 | Scenario 2 | Scenario 3 |
|---|---|---|---|
| $10^3$ | 0 | $10^{-4}$ | $10^{-5}$ |
| $[2 \times 10^3, 7 \times 10^3]$ | - | $10^{-5}$ | $10^{-5}$ |
| $[8 \times 10^3, 1.7 \times 10^4]$ | - | $10^{-5}$ | $10^{-6}$ |
| $[1.8 \times 10^4, 8.4 \times 10^4]$ | - | $10^{-6}$ | $10^{-6}$ |
| $[8.5 \times 10^4, 10^5]$ | - | $10^{-6}$ | $10^{-7}$ |

Let us now see how the vector $\hat{\boldsymbol{\phi}}_n - \boldsymbol{\phi}^0$ behaves when $n$ increases and tends to $10^5$. We remind that components of $\hat{\boldsymbol{\phi}}_n$ are directly calculated from $\hat{\theta}_n$ using the relation

$$\hat{\phi}_{n,j} = \hat{\phi}_{n,j}(\hat{\theta}_n) = \frac{X_{+j}}{1 + \hat{\theta}_n z_j} \Big/ \left( \sum_{k=1}^{r} \frac{X_{+k}}{1 + \hat{\theta}_n z_k} \right)$$

where $X_{+j} = X_{1j} + X_{2j}$. The evolution of $\|\hat{\boldsymbol{\phi}}_n - \boldsymbol{\phi}^0\|_2^2$ for the three scenarios is displayed on Figure 3.2 and the orders are given in Table 3.2. We note on Figure 3.2 that the curves representing the three scenarios have the same decreasing aspect and are confounded. Most

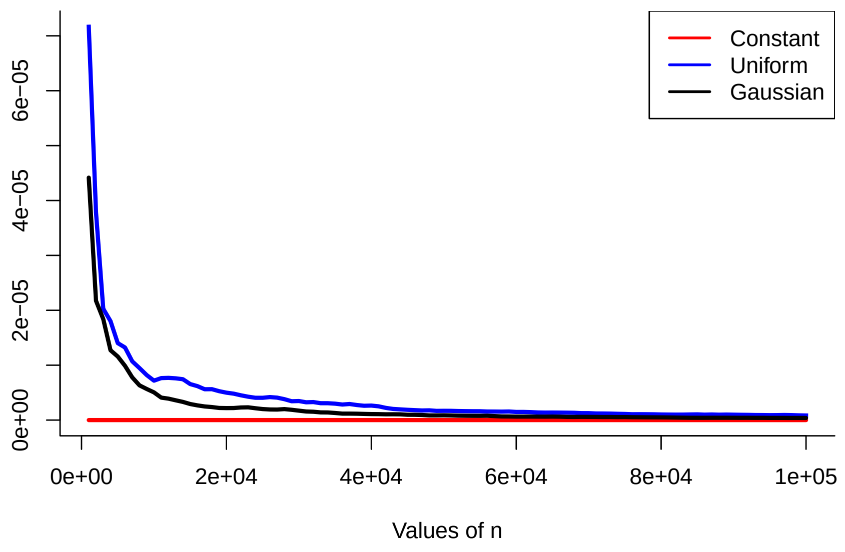

Figure 3.1. Evolution of the squared norm $|\hat{\theta}_n - \theta^0|^2$ for $n$ varying from $10^3$ to $10^5$. The colors red, blue and black respectively correspond to scenarios 1, 2 and 3 for generating $\hat{\theta}_n$.

Table 3.2. Evolution of the order of the error $\|\hat{\boldsymbol{\phi}}_n - \boldsymbol{\phi}^0\|_2^2$ for $n$ varying from $10^3$ to $10^5$.

| Values of $n$ | Scenario 1 | Scenario 2 | Scenario 3 |
|---|---|---|---|
| $10^3$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ |
| $[2 \times 10^3, 1.2 \times 10^4]$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ |
| $[1.3 \times 10^4, 10^5]$ | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ |

importantly, the norm $\|\hat{\boldsymbol{\phi}}_n - \boldsymbol{\phi}^0\|_2^2$ tends to 0 when $n$ increases and tends to $10^5$. This a confirmation of the first part of Theorem 3.4.1.

Before illustrating the second part of Theorem 3.4.1, we introduce a special distribution that will be very useful in the sequel of this section. That is the Dirichlet distribution [68] defined as follows.

**Definition 3.5.1** (Definition of the Dirichlet distribution [68]). Let $\mathbb{S}_{r-1}$ be the simplex of order $r-1$ in $\mathbb{R}^r$ defined by:

$$\mathbb{S}_{r-1} = \Big\{(\phi_1, \ldots, \phi_r) \in \mathbb{R}^r \mid \phi_i > 0, \quad 1 \leqslant i \leqslant r, \quad \sum_{i=1}^{r} \phi_i = 1\Big\}. \tag{3.16}$$

A random vector $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_r) \in \mathbb{S}_{r-1}$ is said to have a Dirichlet distribution if the density of the sub-vector $\boldsymbol{\phi}_{-r} = (\phi_1, \ldots, \phi_{r-1})^T$ is

$$f(\boldsymbol{\phi}_{-r} \mid \mathbf{a}) = \frac{\prod_{i=1}^{r} \Gamma(a_i)}{\Gamma(\sum_{i=1}^{r} a_i)} \prod_{i=1}^{r} \phi_i^{a_i - 1}$$

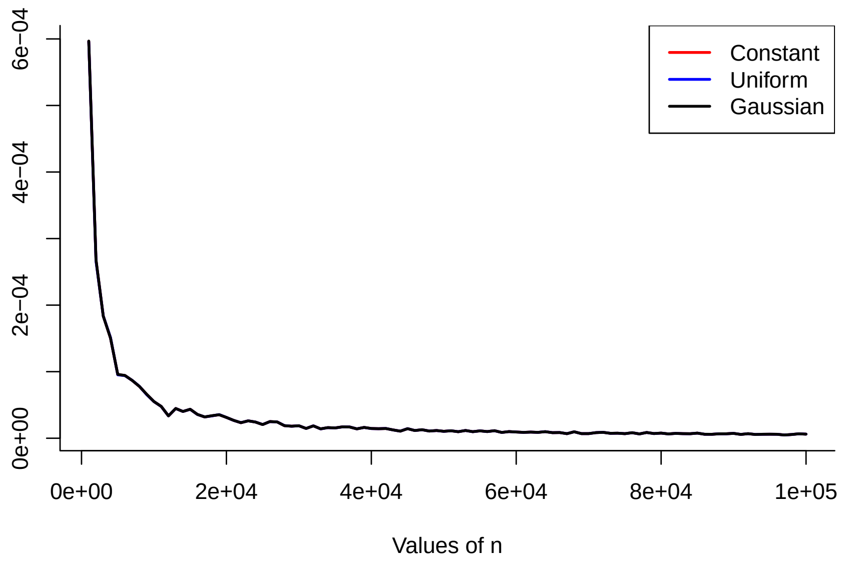

Figure 3.2. Evolution of the norm $\|\hat{\boldsymbol{\phi}}_n - \boldsymbol{\phi}^0\|_2^2$ for $n$ varying from $10^3$ to $10^5$ when $\hat{\theta}_n$ converges a.s. the true value $\theta^0$. The colors red, blue et black correspond respectively to scenarios 1, 2 and 3. The three curves are confounded.

where $\Gamma$ is the classical gamma function

$$\Gamma(s) = \int_0^{+\infty} x^{s-1}\, e^{-x}\, \mathrm{d}x$$

and $\mathbf{a} = (a_1, \ldots, a_r)$ is a vector of positive parameters. A vector $\boldsymbol{\phi}$ following a Dirichlet distribution with parameters $\mathbf{a}$ is denoted $\boldsymbol{\phi} \sim \mathrm{Dir}(a)$.

The following lemma gives the expected values of a Dirichlet distribution.

**Lemma 3.5.1** ([68])**.** *If $\boldsymbol{\phi} \sim \mathrm{Dir}(\mathbf{a})$ in $\mathbb{S}_{r-1}$ and if we denote $a_+ = \sum_{i=1}^r a_i$ then*

$$\mathrm{E}(\phi_i) = \frac{a_i}{a_+}, \quad i = 1, \ldots, r. \tag{3.17}$$

In the second step of the illustrations, the Dirichlet distribution is very useful because it helps us to overcome the additional difficulties introduced by the constraint $\sum_{j=1}^r \phi_j = 1$ existing on the vector $\boldsymbol{\phi}$. For this second step, we consider the following two scenarios for generating $\hat{\boldsymbol{\phi}}$.

– Scenario 4: $\hat{\boldsymbol{\phi}}_n = \boldsymbol{\phi}^0$, and

– Scenario 5: $\hat{\boldsymbol{\phi}}_n = \frac{1}{n}\sum_{i=1}^n \mathbf{Y}_i$ where $\mathbf{Y}_i \sim \mathrm{Dir}(\mathbf{a})$ and $\mathbf{a} = \boldsymbol{\phi}^0$.

The Dirichlet distribution is generated using the `MCMCpack` `R` package developed by Martin et al. [58].

**Remark 3.5.1.** In the scenario 5, the a.s. convergence of $\hat{\boldsymbol{\phi}}_n$ to $\mathrm{E}(\mathbf{Y}_1) = \boldsymbol{\phi}^0$ is guaranteed by the SLLN. This can also be seen in Table 3.3 and Figure 3.3.

Table 3.3. Order of $\|\hat{\boldsymbol{\phi}}_n - \boldsymbol{\phi}^0\|^2$ for $n$ varying from $10^3$ to $10^5$ in scenarios 4 and 5.

| Values of $n$ | Scenario 4 | Scenario 5 |
|---|---|---|
| $[10^3, 5 \times 10^3]$ | 0 | $10^{-4}$ |
| $[6 \times 10^3, 4.5 \times 10^4]$ | - | $10^{-5}$ |
| $[4.6 \times 10^4, 10^5]$ | - | $10^{-6}$ |

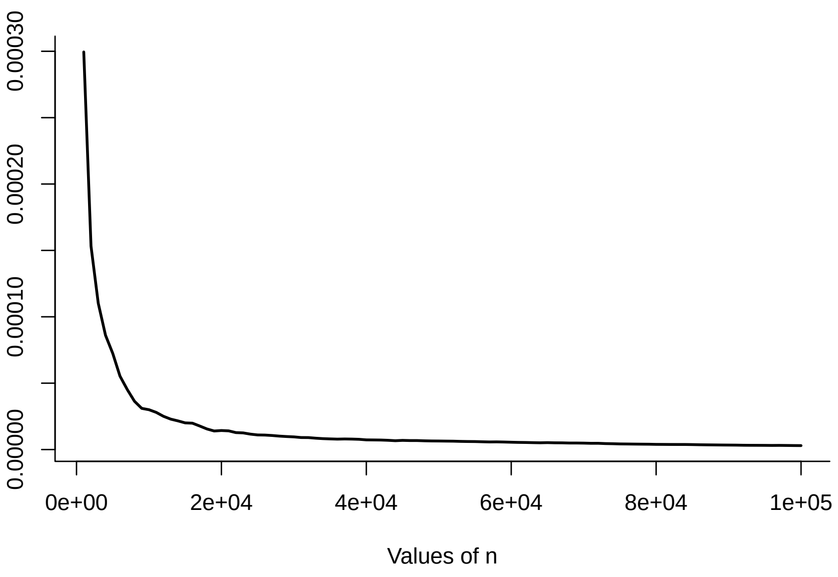

Figure 3.3. Evolution of $|\hat{\phi}_n - \phi^0|^2$ for $n$ varying from $10^3$ to $10^5$ in scenarios 4 and 5.

We study the norm $|\hat{\theta}_n - \theta^0|$ where the estimator $\hat{\theta}_n$ is calculated using the formula:

$$\hat{\theta}_n = \hat{\theta}_n(\hat{\boldsymbol{\phi}}_n) = X_{2+}\left(X_{1+}\sum_{j=1}^{r} z_j \hat{\phi}_{n,j}\right)^{-1}.$$

The evolution of $|\hat{\theta}_n - \theta^0|^2$ for $n$ varying from $10^3$ to $10^5$ when $\hat{\boldsymbol{\phi}}_n$ converges a.s. to the true value $\boldsymbol{\phi}^0$ is given by Table 3.4 and displayed on Figure 3.4. It can be seen that the norm $|\hat{\theta}_n - \theta^0|^2$ tends to 0 when $n$ tends to $10^5$. This is a confirmation of the second part of Theorem 3.4.1.

## 3.6 Conclusion

In this chapter, we studied the strong consistency of the constrained maximum likelihood estimator of a vector parameter when a road safety measure has been applied to a target

Table 3.4. Order of $|\hat{\theta}_n - \theta^0|^2$ for $n$ varying from $10^3$ to $10^5$ in scenarios 4 and 5.

| Values of $n$ | Scenario 4 | Scenario 5 |
|---|---|---|
| $[10^3, 2\times 10^3]$ | $10^{-3}$ | $10^{-3}$ |
| $[3\times 10^3, 2.4\times 10^4]$ | $10^{-4}$ | $10^{-4}$ |
| $[2.5\times 10^4, 10^5]$ | $10^{-5}$ | $10^{-5}$ |

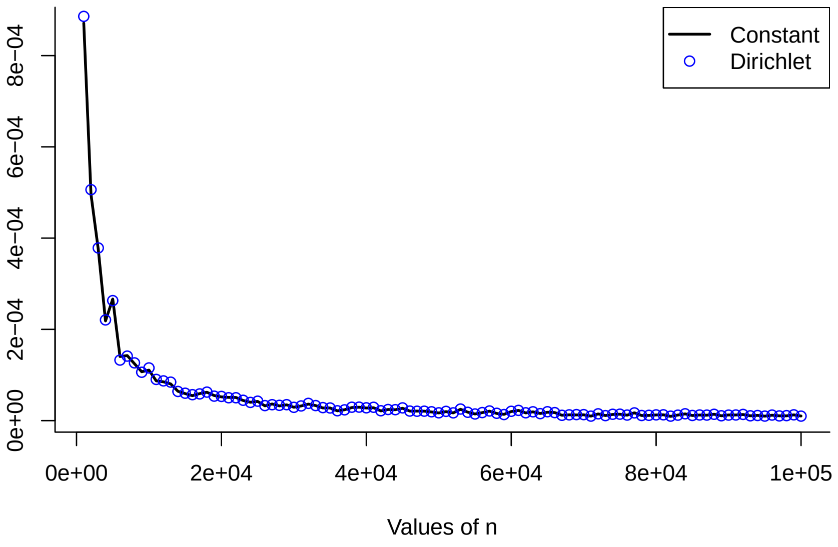

Figure 3.4. Evolution of $|\hat{\theta}_n - \theta^0|^2$ for $n$ varying from $10^3$ to $10^5$ when $\hat{\phi}_n$ converges a.s. to the true value $\phi^0$.

site which counts $r$ classes of accidents. The vector parameter is partitioned under the form $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})$ where $\theta$ is a positive real number while $\boldsymbol{\phi}$ is a vector of dimension $r > 0$ belonging to the simplex of order $r > 1$. We proved the almost sure convergence of the maximum likelihood estimator $\hat{\boldsymbol{\beta}}_n = (\hat{\theta}_n, \hat{\boldsymbol{\phi}}_n)$ of $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})$ by using the cyclic relation which exists between the two subsets of parameters and the numerical simulation studies performed allowed to confirm our theoretical results. A natural extension of this work should be to generalize our results to the multidimensional estimator proposed in [73] when we deal with the estimation of the effect of a road safety measure applied on different experimental sites. Each target site counts $r$ ($r > 1$) mutually exclusive accidents types and is linked to a specific control area where the measure is not directly applied.

# Chapter 4

# Generalization of the Cyclic Algorithm for constrained maximum likelihood estimation

## 4.1 Introduction

As mentioned earlier, Newton-Raphson's algorithm (NR) is known to converge quadratically when the starting point of the iterations is close to the true solution that is unknown in the practice. However if the initial solution is far from the true unknown parameter, this algorithm may fail to converge to the desired value. Moreover, in terms of computational time, numerical inversion of the Hessian matrix at each step of the procedure can be costly in high dimension and may be impossible whenever the Hessian matrix is singular or ill-conditioned. Most of the numerous remedies brought by the scientific results like the quasi-Newton algorithms have proven their efficiency but they come at a cost of greater implementation complexity. In practice, one knows that there exists no "perfect" optimization algorithm and it is up to the statistician to find within several classes of optimization algorithms, the one that best suits his problem. For all these reasons, statisticians keep a few tricks up their sleeves and they have a wide set of other optimization algorithms in their toolkit whenever the NR approach and its modifications fail. A comprehensive review of modern optimization algorithms is available in reference papers and books such as [5, 21, 29, 43, 44, 45, 66, 79].

Since the early 1970's, MM algorithms are introduced and enjoy a large popularization in computational statistics over the years. Actually, MM [36, 46, 108] is more a tool for constructing optimization algorithms. Recall that the MM philosophy for maximizing a function $\ell$ depending on an unknown vector parameter $\boldsymbol{\beta} \in \mathbb{R}^d$ is to define, in the first M step, a minorizing function for the objective function, and to maximize, in the second M step, the minorizing function with respect to the parameter of the underlying model. MM algorithms also enable to transform a difficult optimization problem into a simple one by avoiding for example large matrix inversions and by dealing with equality and inequality constraints gracefully. However this simplification of the original optimization problem can lead to a greater number of iterations or a slow convergence. Moreover MM algorithms can be difficult to implement when the minorizing function is not simple to define.

In this chapter, we consider a special class of algorithms for maximum likelihood estimation called *cyclic algorithms* (CA) and the specific case where the parameter vector $\boldsymbol{\beta}$ can be partitioned into the form $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})$ with $\theta \in \mathbb{R}$ and $\boldsymbol{\phi}$ a vector of dimension $d-1$. A cyclic algorithm for estimation of $\boldsymbol{\beta}$ updates the successive iterates in the following manner: at step $k+1$,

$$\begin{aligned} \theta^{(k+1)} &= \varphi_1(\boldsymbol{\phi}^{(k)}) \\ \boldsymbol{\phi}^{(k+1)} &= \varphi_2(\theta^{(k+1)}) \end{aligned} \tag{4.1}$$

where $\varphi_1$ and $\varphi_2$ are two mappings defined from $\mathbb{R}^{d-1}$ to $\mathbb{R}$ and from $\mathbb{R}$ to $\mathbb{R}^{d-1}$ respectively. Cyclic algorithms are also referred to in the literature as *partitioned algorithms* [94]. They are generally obtained when one proceeds to block optimization [17] but may also obtained while deriving some MM algorithms and are then called cyclic MM algorithms [36, 49, 65]. To solve the simultaneous likelihood equations

$$\frac{\partial \ell(\theta, \boldsymbol{\phi})}{\partial \theta} = 0 \tag{4.2}$$

$$\frac{\partial \ell(\theta, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} = 0 \tag{4.3}$$

one can alternate between solving (4.2) with respect to $\theta$ and (4.3) with respect to $\boldsymbol{\phi}$.

The main specificity of cyclic algorithms is that they cycle through the subsets of components updating one from the other instead of updating the whole parameter vector at once. This can be considered as an advantage since the update of $\boldsymbol{\phi}^{(k+1)}$ takes into account the newly available information on $\theta^{(k+1)}$ instead of $\theta^{(k)}$. Moreover, the cyclic algorithm (4.1) has the effect of restricting the whole parameter estimation to that of $\theta$. Hunter and Lange [36] also claim that cyclic MM algorithms often take fewer iterations than simple MM algorithms because a cyclic MM algorithm always drives the objective function in the right direction.

In Chapter 2, we presented a cyclic algorithm for the estimation of the mean effect and the different accidents risks in the statistical analysis of a road safety measure applied to one experimental site. The numerical experiments on simulated datasets suggest that this algorithm is competitive with the best known algorithms. We also proved that this algorithm is convergent from any starting point. The main motivation of this chapter is to build a cyclic algorithm to estimate the parameters of a model that is a generalization of the one considered in Chapter 2. The main difference is that in this chapter, the road safety measure is applied to $s$ different experimental sites with $s > 1$ while in Chapter 2 we only considered the case $s = 1$.

The remainder of this chapter is organized as follows. In Section 4.2, we describe the data and the problem we are faced with. The statistical model and the constrained maximum likelihood estimation of the model's parameters are also presented. In Section 4.3, we present the principle of the estimation method via the cyclic algorithm while in Section 4.4, we prove that the generalized CA is an ascent algorithm and that it converges to the MLE from any starting point. And finally in Section 4.5, we report on some numerical convergence properties of the CA. We also compare the performance of the CA with those of some algorithms like MM algorithm, quasi-Newton BFGS algorithm, Nelder-Mead's algorithm and Newton's method.

## 4.2 Presentation of the model

### 4.2.1 Data and notations

The problem considered in this chapter is a generalization of the one considered in Chapter 2. That is the multidimensional combination of crash data before and after the introduction of a road safety measure (crossroad lay-out, surface of a motorway section, ...) at $s$ ($s > 0$) experimental sites, having $r$ ($r > 0$) different mutually exclusive accident types (fatal accidents, seriously injured people, slightly injured people, material damage only, ...) over a fixed period of time. The main question is how to estimate the mean effect of the safety measure simultaneously on all the $s$ sites?

For a given site $k$ ($k = 1, \dots, s$), the accidents numbers per type of accidents are collected in the vector

$$\mathbf{x}_k = (x_{11k}, x_{12k}, \dots, x_{1rk}, x_{21k}, x_{22k}, \dots, x_{2rk})^T \in \mathbb{R}^{2r}$$

where $x_{1jk}$ (resp. $x_{2jk}$) represents the number of accidents of type $j$ ($j = 1, \dots, r$) occurred on the site $k$ in the period before (resp. after) the application of the road safety measure and

$$n_k = \sum_{i=1}^{2} \sum_{j=1}^{r} x_{ijk}$$

denotes the total number of accidents counted on site $k$ in both periods. In order to take into account some external factors such as traffic flow, speed limit variation, weather conditions, each experimental site is linked to a much larger area called "control area" where the measure was not directly applied. The accidents data of the control area linked to the experimental site $k$ are described by a vector $\mathbf{Z}_k = (z_{1k}, \dots, z_{rk})^T$ where $z_{jk}$, $j = 1, \dots, r$, $k = 1, \dots, s$ is a non-random variable and represents the ratio of the number of accidents of each type in the "after" period to the number of accidents in the "before" period on the control site. One is faced with the task of combining these crash data of experimental sites and control areas in order to efficiently estimate the mean effect of the measure and the different accident risks relative to all sites and accident types.

### 4.2.2 Statistical model

As mentioned in Chapter 1, depending on the available data, different statistical models can be used to model the mean effect of a road safety measure and the global risks of different accident types. In this chapter, we consider the model proposed by N'Guessan et al. [73]. For a given site $k$, let us consider the random vector

$$\mathbf{X}_k = (X_{11k}, X_{12k}, \dots, X_{1rk}, X_{21k}, X_{22k}, \dots, X_{2rk})^T$$

where $X_{1jk}$ (resp. $X_{2jk}$) is a positive non-zero random variable representing the number of accidents of type $j$ ($j = 1, \dots, r$) occurred in the period before (resp. after) the application of the measure.

### Main underlying assumptions of the selected model

Let the vector

$$\mathbf{x}_k = (x_{11k}, \dots, x_{1rk}, x_{21k}, \dots, x_{2rk})^T$$

be the observed value of $\mathbf{X}_k$ ($k = 1, \dots, s$) and denote by $n_k$ the total number of accidents observed on the experimental site $k$ where the measure was applied i.e.

$$\sum_{i=1}^{2} \sum_{j=1}^{r} x_{ijk} = n_k.$$

The selected model is built under the following assumptions.

(A1) The $s$ random vectors $\mathbf{X}_1, \dots, \mathbf{X}_s$ are assumed to be mutually independent.

(A2) For every $k = 1, \dots, s$,

$$\mathbf{X}_k \sim \mathcal{M}(n_k; \boldsymbol{\pi}_k(\boldsymbol{\beta}))$$

where $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})^T$, $\theta > 0$, $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_s)^T \in (\mathbb{S}_{r-1})^s$,

$$\begin{aligned}
\boldsymbol{\phi}_k &= (\phi_{1k}, \dots, \phi_{rk})^T \in \mathbb{S}_{r-1}, \quad k = 1, \dots, s, \\
\mathbb{S}_{r-1} &= \Big\{ (p_1, \dots, p_r) \quad | \quad 0 < p_j < 1, \quad j = 1, \dots, r \quad \text{and} \quad \sum_{j=1}^{r} p_j = 1 \Big\}, \\
\pi_{ik}(\boldsymbol{\beta}) &= (\pi_{i1k}(\boldsymbol{\beta}), \dots, \pi_{irk}(\boldsymbol{\beta}))^T, \quad i = 1, 2, \\
\pi_{1jk}(\boldsymbol{\beta}) &= \frac{\phi_{jk}}{1 + \theta \langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle}, \quad \pi_{2jk}(\boldsymbol{\beta}) = \frac{\theta z_{jk} \phi_{jk}}{1 + \theta \langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle}, \quad j = 1, \dots, r, \\
\langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle &= \sum_{j=1}^{r} z_{jk} \phi_{jk}.
\end{aligned} \tag{4.4}$$

The parameter of the model (4.4) is then the vector $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})^T$ that belongs to the set $(\mathbb{R}_+^* \times (\mathbb{S}_{r-1})^s) \subset \mathbb{R}^{1+sr}$. In this chapter, we are interested in estimating the value of the unknown vector parameter $\boldsymbol{\beta}$.

**Remark 4.2.1.** The steps of the construction of model (4.4) are not presented here since they are identical to those presented in Chapter 1.

### 4.2.3 Constrained maximum likelihood estimation of the parameters

The likelihood of an observed data $\mathbf{x}_k = (x_{11k}, \dots, x_{1rk}, x_{21k}, \dots, x_{2rk})$, $k = 1, \dots, s$, is given by:

$$L_k(\boldsymbol{\beta} \mid \mathbf{x}_k) = \left( \frac{n_k!}{\prod_{j=1}^{r} \prod_{i=1}^{2} x_{ijk}!} \right) \prod_{j=1}^{r} \frac{\phi_{jk}^{x_{+jk}} (\theta z_{jk})^{x_{2jk}}}{\Big(1 + \theta \langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle\Big)^{x_{+jk}}}, \tag{4.5}$$

where $x_{+jk} = x_{1jk} + x_{2jk}$.

Since it is assumed that the random vectors $\mathbf{X}_1, \ldots, \mathbf{X}_s$ are mutually independent the likelihood of a whole observed accident data

$$\mathbf{x} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_s^T)^T$$

is calculated as the product of the likelihoods associated to each site i.e.

$$L(\boldsymbol{\beta} \mid \mathbf{x}) = L(\boldsymbol{\beta} \mid \mathbf{x}_1, \ldots, \mathbf{x}_s) = \prod_{k=1}^{s} L_k(\boldsymbol{\beta} \mid \mathbf{x}_k).$$

The log-likelihood function is then given up to one additive constant by:

$$\ell(\boldsymbol{\beta}) = \sum_{k=1}^{s} \sum_{j=1}^{r} \Big\{ x_{+jk} \log(\phi_{jk}) + x_{2jk} \log(\theta) - x_{+jk} \log\left(1 + \theta \langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle\right) \Big\}. \tag{4.6}$$

The ML estimation problem is the constrained optimization problem

$$\begin{cases} \max\limits_{\boldsymbol{\beta} \in \mathbb{R}^{1+sr}} \ell(\boldsymbol{\beta}) \\ \text{subject to} \quad \theta > 0, \quad \phi_{jk} > 0, \quad j = 1, \ldots, r, \quad k = 1, \ldots, s, \\ \text{and} \quad \sum_{j=1}^{r} \phi_{jk} = 1, \quad k = 1, \ldots, s. \end{cases} \tag{4.7}$$

The following theorem extracted from [73] expresses the estimate $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ as the solution of a constrained system of non-linear equations.

**Proposition 4.2.1.** *The Maximum Likelihood Estimator (MLE) $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ is solution to the following constrained system of non-linear equations:*

$$\begin{cases} \displaystyle\sum_{k=1}^{s} \sum_{j=1}^{r} \frac{x_{2jk} - \hat{\theta} x_{1jk} \langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle}{1 + \hat{\theta} \langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle} = 0 \\ x_{+jk} \left(1 + \hat{\theta} \langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle\right) - n_k \hat{\phi}_{jk} (1 + \hat{\theta} z_{jk}) = 0, \quad j = 1, \ldots, r; \quad k = 1, \ldots, s. \\ \hat{\theta} > 0, \quad \hat{\phi}_{jk} > 0, \quad j = 1, \ldots, r, \quad k = 1, \ldots, s. \end{cases} \tag{4.8}$$

**Proof**. We only remind the key steps of the proof since it is a generalization of the proof given in Chapter 2 for $s = 1$. Using $s$ Lagrange's multipliers $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_s)^T$ to handle the equality constraints, one gets the augmented log-likelihood

$$\tilde{\ell}(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \ell(\boldsymbol{\beta}) + \sum_{k=1}^{s} \lambda_k \left(1 - \sum_{j=1}^{r} \phi_{jk}\right)$$

that can be rewritten as:

$$\tilde{\ell}(\boldsymbol{\beta},\boldsymbol{\lambda}) = \sum_{k=1}^{s}\sum_{j=1}^{r} x_{+jk}\log(\phi_{jk}) \quad + \quad x_{2++}\log(\theta) \quad - \\ \sum_{k=1}^{s} n_k \log\left(1+\theta\sum_{m=1}^{r} z_{mk}\phi_{mk}\right) \quad + \quad \sum_{k=1}^{s}\lambda_k - \sum_{k=1}^{s}\sum_{j=1}^{r}\lambda_k\phi_{jk} \qquad (4.9)$$

where $x_{2++} = \sum_{j=1}^{r}\sum_{k=1}^{s} x_{2jk}$ and $n_k = \sum_{j=1}^{r}\sum_{i=1}^{2} x_{ijk}$.

The first line of (4.8) is obtained by setting

$$\frac{\partial\tilde{\ell}}{\partial\theta} = \frac{1}{\theta}\sum_{k=1}^{s}\sum_{j=1}^{r}\frac{x_{2jk} - \theta\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k\rangle x_{1jk}}{1+\theta\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k\rangle} = 0.$$

Setting $\partial\tilde{\ell}/\partial\phi_{jk} = 0$, one gets

$$\lambda_k\phi_{jk} = x_{+jk} - \frac{\theta n_k z_{jk}\phi_{jk}}{1+\theta\sum_{m=1}^{r} z_{mk}\phi_{mk}}.$$

Summing on the index $j$ and noting that $\sum_{j=1}^{r}\phi_{jk} = 1$ and $\sum_{j=1}^{r} x_{+jk} = n_k$ leads to the relation

$$\lambda_k = \frac{n_k}{1+\theta\sum_{m=1}^{r} z_{mk}\phi_{mk}}, \quad k = 1,\dots,s. \qquad (4.10)$$

Finally the second line of (4.8) is obtained by setting $\partial\tilde{\ell}/\partial\phi_{jk}$ to 0 and replacing $\lambda_k$ by its value. □

Different iterative methods can be used to solve system (4.8). Most of them need at each iteration not only the computation of first-order or even second-order derivatives but also a matrix inversion or the resolution of a linear system of equations whose matrix can easily have very large number of components when $s$ and $r$ increase. We present here a generalized cyclic algorithm for resolution of the system (4.8) without computing derivatives nor inverting matrix.

## 4.3 The generalized cyclic algorithm

This algorithm is a generalization of that proposed by N'Guessan [71] and presented earlier in Chapter 2. With the purpose of looking for a solution to (4.8), we set the first component $\theta$ and solve the second subsystem in relation to the components $\phi_{jk}$, $j = 1,\dots,r$; $k = 1,\dots,s$ using the Schur complement (presented in Chapter 2). Then we use the solution to solve the first equation of (4.8) with respect to $\theta$.

### Expression of $\hat{\boldsymbol{\phi}}$ as a function of $\hat{\theta}$

**Theorem 4.3.1.** *Given the MLE $\hat{\theta}$ of $\theta$, the vector $\hat{\boldsymbol{\phi}} = (\hat{\boldsymbol{\phi}}_1,\dots,\hat{\boldsymbol{\phi}}_s)^T$ is solution to the linear system*

$$\mathbf{D}_{\hat{\theta}}\hat{\boldsymbol{\phi}} = \mathbf{B} \qquad (4.11)$$

*where*

$$\mathbf{D}_{\hat{\theta}} = \operatorname{diag}(\mathbf{D}_{\hat{\theta},1}, \dots, \mathbf{D}_{\hat{\theta},s}) = \begin{pmatrix} \mathbf{D}_{\hat{\theta},1} & & & \\ & \mathbf{D}_{\hat{\theta},2} & & \\ & & \ddots & \\ & & & \mathbf{D}_{\hat{\theta},s} \end{pmatrix}, \tag{4.12}$$

$$\mathbf{D}_{\hat{\theta},k} = \begin{pmatrix} 1+(1-\frac{x_{+1k}}{n_k})\hat{\theta}z_{1k} & -\frac{x_{+1k}}{n_k}\hat{\theta}z_{2k} & \dots & -\frac{x_{+1k}}{n}\hat{\theta}z_{rk} \\ -\frac{x_{+2k}}{n_k}\hat{\theta}z_{1k} & 1+(1-\frac{x_{+2k}}{n_k})\hat{\theta}z_{2k} & \dots & -\frac{x_{+2k}}{n_k}\hat{\theta}z_{rk} \\ \vdots & \ddots & \ddots & \vdots \\ -\frac{x_{+rk}}{n_k}\hat{\theta}z_{1k} & -\frac{x_{+rk}}{n_k}\hat{\theta}z_{2k} & \dots & 1+(1-\frac{x_{+rk}}{n_k})\hat{\theta}z_{rk} \end{pmatrix}, \quad k = 1, \dots, s,$$

$\mathbf{B} = (\mathbf{B}_1^T, \dots, \mathbf{B}_s^T)^T$ *and*

$$\mathbf{B}_k = \frac{1}{n_k}\left(x_{+1k}, \dots, x_{+rk}\right)^T, \quad k = 1, \dots, s.$$

**Proof**. It is adapted form the proof given in Chapter 2. Actually, the second line of (4.8) is a system of $sr$ equations

$$x_{+jk}\left(1+\hat{\theta}\sum_{m=1}^{r} z_{mk}\hat{\phi}_{mk}\right) - n_k\hat{\phi}_{jk}(1+\hat{\theta}z_{jk}) = 0, \quad j = 1, \dots, r; \quad k = 1, \dots, s.$$

This system is equivalent to:

$$\frac{x_{+jk}}{n_k}\sum_{m\neq j}\hat{\theta}z_{mk}\hat{\phi}_{mk} - \hat{\phi}_{jk}(1+\hat{\theta}z_{jk}) + \frac{x_{+jk}}{n_k}\hat{\theta}z_{jk}\hat{\phi}_{jk} = -\frac{x_{+jk}}{n_k}, \quad j = 1, \dots, r; \quad k = 1, \dots, s.$$

After multiplication by $-1$, we get:

$$-\sum_{m\neq j}\left(\frac{x_{+jk}}{n_k}\hat{\theta}z_{mk}\right)\hat{\phi}_{mk} + \left(1+(1-\frac{x_{+jk}}{n_k})\hat{\theta}z_{jk}\right)\hat{\phi}_{jk} = \frac{x_{+jk}}{n_k}, \quad j = 1, \dots, r; \quad k = 1, \dots, s \tag{4.13}$$

which completes the proof. □

**Remark 4.3.1.** It can be noted that the assumption of independence of the $s$ random vectors $\mathbf{X}_1, \dots, \mathbf{X}_s$ allows to transform the linear system of $sr$ equations and $sr$ variables into $s$ linear sub-systems of $r$ equations each. Each of these linear sub-systems is linked to a given site $k$, $k = 1, \dots, s$. Given $\hat{\theta}$, to find $\hat{\boldsymbol{\phi}}$ it is then sufficient to solve the $s$ linear sub-systems

$$\mathbf{D}_{\hat{\theta},k}\hat{\boldsymbol{\phi}}_k = \mathbf{B}_k, \quad k = 1, \dots, s. \tag{4.14}$$

The following theorem shows that the matrix $\mathbf{D}_{\hat{\theta},k}$ of Equation (4.14) is invertible and gives the closed-form expression of the unique vector solution $\hat{\boldsymbol{\phi}}_k$.

**Theorem 4.3.2.** *For every $k = 1, \ldots, s$, the matrix $\mathbf{D}_{\hat{\theta},k}$ is invertible and therefore the system* (4.14) *has a unique vector solution given by*

$$\hat{\boldsymbol{\phi}}_k = (\mathbf{M}_{\hat{\theta},k}/\boldsymbol{\Delta}_{\hat{\theta},k})^{-1}\boldsymbol{\Delta}_{\hat{\theta},k}^{-1}\mathbf{B}_k$$

*where*

$$\boldsymbol{\Delta}_{\hat{\theta},k} = \begin{pmatrix} 1+\hat{\theta}z_{1k} & & & \\ & 1+\hat{\theta}z_{2k} & & \\ & & \ddots & \\ & & & 1+\hat{\theta}z_{rk} \end{pmatrix} \quad \textit{and} \quad \mathbf{M}_{\hat{\theta},k} = \begin{pmatrix} \boldsymbol{\Delta}_{\hat{\theta},k} & \hat{\theta}\mathbf{B}_k \\ \mathbf{Z}_k^T & 1 \end{pmatrix}.$$

**Proof**. Note that for a given $\hat{\theta}$ and for a fixed $k \in \{1, \ldots, s\}$, the matrices and vectors involved in the linear system (4.14) are the same as those used in Chapter 2. Thus the proof of the theorem can be obtained by adapting the proof of Lemmas 2.3.3 and 2.3.4. It consists in identifying the matrix $\mathbf{D}_{\hat{\theta},k}$ as the Schur complement of 1 in $\mathbf{M}_{\hat{\theta},k}$. By Theorem 2.2.1, we have

$$\det(\mathbf{M}_{\hat{\theta},k}) = \det(\mathbf{M}_{\hat{\theta},k}/1) \times 1 = \det(\mathbf{M}_{\hat{\theta},k}/\boldsymbol{\Delta}_{\hat{\theta},k}) \times \det(\boldsymbol{\Delta}_{\hat{\theta},k}).$$

Since $\mathbf{D}_{\hat{\theta},k} = (\mathbf{M}_{\hat{\theta},k}/1)$ and $(\mathbf{M}_{\hat{\theta},k}/\boldsymbol{\Delta}_{\hat{\theta},k})$ is a real number, we also have

$$\begin{aligned} \det(\mathbf{D}_{\hat{\theta},k}) &= (\mathbf{M}_{\hat{\theta},k}/\boldsymbol{\Delta}_{\hat{\theta},k}) \times \det(\boldsymbol{\Delta}_{\hat{\theta},k}) \\ &= \left(\frac{1}{n_k}\sum_{j=1}^{r}\frac{x_{+jk}}{1+\hat{\theta}z_{jk}}\right) \times \prod_{j=1}^{r}(1+\hat{\theta}z_{jk}) \quad > \quad 0 \end{aligned}$$

and thus the matrix $\mathbf{D}_{\hat{\theta},k}$ is non-singular. To find the expression of $\hat{\boldsymbol{\phi}}_k$, it is sufficient to note that by Lemma 2.2.2:

$$\mathbf{D}_{\hat{\theta},k}^{-1} = (\mathbf{M}_{\hat{\theta},k}/1)^{-1} = \boldsymbol{\Delta}_{\hat{\theta},k}^{-1} + \hat{\theta}\boldsymbol{\Delta}_{\hat{\theta},k}^{-1}\mathbf{B}_k(\mathbf{M}_{\hat{\theta},k}/\boldsymbol{\Delta}_{\hat{\theta},k})^{-1}\mathbf{Z}_k^T\boldsymbol{\Delta}_{\hat{\theta},k}^{-1}$$

hence (see the proof of Lemma 2.3.4)

$$\begin{aligned} \hat{\boldsymbol{\phi}}_k &= \mathbf{D}_{\hat{\theta},k}^{-1}\mathbf{B}_k \\ &= (\mathbf{M}_{\hat{\theta},k}/\boldsymbol{\Delta}_{\hat{\theta},k})^{-1}\boldsymbol{\Delta}_{\hat{\theta},k}^{-1}\mathbf{B}_k. \end{aligned}$$

□

We can now deduce the expression of the components of each vector $\hat{\boldsymbol{\phi}}_k$, $k = 1, \ldots, s$.

**Corollary 4.3.1.** *Given $\hat{\theta}$, the components of the vector*

$$\hat{\boldsymbol{\phi}} = (\hat{\phi}_{11}, \ldots, \hat{\phi}_{r1}, \hat{\phi}_{12}, \ldots, \hat{\phi}_{r2}, \ldots, \hat{\phi}_{1k}, \ldots, \hat{\phi}_{rk}, \ldots, \hat{\phi}_{1s}, \ldots, \hat{\phi}_{rs})^T$$

*are given by:*

$$\hat{\phi}_{jk} = \frac{1}{\sum_{m=1}^{r} \frac{x_{+mk}}{1+\hat{\theta} z_{mk}}} \frac{x_{+jk}}{(1+\hat{\theta} z_{jk})}, \qquad j = 1, \ldots, r; \quad k = 1, \ldots, s. \tag{4.15}$$

**Proof**. It is similar to the proof of Lemma 2.3.4. □

**Expression of $\hat{\theta}$ as a function of $\hat{\phi}$**

**Proposition 4.3.1.** *Suppose that the estimated $\hat{\phi}_{jk}$, $j = 1, \ldots, r$; $k = 1, \ldots, s$ are known. Let $\Psi_{\hat{\phi}}$ be the function from $\mathbb{R}_+^*$ to $\mathbb{R}$ defined by:*

$$\Psi_{\hat{\phi}}(u) = \sum_{k=1}^{s} \frac{n_k}{1 + u\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle} - \sum_{k=1}^{s} x_{1+k}.$$

*Then, the non-linear equation*

$$\Psi_{\hat{\phi}}(u) = 0 \tag{4.16}$$

*accepts the estimate $\hat{\theta}$ of $\theta$ as unique solution.*

**Proof**. It follows from the first line of system (4.8) that

$$\sum_{k=1}^{s} \frac{x_{2+k} - \hat{\theta}\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle x_{1+k}}{1 + \hat{\theta}\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle} = 0.$$

This can be rewritten as:

$$\sum_{k=1}^{s} \frac{n_k - (1 + \hat{\theta}\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle) x_{1+k}}{1 + \hat{\theta}\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle} = 0$$

because $x_{2+k} = n_k - x_{1+k}$. Thus $\Psi_{\hat{\phi}}(\hat{\theta}) = 0$.

Moreover, the function $\Psi_{\hat{\phi}}$ is differentiable and its derivative $\Psi'_{\hat{\phi}}$ verifies:

$$\Psi'_{\hat{\phi}}(u) = -\sum_{k=1}^{s} \frac{n_k \langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle}{(1 + u\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k \rangle)^2} < 0.$$

Thus, $\Psi_{\hat{\phi}}$ is a strictly decreasing function. Since

$$\lim_{u \to 0} \Psi_{\hat{\phi}}(u) \times \lim_{u \to +\infty} \Psi_{\hat{\phi}}(u) < 0,$$

we conclude that $\Psi_{\hat{\phi}}$ is bijective and that it has a unique root. The proof of Proposition 4.3.1 is thus completed. □

The non-linear equation (4.16) is fairly complicated to solve analytically for any $s > 1$ and it seems impossible to find its exact solutions. Even in the particular case of $r = 1$, it remains difficult to solve analytically and Tanner [97] then proposed an approximation method that consisted in calculating the left-hand side for suitable trial values of $\theta$ until a sufficiently accurate approximation of the solution is obtained. Here, we propose instead a numerical approximation of $\theta$ using the Newton-Raphson's algorithm.

**Theorem 4.3.3.** *If the estimated values $\hat{\phi}_{jk}$, $j = 1, \dots, r$, $k = 1, \dots, s$ are known, the iterations of the Newton-Raphson's algorithm for approximating the unique value $\hat{\theta}$ solution of Equation (4.16) are given by*

$$\theta^{(m+1)} = \theta^{(m)} - \frac{\Psi_{\hat{\phi}}(\theta^{(m)})}{\Psi'_{\hat{\phi}}(\theta^{(m)})}, \qquad m = 1, 2, \dots, \tag{4.17}$$

*where the stating guess $\theta^{(0)} > 0$ is given.*

**Remark 4.3.2.** From formula (4.17) it is clear that the larger the value of the derivative $\Psi'_{\hat{\phi}}(\theta^{(m)})$, the smaller the correction $-\Psi_{\hat{\phi}}(\theta^{(m)})/\Psi'_{\hat{\phi}}(\theta^{(m)})$ that has to be added to the iterate $\theta^{(m)}$ in order to obtain $\theta^{(m+1)}$. But if the value of $\Psi'_{\hat{\phi}}(\theta^{(m)})$ tends to zero near the root of Equation (4.16), then the next iterate may be impossible to compute and the Newton's method will fail. In the present case, the derivative $\Psi'_{\hat{\phi}}(u)$ tends to zero only if $u$ tends to $+\infty$ so that the Newton's iterations (4.17) can be assumed to be convenient.

One of the main questions on the Newton's algorithm (4.17) is its numerical convergence to the desired value $\hat{\theta}$. Sufficient conditions that guarantee the convergence of Newton's method from a starting point are given in the literature. For example, we can cite Demidovich and Maron [19], Golub and Ortega [28], Ortega and Rheinbolt [82], Phillips and Taylor [86].

In our work, we consider the following lemma from Demidovich and Maron [19, Section 4.5].

**Lemma 4.3.1.** *Let $f : [a, b] \longrightarrow \mathbb{R}$, $a < b$, be a function such that $f \in C^2[a, b]$. If $f(a)f(b) < 0$, and $f'(u)$ and $f''(u)$ are non-zero and preserve signs over $[a, b]$, then, proceeding from the initial approximation $u^{(0)} \in [a, b]$ which satisfied the inequality*

$$f(u^{(0)})f''(u^{(0)}) > 0, \tag{4.18}$$

*the sequence $(u^{(m)})$, defined by Newton's method,*

$$u^{(m+1)} = u^{(m)} - \frac{f(u^{(m)})}{f'(u^{(m)})}, \qquad m \geqslant 0,$$

*converges, to the sole root of equation $f(u) = 0$ in $[a, b]$ to any degree of accuracy.*

This lemma gives some sufficient conditions that guarantee the convergence of a Newton-type iterative scheme to the true value of the root of any function $f$ having a unique root. In addition to the conditions related to the signs of the first and second derivatives, there is a condition on the starting point. So we face the problem of choosing a convenient starting point $\theta^{(0)}$ in Algorithm (4.17) in order to ensure numerical convergence. We give the following convergence theorem concerning the Newton's algorithm (4.17).

**Theorem 4.3.4.** *If we set the starting point $\theta^{(0)} = 0$ then the Newton's iterations* (4.17) *numerically converge to the desired point that is $\hat{\theta}$.*

**Proof**. Let $u_M = \hat{\theta} + 1$. As $\Psi_{\hat{\phi}}$ is a decreasing function and $\Psi_{\hat{\phi}}(\hat{\theta}) = 0$, we have $\Psi_{\hat{\phi}}(u_M) < 0$. It follows that $\Psi_{\hat{\phi}}(0) \times \Psi_{\hat{\phi}}(u_M) < 0$. Moreover, the function $\Psi_{\hat{\phi}}$ is twice differentiable and its derivatives $\Psi'_{\hat{\phi}}$ and $\Psi''_{\hat{\phi}}$ verify:

$$\Psi'_{\hat{\phi}}(u) = -\sum_{k=1}^{s} \frac{n_k \langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle}{(1 + u\langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle)^2} < 0, \qquad u \in [0, +\infty[,$$

$$\Psi''_{\hat{\phi}}(u) = \sum_{k=1}^{s} \frac{2n_k (\langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle)^2}{(1 + u\langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle)^3} > 0, \qquad u \in [0, +\infty[.$$

We also have $\theta^{(0)} = 0 \in [0, u_M]$. The hypothesis of Lemma 4.3.1 are satisfied and thus we can conclude that the sequence $(\theta^{(m)})$ converges to $\hat{\theta}$. □

**Remark 4.3.3.** In our implementation of the Newton's algorithm, we have chosen $\theta^{(0)} = 0$ as starting point and we proved that numerical convergence is then guaranteed. Actually since $\Psi''_{\hat{\phi}}(u) > 0$ for all $u > 0$, any starting point $\theta^{(0)}$ such that $\Psi_{\hat{\phi}}(\theta^{(0)}) > 0$ could also make it. If one takes into account the fact that $\Psi_{\hat{\phi}}(\hat{\theta}) = 0$ and also that $\Psi_{\hat{\phi}}$ is bijective and decreasing, it is equivalent to say that any value $\theta^{(0)} < \hat{\theta}$ should also guarantee the numerical convergence. However, since $\hat{\theta}$ is not known before running the algorithm 4.17, the starting point $\theta^{(0)} = 0$ seems to be ideal and it also serves as automatic starting point.

**Remark 4.3.4.** Demidovich and Maron [19] strongly recommend to chose the starting value in order to satisfy the condition $\Psi_{\hat{\phi}}(\theta^{(0)})\Psi''_{\hat{\phi}}(\theta^{(0)}) > 0$. One question that arises is what would happen if we chose the starting point $\theta^{(0)}$ such that $\Psi_{\hat{\phi}}(\theta^{(0)})\Psi''_{\hat{\phi}}(\theta^{(0)}) < 0$. In this case, it is possible that we get a point $\theta^{(1)}$ lying outside the interval $[a, b]$ and there is not much sense in using the algorithm any longer. This is illustrated by the following example.

**Example 4.3.1.** *We set $s = 2$, $r = 3$, $n_1 = n_2 = 500$ and the true parameters are $\theta^0 = 3$, $\boldsymbol{\phi}_1 = (0.2, 0.3, 0.5)^T$, $\boldsymbol{\phi}_2 = (0.05, 0.15, 0.8)^T$. The starting point is $\theta^{(0)} = 8$. Since $\theta^{(0)} > \theta^0$ we have $\Psi_{\hat{\phi}}(\theta^{(0)})\Psi''_{\hat{\phi}}(\theta^{(0)}) < 0$. It is seen on Figure 4.1 and Table 4.1 that the first iterate $\theta^{(1)}$ lies outside the interval* $[0, +\infty[$. *The next iterates are also negative and the sequence $\theta^{(k)}$ diverges to negative infinity.*

### The cyclic algorithm

It follows from Equations (4.16) and (4.15) that the components $\hat{\theta}$ and $\hat{\phi}_{jk}$ ($j = 1, \ldots, r$, $k = 1, \ldots, s$) are not separable. This link between the two sub-parameters makes any attempt of simultaneously updating them difficult. And the fact that $\hat{\theta}$ cannot be obtained in closed-form is a further evidence of it. The MLE $\hat{\boldsymbol{\beta}}$ can then only be obtained by an iterative procedure alternating between updating $\hat{\theta}$ from the $\hat{\boldsymbol{\phi}}$ and updating $\hat{\boldsymbol{\phi}}$ from $\hat{\theta}$.

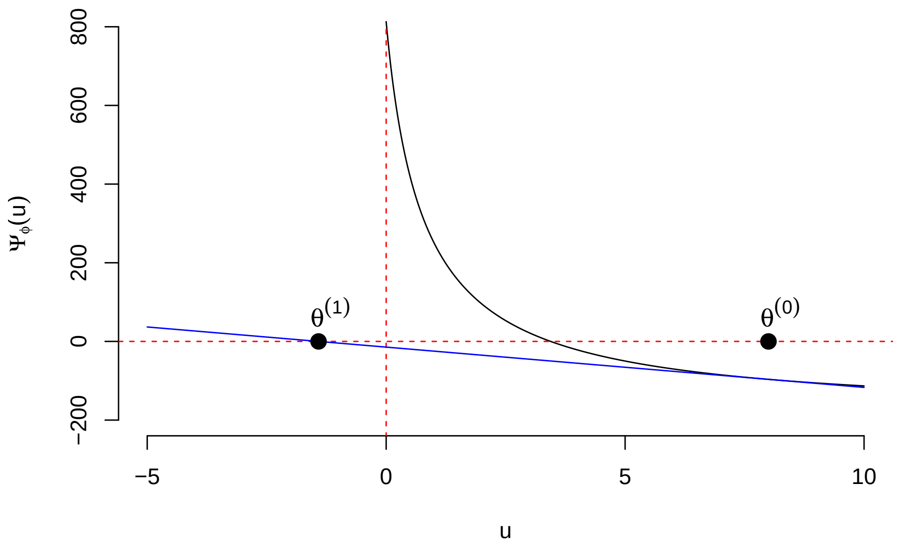

Figure 4.1. Illustration of the misbehaviour of the Newton's algorithm when the starting point $\theta^{(0)}$ does not satisfy the condition $\Psi_{\hat{\phi}}(\theta^{(0)})\Psi''_{\hat{\phi}}(\theta^{(0)}) > 0$. The axis are drawn with red dashed lines. The curve of $\Psi_{\hat{\phi}}(u)$ is drawn in black while the tangent at the point $\left(\theta^{(0)}, \Psi_{\hat{\phi}}(\theta^{(0)})\right)$ is drawn in blue.

Table 4.1. Successive iterates in the Newton's algorithm when the starting point $\theta^{(0)}$ does not satisfy the condition $\Psi_{\hat{\phi}}(\theta^{(0)})\Psi''_{\hat{\phi}}(\theta^{(0)}) > 0$.

| Iteration $k$ | $\theta^{(k)}$ |
|---|---|
| 0 | 8 |
| 1 | -1.414190 |
| 2 | -1.937741 |
| 4 | -6.846501 |
| 7 | -4673.652 |
| 8 | -5.105619 e+06 |
| 13 | -1.171109e+195 |

We therefore propose the following cyclic algorithm. For example if the starting sub-parameter $\theta^{(0)}$ is given, the algorithm computes automatically the missing sub-parameter $\boldsymbol{\phi}^{(0)} = (\boldsymbol{\phi}_1^{(0)}, \ldots, \boldsymbol{\phi}_s^{(0)})$ using the value of $\theta^{(0)}$. At the $(k+1)-$step, the update $\theta^{(k+1)}$ is calculated from $\boldsymbol{\phi}^{(k)}$, afterwards $\boldsymbol{\phi}^{(k+1)}$ is calculated from the $\theta^{(k+1)}$ and so on. This process is repeated until a convergence criteria is satisfied. A version of the algorithm with initial sub-parameter $(\boldsymbol{\phi}_1^{(0)}, \ldots, \boldsymbol{\phi}_s^{(0)})$ is given below (see Algorithm 4.1).

**Remark 4.3.5.** Since the $1 + sr$ components are computed one by one, it makes our proposed cyclic algorithm easy to implement. We deduce from Theorem 4.3.4 and Equation

**Algorithm 4.1** Generalization of the cyclic algorithm

From a starting vector parameter $\boldsymbol{\phi}^{(0)} = (\boldsymbol{\phi}_1^{(0)}, \ldots, \boldsymbol{\phi}_s^{(0)})^T$ such that for every $k = 1, \ldots, s$, $\boldsymbol{\phi}_k^{(0)} = (\phi_{1k}^{(0)}, \ldots, \phi_{rk}^{(0)})^T$ and $\sum_{j=1}^{r} \phi_{jk} = 1$, the CA updates are computed as follows.

(a) Update $\theta^{(m+1)}$ as the limit of the Newton's iteration:

$$u^{(i+1)} = u^{(i)} - \frac{\Psi_{\boldsymbol{\phi}^{(m)}}(u^{(i)})}{\Psi'_{\boldsymbol{\phi}^{(m)}}(u^{(i)})}, \qquad i = 1, 2, \ldots, \tag{4.19}$$

where $u^{(0)} = 0$.

(b) For every $k = 1, \ldots, s$, update $\boldsymbol{\phi}_k^{(m+1)} = (\phi_{1k}^{(m+1)}, \ldots, \phi_{rk}^{(m+1)})^T$:

$$\phi_{jk}^{(m+1)} = \frac{1}{\displaystyle\sum_{l=1}^{r} \frac{x_{+lk}}{1 + \theta^{(m+1)} z_{lk}}} \times \frac{x_{+jk}}{1 + \theta^{(m+1)} z_{jk}}, \qquad j = 1, 2, \ldots, r.$$

(4.15) that, if the starting vector $\boldsymbol{\phi}^{(0)}$ has all its components positive then at the $m-$step of our algorithm ($m > 0$), all the components of the vector parameter $\boldsymbol{\beta}^{(m)}$ are positive and the constraint $\sum_{j=1}^{r} \phi_{jk}^{(m)} = 1$ is also satisfied for every $k = 1, \ldots, s$.

## 4.4 Theoretical study of some numerical properties of the proposed cyclic algorithm

In this section, we want to prove that the cyclic algorithm 4.1 proposed in this chapter verifies the two main properties generally required for maximum likelihood estimation iterative algorithms and demonstrated in Chapter 2 for the specific case $s = 1$. The first one is the convergence of the iterative algorithm 4.1 to the MLE $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})$ from the algorithmic point of view. Proving convergence of an iterative optimization algorithm is generally a delicate exercise that is even more complicated in this case since we do not have the closed-form expression of $\hat{\theta}$.

### 4.4.1 Main results

The first main result that we prove in this section is stated by the following theorem.

**Theorem 4.4.1.** *For all starting point $\boldsymbol{\beta}^{(0)} = (\theta^{(0)}, \boldsymbol{\phi}^{(0)})$, the cyclic algorithm 4.1 converges to the MLE $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})$ of $\boldsymbol{\beta} = (\theta, \boldsymbol{\phi})$.*

The second property that we prove is the ascent property of the cyclic algorithm 4.1 i.e. the fact that the log-likelihood is increased monotonically by the algorithm. That is given by the following theorem.

**Theorem 4.4.2.** *The cyclic algorithm 4.1 enjoys the ascent property, that is*

$$\ell(\boldsymbol{\beta}^{(m+1)}) \geqslant \ell(\boldsymbol{\beta}^{(m)}), \quad m = 0, 1, \ldots \tag{4.20}$$

### 4.4.2 Proofs

The proof of Theorem 4.4.1 is based on the following lemmas that are adapted from Lemmas 2.4.1 and 2.4.2 of Chapter 2.

**Lemma 4.4.1.** *Let $\psi$ be the mapping defined on $\mathbb{R}_+$ by*

$$\psi(u) = \sum_{k=1}^{s}\sum_{j=1}^{r} \frac{x_{+jk}}{(1+uz_{jk})} - x_{1++} \tag{4.21}$$

*where $x_{1++} = \sum_{k=1}^{s}\sum_{j=1}^{r} x_{1jk}$.*

**i)** *There exists a unique real number $u > 0$, denoted $\theta^*$, that is solution to the equation $\psi(u) = 0$.*

**ii)** *Let $u \in ]0, +\infty[$. Then, $\psi(u) \geqslant 0$ if $0 < u \leqslant \theta^*$ and $\psi(u) \leqslant 0$ if $u \geqslant \theta^*$.*

**iii)** *The MLE $\hat{\theta}$ of $\theta$ is equal to the unique root $\theta^*$ of $\psi$.*

**Proof**. **i)** It is easily seen that $\psi$ is continuous and its derivative $\psi'(u)$ is strictly negative for every $u > 0$ and therefore $\psi$ is bijective. Moreover,

$$\lim_{u\to 0}\psi(u) \times \lim_{u\to+\infty}\psi(u) = \Big(\sum_{k=1}^{s}\sum_{j=1}^{r} x_{2jk}\Big) \times (-x_{1++}) \ < \ 0$$

hence the equation $\psi(u) = 0$ has a unique solution.

**ii)** The function $\psi$ is a strictly decreasing function. Therefore,

$$\forall u \leqslant \theta^*, \quad \psi(u) \geqslant \psi(\theta^*) = 0 \quad \text{and} \quad \forall u \geqslant \theta^*, \quad \psi(u) \leqslant \psi(\theta^*) = 0.$$

**iii)** For every $k = 1, \ldots, s$, the equality

$$\hat{\phi}_{jk} = \frac{x_{+jk}/(1+\hat{\theta}z_{jk})}{\sum_{m=1}^{r} x_{+mk}/(1+\hat{\theta}z_{mk})}, \quad j = 1, \ldots, r$$

yields

$$\langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k\rangle = \sum_{j=1}^{r} z_{jk}\hat{\phi}_{jk} = \left(\sum_{j=1}^{r}\frac{z_{jk}x_{+jk}}{1+\hat{\theta}z_{jk}}\right)\Big/\left(\sum_{m=1}^{r}\frac{x_{+mk}}{1+\hat{\theta}z_{mk}}\right)$$

and

$$\begin{aligned}
1+\hat{\theta}\langle \mathbf{Z}_k,\hat{\boldsymbol{\phi}}_k\rangle &= 1+\hat{\theta}\left(\sum_{j=1}^{r}\frac{z_{jk}x_{+jk}}{1+\hat{\theta}z_{jk}}\right)\Big/\left(\sum_{m=1}^{r}\frac{x_{+mk}}{1+\hat{\theta}z_{mk}}\right)\\
&= \left(\sum_{j=1}^{r}\frac{x_{+jk}}{1+\hat{\theta}z_{jk}}+\sum_{j=1}^{r}\frac{\hat{\theta}z_{jk}x_{+jk}}{1+\hat{\theta}z_{jk}}\right)\Big/\left(\sum_{m=1}^{r}\frac{x_{+mk}}{1+\hat{\theta}z_{mk}}\right)\\
&= \left(\sum_{j=1}^{r}\frac{x_{+jk}(1+\hat{\theta}z_{jk})}{1+\hat{\theta}z_{jk}}\right)\Big/\left(\sum_{m=1}^{r}\frac{x_{+mk}}{1+\hat{\theta}z_{mk}}\right)\\
1+\hat{\theta}\langle \mathbf{Z}_k,\hat{\boldsymbol{\phi}}_k\rangle &= n_k\Big/\sum_{m=1}^{r}\frac{x_{+mk}}{1+\hat{\theta}z_{mk}} \qquad (4.22)
\end{aligned}$$

since $\sum_{j=1}^{r} x_{+jk}=n_k$. By summing on the index $k$, we get:

$$\sum_{k=1}^{s}\frac{n_k}{1+\hat{\theta}\langle \mathbf{Z}_k,\hat{\boldsymbol{\phi}}_k\rangle}=\sum_{k=1}^{s}\sum_{m=1}^{r}\frac{x_{+mk}}{1+\hat{\theta}z_{mk}}.$$

But we have also proved (see Proposition 4.3.1) that

$$\sum_{k=1}^{s}\frac{n_k}{1+\hat{\theta}\langle \mathbf{Z}_k,\hat{\boldsymbol{\phi}}_k\rangle}-x_{1++}=0.$$

This is equivalent to

$$\sum_{k=1}^{s}\sum_{m=1}^{r}\frac{x_{+mk}}{1+\hat{\theta}z_{mk}}-x_{1++}=0$$

hence the equality $\psi(\hat{\theta})=0$. Since $\psi$ is bijective and $\psi(\theta^*)=0$ then $\hat{\theta}=\theta^*$. This completes the proof. $\square$

**Lemma 4.4.2.** *There exists a continuous function $\varphi$ from $]0;+\infty[$ to $]0;+\infty[$ such that*

$$\theta^{(m+1)}=\varphi(\theta^{(m)})$$

*and satisfying the following properties:*

**i)** *The function $\varphi$ is an increasing function and*

$$\sup_{u>0}\varphi(u)=\lim_{u\to+\infty}\varphi(u) < \infty. \qquad (4.23)$$

**ii)** *There exists a unique point $\theta^*$ such that $\varphi(\theta^*)=\theta^*$.*

*Moreover,*

**iii)** *The sequence $(\theta^{(k)})$ is monotonous and its monotony depends on $\theta^{(0)}$. It is an increasing sequence if $\theta^{(0)}\leqslant\theta^*$ and decreasing if $\theta^{(0)}\geqslant\theta^*$.*

**iv)** *The sequence $(\theta^{(k)})$ is also bounded. Then it is convergent and its limit is $\theta^*$.*

The proof owes to the following lemma that is an adaptation of the Implicit Function Theorem [40, 90, 96] to a function of two variables.

**Lemma 4.4.3** (Adaptation of the Implicit Function Theorem to a function of two variables)**.** *Let $g$ be a real-valued continuously differentiable function defined in a neighbourhood of $(a,b) \in \mathbb{R}^2$. Suppose that $g$ satisfies the two conditions*

$$g(a,b) = c \qquad \text{and} \qquad \frac{\partial g}{\partial v}(a,b) \neq 0.$$

*Then there exist open intervals $\mathbb{U}$ and $\mathbb{V}$, with $(a,b) \in \mathbb{U} \times \mathbb{V}$, and a unique function $\varphi : \mathbb{U} \to \mathbb{V}$ satisfying*

$$g(u, \varphi(u)) = c, \qquad \text{for all } u \in \mathbb{U}$$

*and this function $\varphi$ is continuously differentiable with*

$$\varphi'(u) = -\left(\frac{\partial g}{\partial v}(u, \varphi(u))\right)^{-1} \cdot \frac{\partial g}{\partial u}(u, \varphi(u)). \tag{4.24}$$

**Proof of Lemma 4.4.2**

At the $(m+1)-$step of the algorithm, $\theta^{(m+1)}$ is estimated from $\boldsymbol{\phi}^{(m)}$ as the unique solution of the equation

$$\sum_{k=1}^{s} \frac{n_k}{1 + \hat{\theta}^{(m+1)} \langle \mathbf{Z}_k, \hat{\boldsymbol{\phi}}_k^{(m)} \rangle} - x_{1++} = 0$$

and

$$\langle \mathbf{Z}_k, \boldsymbol{\phi}_k^{(m)} \rangle = \sum_{j=1}^{r} z_{jk} \hat{\phi}_{jk}^{(m)} = \left(\sum_{j=1}^{r} \frac{z_{jk} x_{+jk}}{1 + \hat{\theta}^{(m)} z_{jk}}\right) \Big/ \left(\sum_{i=1}^{r} \frac{x_{+ik}}{1 + \theta^{(m)} z_{ik}}\right)$$

Then the iterates $\theta^{(m)}$ and $\theta^{(m+1)}$ are linked by the relationship:

$$\sum_{k=1}^{s} \left(n_k \sum_{i=1}^{r} \frac{x_{+ik}}{1 + \theta^{(m)} z_{ik}}\right) \left(\sum_{j=1}^{r} \frac{x_{+jk}(1 + \theta^{(m+1)} z_{jk})}{1 + \theta^{(m)} z_{jk}}\right)^{-1} = x_{1++} \tag{4.25}$$

Let $g_1, \ldots, g_s$ and $g$ be the functions from $(\mathbb{R}_+^*)^2$ to $\mathbb{R}_+^*$ defined by

$$g(u,v) = \sum_{k=1}^{s} n_k g_k(u,v)$$

and

$$g_k(u,v) = \left(\sum_{i=1}^{r} \frac{x_{+ik}}{1 + u z_{ik}}\right) \left(\sum_{j=1}^{r} \frac{x_{+jk}(1 + v z_{jk})}{1 + u z_{jk}}\right)^{-1}, \qquad k = 1, \ldots, s. \tag{4.26}$$

For a fixed $u > 0$, let $g(u, \cdot)$ be the function from $\mathbb{R}_+^*$ to itself defined by $g(u,\cdot)(v) = g(u,v)$. Then for every constant $c \in \,]0,n]$ the equation

$$g(u,\cdot)(v) = c$$

has a unique solution $v > 0$. Indeed the function $g(u, \cdot)$ is differentiable and its derivative

$$(g(u, \cdot))'(v) = \sum_{k=1}^{s} n_k \frac{N_{2,k}(u,v)}{(D_k(u,v))^2}$$

where

$$\begin{aligned} N_{2,k}(u,v) &= -\sum_{i=1}^{r} \frac{x_{+ik}}{1+uz_{ik}} \sum_{j=1}^{r} \frac{z_{jk}x_{+jk}}{1+uz_{jk}} \\ D_k(u,v) &= \sum_{j=1}^{r} \frac{x_{+jk}(1+vz_{jk})}{1+uz_{jk}}. \end{aligned} \tag{4.27}$$

The derivative of $g(u, \cdot)$ is strictly negative hence $g(u, \cdot)$ is strictly decreasing and therefore is a bijective function. Moreover,

$$\begin{aligned} \lim_{v \to 0} g(u, \cdot)(v) &= \sum_{k=1}^{s} n_k = n \\ \lim_{v \to +\infty} g(u, \cdot)(v) &= 0. \end{aligned}$$

We have $x_{1++} \in ]0, n]$ and with the demonstration above, it is clear that for every $u > 0$, there exists a unique $v$ such that $g(u, v) = x_{1++}$. Let $\varphi$ be the function that assigns to $u > 0$ the unique $v$ such that $g(u, v) = x_{1++}$. By Lemma 4.4.3, we can conclude that $\varphi$ is continuous. From Equation (4.25), we can write $g(\theta^{(k)}, \theta^{(k+1)}) = x_{1++}$ and it is clear that $\theta^{(k+1)} = \varphi(\theta^{(k)})$.

**i)** Let $h$ be the function from $\mathbb{R}_+^*$ into itself defined by $h(u) = g(u, \varphi(u))$. As the function $h$ is equal to the constant $x_{1++}$ for all $u > 0$, then $h'(u) = 0$ for every $u > 0$. By the Implicit Function Theorem [90, 96], we have:

$$h'(u) = \frac{\partial g}{\partial u}(u, \varphi(u)) + \varphi'(u) \cdot \frac{\partial g}{\partial v}(u, \varphi(u)) = 0, \quad u > 0$$

or equivalently

$$\varphi'(u) = -\left(\frac{\partial g}{\partial v}(u, \varphi(u))\right)^{-1} \cdot \frac{\partial g}{\partial u}(u, \varphi(u)). \tag{4.28}$$

After a few calculus, we get:

$$\frac{\partial g_k}{\partial u}(u,v) = \frac{N_{1,k}(u,v)}{(D_k(u,v))^2}, \qquad \frac{\partial g_k}{\partial v}(u,v) = \frac{N_{2,k}(u,v)}{(D_k(u,v))^2}$$

where

$$N_{1,k}(u,v) = -\sum_{i=1}^{i} \frac{z_{ik}x_{+ik}}{(1+uz_{ik})^2} \sum_{j=1}^{r} \frac{x_{+jk}(1+vz_{jk})}{1+uz_{jk}} + \sum_{i=1}^{r} \frac{x_{+ik}}{1+uz_{ik}} \sum_{j=1}^{r} \frac{z_{jk}x_{+jk}(1+vz_{jk})}{(1+uz_{jk})^2}$$

while the terms $N_{2,k}(u, v)$ and $D_k(u, v)$ are defined by formula (4.27).

Let us rewrite $N_{1,k}(u, v)$ in a simplified form. We have:

$$\begin{aligned}
N_{1,k}(u,v) &= \sum_{i=1}^{r}\sum_{j=1}^{r}\left\{-\frac{z_{ik}(1+vz_{jk})x_{+ik}x_{+jk}}{(1+uz_{ik})^2(1+uz_{jk})}+\frac{z_{ik}(1+vz_{ik})x_{+ik}x_{+jk}}{(1+uz_{ik})^2(1+uz_{jk})}\right\}\\
&= \sum_{i\neq j}\left\{\frac{v((z_{ik})^2-z_{ik}z_{jk})x_{+ik}x_{+jk}}{(1+uz_{ik})^2(1+uz_{jk})}\right\}\\
&= \sum_{i<j}\left\{\frac{v((z_{ik})^2-z_{ik}z_{jk})x_{+ik}x_{+jk}}{(1+uz_{ik})^2(1+uz_{jk})}\right\}+\sum_{j<i}\left\{\frac{v((z_{ik})^2-z_{ik}z_{jk})x_{+ik}x_{+jk}}{(1+uz_{ik})^2(1+uz_{jk})}\right\}
\end{aligned}$$

By swapping indices $i$ and $j$ in the second right-hand term, we get

$$\begin{aligned}
N_{1,k}(u,v) &= \sum_{i<j}\left\{\frac{v((z_{ik})^2-z_{ik}z_{jk})x_{+ik}x_{+jk}}{(1+uz_{ik})^2(1+uz_{jk})}+\frac{v((z_{jk})^2-z_{ik}z_{jk})x_{+ik}x_{+jk}}{(1+uz_{ik})(1+uz_{jk})^2}\right\}\\
&= \sum_{i<j}\left\{\frac{vx_{+ik}x_{+jk}}{(1+uz_{ik})(1+uz_{jk})}\left(\frac{(z_{ik})^2-z_{ik}z_{jk}}{(1+uz_{ik})}+\frac{(z_{jk})^2-z_{ik}z_{jk}}{(1+uz_{jk})}\right)\right\}\\
&= \sum_{i<j}\left\{\frac{vx_{+ik}x_{+jk}}{(1+uz_{ik})(1+uz_{jk})}\left(\frac{(z_{ik}-z_{jk})^2}{(1+uz_{ik})(1+uz_{jk})}\right)\right\}\geqslant 0.
\end{aligned}$$

Thus for all $u>0$,

$$\frac{\partial g}{\partial u}(u,\varphi(u))=\sum_{k=1}^{s}n_k\frac{\partial g_k}{\partial u}(u,\varphi(u))=\sum_{k=1}^{s}n_k\frac{N_{1,k}(u,\varphi(u))}{(D_k(u,\varphi(u)))^2}\geqslant 0$$

and

$$\frac{\partial g}{\partial v}(u,\varphi(u))=\sum_{k=1}^{s}n_k\frac{\partial g_k}{\partial v}(u,\varphi(u))=\sum_{k=1}^{s}n_k\frac{N_{2,k}(u,\varphi(u))}{(D_k(u,\varphi(u)))^2}\ <\ 0.$$

By (4.28), $\varphi'(u)\geqslant 0$ for all $u>0$ and therefore $\varphi$ is an increasing function.

Now suppose that $u\to+\infty$. Since $\varphi$ is an increasing function, either $\varphi(u)$ tends to a constant or it tends to $+\infty$. Let us prove by contradiction that $\varphi(u)$ tends to a finite constant. Suppose that $\varphi(u)\to+\infty$. Then by dividing both the numerator and denominator of Equation (4.26) by $u$, we get

$$g_k(u,\varphi(u))=\left(\sum_{i=1}^{r}\frac{x_{+ik}}{1/u+z_{ik}}\right)\left(\sum_{j=1}^{r}\frac{x_{+jk}(1+\varphi(u)z_{jk})}{1/u+z_{jk}}\right)^{-1},\qquad k=1,\ldots,s. \tag{4.29}$$

Thus $1/u\to 0$, $\varphi(u)\to+\infty$ and

$$g(u,\varphi(u))=\sum_{k=1}^{s}n_kg_k(u,\varphi(u))\ \to\ 0.$$

This is absurd because $g(u,\varphi(u))=x_{1++}\neq 0$. Therefore we conclude that

$$\lim_{u\to+\infty}\varphi(u)<\infty.$$

**ii)** The equation $\varphi(u) = u$ is equivalent to $g(u,u) = x_{1++}$ i.e. $\psi(u) = 0$ where $\psi$ is the function defined in Lemma 4.4.1. By the latter lemma, this equation has a unique solution that is $\theta^*$, therefore $\theta^*$ is the unique fixed point of $\varphi$.

**iii)** Only two cases are possible: either $\theta^{(0)} \leqslant \theta^*$ or $\theta^{(0)} \geqslant \theta^*$.

- Assume that $\theta^{(0)} \leqslant \theta^*$. Then $\theta^{(0)} \leqslant \theta^{(1)}$. Indeed, if we had $\theta^{(1)} < \theta^{(0)}$, then we would also have
$$\theta^{(2)} = \varphi(\theta^{(1)}) \leqslant \varphi(\theta^{(0)}) = \theta^{(1)}$$
and by induction $\theta^{(m+1)} < \theta^{(m)} < \ldots < \theta^{(1)} < \theta^{(0)}$ because $\varphi$ is an increasing function. Thus the sequence $(\theta^{(m)})$ would be strictly decreasing and lower-bounded by 0 i.e. it would converge to its infimum that is all but $\theta^*$ (since $\theta^*$ is an upper bound). This is absurd because $\theta^*$ is the only possible value for the limit of $(\theta^{(m)})$ if its exists. So we conclude that if $\theta^{(0)} \leqslant \theta^*$ then $\theta^{(0)} \leqslant \theta^{(1)}$.

  We will then have
$$\theta^{(1)} = \varphi(\theta^{(0)}) \leqslant \varphi(\theta^{(1)}) = \theta^{(2)}$$
and by induction $\theta^{(0)} \leqslant \theta^{(1)} \leqslant \ldots \leqslant \theta^{(m)} \leqslant \theta^{(m+1)}$. We conclude that the sequence $(\theta^{(m)})$ is an increasing sequence.

- Assume that $\theta^{(0)} \geqslant \theta^*$. Then one can prove using similar arguments that $\theta^{(1)} \leqslant \theta^{(0)}$. Indeed if we had $\theta^{(0)} < \theta^{(1)}$ then we would also have by induction $\theta^{(0)} < \theta^{(1)} < \ldots < \theta^{(m)} < \theta^{(m+1)}$ because $\varphi$ is an increasing function. Thus the sequence $(\theta^{(m)})$ would be strictly increasing and upper-bounded by
$$\max\left(\theta^{(0)}, \lim_{u\to+\infty} \varphi(u)\right) \;<\; +\infty$$
i.e. it would converge to its supremum that is all but $\theta^*$ (since $\theta^*$ is a lower bound). Here also, this is absurd because $\theta^*$ is the only possible value for the limit of $(\theta^{(m)})$ if its exists. So we conclude that if $\theta^{(0)} \geqslant \theta^*$ then $\theta^{(1)} \leqslant \theta^{(0)}$ and by induction $\theta^{(m+1)} \leqslant \theta^{(m)} \leqslant \ldots \leqslant \theta^{(1)} \leqslant \theta^{(0)}$. We conclude that the sequence $(\theta^{(m)})$ is a decreasing sequence.

**iv)** We have
$$0 < \theta^{(k)} < \max\left(\theta^{(0)}, \lim_{u\to+\infty} \varphi(u)\right) \;<\; +\infty.$$

The real sequence $\theta^{(m)}$ is monotonic and bounded. Thus it converges to $\theta^*$ the only fixed point of the function $\varphi$ that is also equal to the MLE $\hat{\theta}$. □

**Remark 4.4.1.** We proved that the sequence $(\theta^{(m)})$ converges to the MLE $\hat{\theta}$. As each $\phi_{jk}^{(m)}$, $j = 1, \ldots, r$, $k = 1, \ldots, s$, is the image of $\theta^{(m)}$ by the continuous mapping

$$G_{j,k}(\theta) = \frac{1}{\displaystyle\sum_{m=1}^{r} \frac{x_{+mk}}{1+\theta z_{mk}}} \, \frac{x_{+jk}}{(1+\theta z_{jk})},$$

the sequence $(\phi_{jk}^{(m)})$ also has a limit that is $G_{j,k}(\hat{\theta}) = \hat{\phi}_{j,k}$. Thus, the vector $\boldsymbol{\beta}^{(m)} = (\theta^{(m)}, \boldsymbol{\phi}^{(m)})$ converges to the MLE $\hat{\boldsymbol{\beta}} = (\hat{\theta}, \hat{\boldsymbol{\phi}})$. Theorem 4.4.1 is thus proved.

### Proof of the ascent property

As in Chapter 2, the partition of the parameter vector $\boldsymbol{\beta}$ into two sub-parameters $(\theta, \boldsymbol{\phi})$ and the link between them allow us to consider the concentrated (or profile) likelihood function that is the likelihood function $\ell(\boldsymbol{\beta}) = \ell(\theta, \boldsymbol{\phi})$ re-written as a function only of $\theta$.

**Lemma 4.4.4.** *The concentrated (or profile) likelihood function is defined up to an additive constant by*

$$\ell_c(\theta) = x_{2++} \log \theta - \sum_{k=1}^{s} \sum_{j=1}^{r} x_{+jk} \log(1 + \theta z_{jk}) \tag{4.30}$$

*where* $x_{2++} = \sum_{k=1}^{s} \sum_{j=1}^{r} x_{2jk}$.

**Proof**. The expression (4.6) is equivalent to

$$\ell(\boldsymbol{\beta}) = \sum_{k=1}^{s} \sum_{j=1}^{r} x_{+jk} \log(\phi_{jk}) + x_{2++} \log(\theta) - \sum_{k=1}^{s} n_k \log\left(1 + \theta \langle \mathbf{Z}_k, \boldsymbol{\phi}_k \rangle\right)$$

and the relationships (4.15) and (4.22) enable us to write

$$\begin{aligned} \ell_c(\theta) = \sum_{k=1}^{s} \sum_{j=1}^{r} x_{+jk} \log\left(\frac{x_{+jk}}{1 + \hat{\theta} z_{jk}}\right) - \sum_{k=1}^{s} \sum_{j=1}^{r} x_{+jk} \log\left(\sum_{m=1}^{r} \frac{x_{+mk}}{1 + \hat{\theta} z_{mk}}\right) \\ + x_{2++} \log(\theta) - \sum_{k=1}^{s} n_k \log\left(n_k \Big/ \sum_{m=1}^{r} \frac{x_{+mk}}{1 + \hat{\theta} z_{mk}}\right). \end{aligned}$$

After some manipulations on the second and the fourth terms, we get:

$$\begin{aligned} \ell_c(\theta) = \sum_{k=1}^{s} \sum_{j=1}^{r} x_{+jk} \log\left(\frac{x_{+jk}}{1 + \hat{\theta} z_{jk}}\right) - \sum_{k=1}^{s} n_k \log\left(\sum_{m=1}^{r} \frac{x_{+mk}}{1 + \hat{\theta} z_{mk}}\right) + x_{2++} \log(\theta) \\ - \sum_{k=1}^{s} n_k \log n_k + \sum_{k=1}^{s} n_k \log\left(\sum_{m=1}^{r} \frac{x_{+mk}}{1 + \hat{\theta} z_{mk}}\right). \end{aligned}$$

Removing the second and the fifth terms and the constants, one gets the expression (4.30). □

We may now proceed to the proof of Theorem 4.4.2.

### Proof of Theorem 4.4.2

The profile log-likelihood $\ell_c(\theta)$ is differentiable for every $\theta > 0$ and its derivative is

$$\begin{aligned}\ell_c'(\theta) &= \frac{x_{2++}}{\theta} - \sum_{k=1}^{s}\sum_{j=1}^{r}\frac{x_{+jk}z_{jk}}{1+\theta z_{jk}}\\ &= \frac{1}{\theta}\left(x_{2++} - \sum_{k=1}^{s}\sum_{j=1}^{r}\left(x_{+jk} - \frac{x_{+jk}}{1+\theta z_{jk}}\right)\right)\\ &= \frac{1}{\theta}\left(x_{2++} - n + \sum_{k=1}^{s}\sum_{j=1}^{r}\frac{x_{+jk}}{1+\theta z_{jk}}\right).\end{aligned}$$

Since $x_{1++} + x_{2++} = n$, we have

$$\ell_c'(\theta) = \theta^{-1}\ \psi(\theta)$$

where the function $\psi$ is defined by Equation (4.21). From Lemma 4.4.1, we deduce that

$$\forall\theta \leqslant \theta^*, \quad \ell_c'(\theta) \geqslant 0 \quad \text{and} \quad \forall\theta \geqslant \theta^*, \quad \ell_c'(\theta) \leqslant 0$$

where $\theta^*$ is the MLE of $\theta$ and also the unique root of $\psi$. Hence the function $\ell_c$ is increasing on the interval $]0, \theta^*]$ and decreasing on $[\theta^*, +\infty[$. To finish the proof, we consider the two cases $\theta^{(0)} < \theta^*$ and $\theta^{(0)} > \theta^*$.

- If $\theta^{(0)} \leqslant \theta^*$, then one can prove by induction that

  $$\theta^{(m)} \leqslant \theta^*, \quad m = 0, 1, \dots$$

  Indeed the statement holds for $m = 0$ by assumption. Moreover, if the statement holds for some $m > 0$ then

  $$\theta^{(m+1)} = \varphi(\theta^{(m)}) \leqslant \theta^* = \varphi(\theta^*)$$

  because $\varphi$ is an increasing function and $\theta^*$ is a fixed point of $\varphi$. Therefore the statement also holds for $m+1$ and the proof is complete.

  We have proved that the sequence $\theta^{(m)}$ is increasing and still belongs to the interval $]0, \theta^*]$. Then $\theta^{(m)} \leqslant \theta^{(m+1)}$ and $\ell_c(\theta^{(m)}) \leqslant \ell_c(\theta^{(m+1)})$ because $\ell_c$ is increasing on $]0, \theta^*]$.

- If $\theta^{(0)} \geqslant \theta^*$, then using similar arguments to those above, one can prove by induction that

  $$\theta^{(m)} \geqslant \theta^*, \quad m = 0, 1, \dots$$

  The sequence $\theta^{(m)}$ is decreasing and still belongs to the interval $[\theta^*, +\infty[$. Then $\theta^{(m+1)} \leqslant \theta^{(m)}$ and $\ell_c(\theta^{(m+1)}) \geqslant \ell_c(\theta^{(m)})$ because $\ell_c$ is decreasing on $[\theta^*, +\infty[$.

In all the cases, we have

$$\ell(\boldsymbol{\beta}^{(m)}) = \ell_c(\theta^{(m)}) \leqslant \ell_c(\theta^{(m+1)}) = \ell(\boldsymbol{\beta}^{(m+1)})$$

and the proof of Theorem 4.4.2 is complete. □

## 4.5 Numerical experiments

In this section, we compare our CA with standard constrained optimization routines available on R and MATLAB software in terms of accuracy, robustness and computation time. The methods selected for comparison on R software are the MM algorithm, the Newton-Raphson's, the quasi-newton BFGS algorithm [7, 25, 27, 93] and derivative-free Nelder-Mead's [67] (denoted NM) while on MATLAB the CA is compared to the Interior Point [103] (denoted IP). The BFGS and NM algorithms are implemented using the `constrOptim.nl` function of the R package `alabama` by Varadhan [100]. The MATLAB function `fmincon` of the Optimization Toolbox is used to implement the IP procedure.

### Implementation of the MM algorithm

We code the MM algorithm in R software using Algorithm 4.2 proposed by Mkhadri et al. [65].

**Algorithm 4.2** Implementation of the MM algorithm [65]

The MM udpates are given by

$$\theta^{(m+1)} = \frac{ws\ \theta^{(m)} + \sum_{k=1}^{s}\sum_{j=1}^{r} x_{2jk}}{ws + \sum_{k=1}^{s} n_k a_k^{(m)} \sum_{k=1}^{s}\sum_{j=1}^{r} z_{jk}\phi_{jk}^{(m)}}$$

and

$$\phi_{jk}^{(m+1)} = \frac{(x_{1jk} + x_{2jk}) + w\phi_{jk}^{(m)}}{ws + n_k + n_k a_k^{(m)} \theta^{(m+1)} \left(z_{jk} - \sum_{i=1}^{r} z_{ik}\phi_{ik}^{(m)}\right)}, \quad j = 1, \ldots, r; \quad k = 1, \ldots, s$$

where

$$a_k^{(m)} = \frac{1}{1 + \theta^{(m)} \sum_{i=1}^{r} z_{ik}\phi_{ik}^{(m)}}, \quad k = 1, \ldots, s$$

and $w$ is a non-negative tuning parameter.

### Implementation of the Newton's algorithm

We code the NR algorithm in R software. As in Chapter 2, to ensure that the positivity constraints existing on the parameters are respected, we opt for the following reparameterization:

$$(\theta, \phi_{11}, \ldots, \phi_{r1}, \ldots, \phi_{1s}, \ldots, \phi_{rs}) = (\exp(\alpha), \exp(\eta_{11}), \ldots, \exp(\eta_{r1}), \ldots, \exp(\eta_{1s}), \ldots, \exp(\eta_{rs})). \quad (4.31)$$

The new vector parameter is

$$\boldsymbol{\beta}_{\text{new}} = (\alpha, \eta_{11}, \ldots, \eta_{r1}, \ldots, \eta_{1s}, \ldots, \eta_{rs}) \in \mathbb{R}^{1+sr}.$$

This latter is no more subject to inequality constraints since its components can take any value in the set of real numbers $\mathbb{R}$. The non-linear system (4.8) may be rewritten

$$F(\boldsymbol{\beta}_{\text{new}}) = \mathbf{0} \tag{4.32}$$

where the function $F$ is defined from $\mathbb{R}^{1+sr}$ in itself by

$$F(\boldsymbol{\beta}_{\text{new}}) = \left(F_0(\boldsymbol{\beta}_{\text{new}}), F_{11}(\boldsymbol{\beta}_{\text{new}}), \ldots, F_{r1}(\boldsymbol{\beta}_{\text{new}}), \ldots, F_{1s}(\boldsymbol{\beta}_{\text{new}}), \ldots, F_{rs}(\boldsymbol{\beta}_{\text{new}})\right)^T$$

and

$$\begin{aligned} F_0(\boldsymbol{\beta}_{\text{new}}) &= \sum_{k=1}^{s} \frac{n_k}{1 + \mathrm{e}^{\alpha} \sum_{i=1}^{r} z_{ik} \mathrm{e}^{\eta_{ik}}} - \sum_{k=1}^{s} x_{1+k} \\ F_{jk}(\boldsymbol{\beta}_{\text{new}}) &= (x_{+jk}) \left(1 + \mathrm{e}^{\alpha} \sum_{m=1}^{r} z_{mk} \mathrm{e}^{\eta_{mk}}\right) - n_k \mathrm{e}^{\eta_{jk}} \left(1 + \mathrm{e}^{\alpha} z_{jk}\right), \\ & j = 1, \ldots, r; \quad k = 1, \ldots, s. \end{aligned} \tag{4.33}$$

**Theorem 4.5.1.** *The Newton-Raphson's method applied to the estimation of $\beta_{\text{new}}$ corresponds to the iterative scheme*

$$-\left[\mathbf{J}(\boldsymbol{\beta}_{\text{new}}^{(m)})\right] \mathbf{p}^{(m)} = F(\boldsymbol{\beta}_{\text{new}}^{(m)}) \tag{4.34}$$

$$\boldsymbol{\beta}_{\text{new}}^{(m+1)} = \boldsymbol{\beta}_{\text{new}}^{(m)} + \mathbf{p}^{(m)} \tag{4.35}$$

*where*

$$\mathbf{J}(\boldsymbol{\beta}_{\text{new}}) = \begin{pmatrix} J_{0,0} & \mathbf{J}_{0,1} & \cdots & \cdots & \mathbf{J}_{0,s} \\ \mathbf{J}_{1,0} & \mathbf{J}_{1,1} & \mathbf{0}_{r,r} & \cdots & \mathbf{0}_{r,r} \\ \vdots & \mathbf{0}_{r,r} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0}_{r,r} \\ \mathbf{J}_{s,0} & \mathbf{0}_{r,r} & \cdots & \mathbf{0}_{r,r} & \mathbf{J}_{s,s} \end{pmatrix} \in \mathbb{R}^{(1+sr)\times(1+sr)},$$

*$\mathbf{0}_{r,r}$ is a the $r \times r$ matrix full of zeros,*

$$\begin{aligned} J_{0,0} &= -\sum_{k=1}^{s} \frac{n_k \mathrm{e}^{\alpha} \langle \mathbf{Z}_k, \boldsymbol{\phi}_k(\boldsymbol{\eta}_k)\rangle}{\left(1 + \mathrm{e}^{\alpha} \langle \mathbf{Z}_k, \boldsymbol{\phi}_k(\boldsymbol{\eta}_k)\rangle\right)^2}, \\ \mathbf{J}_{0,k} &= -n_k \mathrm{e}^{\alpha} \left(1 + \mathrm{e}^{\alpha} \langle \mathbf{Z}_k, \boldsymbol{\phi}_k(\boldsymbol{\eta}_k)\rangle\right)^{-2} \mathbf{Z}_k \odot \boldsymbol{\phi}_k(\boldsymbol{\eta}_k), \qquad k = 1, \ldots, s, \\ \mathbf{J}_{k,0} &= \mathrm{e}^{\alpha} \left(\langle \mathbf{Z}_k, \boldsymbol{\phi}_k(\boldsymbol{\eta}_k)\rangle \mathbf{X}_{+k} - n_k \mathbf{Z}_k \odot \boldsymbol{\phi}_k(\boldsymbol{\eta}_k)\right), \qquad k = 1, \ldots, s, \end{aligned}$$

*$\boldsymbol{\phi}_k(\boldsymbol{\eta}_k) = (\mathrm{e}^{\eta_{1k}}, \ldots, \mathrm{e}^{\eta_{rk}})^T$ for every $k = 1, \ldots, s$,, $\mathbf{J}_{k,k}$ is a $r \times r$ matrix whose components are*

$$(\mathbf{J}_{k,k})_{ij} = \mathrm{e}^{\eta_{jk}} \left(x_{+ik}\ \mathrm{e}^{\alpha} z_{jk} - n_k(1 + \mathrm{e}^{\alpha} z_{ik})\ \delta_i^j\right), \qquad i, j = 1, \ldots, r,$$

*$\odot$ represents the Hadamard product of matrices and $\delta_i^j$ represents the classical Kronecker's symbol.*

**Proof**. The matrix $\mathbf{J}$ is the Jacobian matrix of $F$. Thus its components are obtained as follows:

$$J_{00} = \frac{\partial F_0}{\partial \alpha},$$

$\mathbf{J}_{0,k}$ and $\mathbf{J}_{k,0}$ are matrices of dimension $r \times 1$ whose components are respectively

$$(\mathbf{J}_{0,k})_i = \frac{\partial F_0}{\partial \eta_{ik}}, \quad (\mathbf{J}_{k,0})_i = \frac{\partial F_{ik}}{\partial \alpha} \quad j = 1, \ldots, r$$

and

$$(\mathbf{J}_{k,k})_{ij} = \frac{\partial F_{ik}}{\partial \eta_{jk}}, \qquad i, j = 1, \ldots, r.$$

□

**Comparison criteria for the selected algorithms**

As in Chapter 2, three pertinent criteria are used for adequate comparison of the algorithms: robustness (algorithms should perform well for all reasonable choices of the initial solution), accuracy (algorithms should be able to identify a solution near the true values with precision) and efficiency (algorithms should not require too much computation time or storage). The accuracy is measured through the Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{1+sr}\left((\hat{\theta} - \theta^0)^2 + \sum_{k=1}^{s}\sum_{j=1}^{r}(\hat{\phi}_{jk} - \phi^0_{jk})^2\right)$$

and the robustness is measured through the number of iterations. It is known that the larger is the difference between the minimum and maximum numbers of iterations, the greater is the dependence of the algorithm towards to the initial solution. The efficiency is examined through the computation time.

### 4.5.1 Data generation principle

The simulated data are presented under four scenarios described below:

1. In Scenario 1, $s = 5$, $r = 3$ and the true parameter $\boldsymbol{\beta}^0$ is given by $\theta^0 = 0.8$ and

$$\begin{aligned}\boldsymbol{\phi}_1^0 &= (0.80, 0.15, 0.05),\\ \boldsymbol{\phi}_2^0 &= (0.10, 0.30, 0.60),\\ \boldsymbol{\phi}_3^0 &= (0.35, 0.30, 0.35),\\ \boldsymbol{\phi}_4^0 &= (0.70, 0.20, 0.10),\\ \boldsymbol{\phi}_5^0 &= (0.30, 0.40, 0.30).\end{aligned}$$

2. In Scenario 2, $s = 5$, $r = 10$ and the true parameter $\boldsymbol{\beta}^0$ is given by $\theta^0 = 0.8$ and

$$\boldsymbol{\phi}_i^0 = (0.4, 0.1, 0.05, \underbrace{0.1}_{2}, \underbrace{0.05}_{5}), \quad i \in \{1, 5\};$$

$$\boldsymbol{\phi}_i^0 = (\underbrace{0.1}_{3}, \underbrace{0.05}_{2}, 0.10, 0.25, \underbrace{0.05}_{2}, 0.15), \quad i \in \{2, 4\};$$

$$\boldsymbol{\phi}_i^0 = (\underbrace{0.1, \ldots, 0.1}_{10}), \quad i \in \{3\}.$$

3. In Scenario 3, $s = 10$, $r = 10$ and the true parameter $\boldsymbol{\beta}^0$ is given by $\theta^0 = 0.8$ and

$$\boldsymbol{\phi}_i^0 = (0.4, 0.1, 0.05, \underbrace{0.1}_{2}, \underbrace{0.05}_{5}), \quad i \in \{1, 5, 7, 10\};$$

$$\boldsymbol{\phi}_i^0 = (\underbrace{0.1}_{3}, \underbrace{0.05}_{2}, 0.10, 0.25, \underbrace{0.05}_{2}, 0.15), \quad i \in \{2, 3, 6\};$$

$$\boldsymbol{\phi}_i^0 = (\underbrace{0.1, \ldots, 0.1}_{10}), \quad i \in \{4, 8, 9\};$$

4. In Scenario 4, $s = 20$, $r = 10$ and the true parameter $\boldsymbol{\beta}^0$ is given by $\theta^0 = 0.8$ and

$$\boldsymbol{\phi}_i^0 = (0.4, 0.1, 0.05, \underbrace{0.1}_{2}, \underbrace{0.05}_{5}), \quad i \in \{1, 5, 7, 10, 11, 15, 17, 20\};$$

$$\boldsymbol{\phi}_i^0 = (\underbrace{0.1}_{3}, \underbrace{0.05}_{2}, 0.10, 0.25, \underbrace{0.05}_{2}, 0.15), \quad i \in \{2, 3, 6, 12, 13, 16\};$$

$$\boldsymbol{\phi}_i^0 = (\underbrace{0.1, \ldots, 0.1}_{10}), \quad i \in \{4, 8, 9, 14, 18, 19\};$$

Scenario 1 was chosen in order to apply the algorithms for low values of $s$ and $r$. Scenarios 2 to 4 were chosen in order to obtain low values in the different accident classes. Indeed, for $r = 10$ and low values of $n_k$ (50 for example), the multinomial sampling of $n_k$ accidents into $2r = 20$ classes will enable to have low values for $x_{ijk}$. This latter situation can be encountered in practice.

For each scenario, two values were given to $n_k$: 50 (low value) and 5000 (great value).

One knows that the performance of iterative algorithms usually relies on the initial solution. In order to explore a plethora of initial solutions in the parameter space and study the impact of the initialization strategy, we have considered four different initialization schemes for setting the starting parameter vector $\boldsymbol{\beta}^{(0)} = (\theta^{(0)}, (\boldsymbol{\phi}^{(0)}))^T$. The parameter $\theta^{(0)}$ is randomly generated from an uniform distribution and the parameter vector $\boldsymbol{\phi}^{(0)}$ is randomly generated in four different ways:

1. In Initialization 1, $\boldsymbol{\phi}_k^{(0)} = \left(\frac{1}{r}, \cdots, \frac{1}{r}\right)^T$.
2. In Initialization 2,
$$\boldsymbol{\phi}_k^{(0)} = \left(\frac{x_{11k} + x_{21k}}{n_k}, \ \ldots \ , \frac{x_{1rk} + x_{2rk}}{n_k}\right)^T.$$
3. In Initialization 3, $\boldsymbol{\phi}_k^{(0)} = \mathbf{U}_k / \sum_{j=1}^r u_{jk}$ where $\mathbf{U}_k = (u_{1k}, \ldots, u_{rk})^T$ is a $r-$dimensional vector whose components are randomly generated from an uniform distribution $\mathcal{U}[0.05; 0.95]$.
4. In Initialization 4,
$$\boldsymbol{\phi}_k^{(0)} = (x_{1+k})^{-1} (x_{11k}, x_{12k}, \ldots, x_{1rk})^T$$
where $x_{1+k} = \sum_{j=1}^r x_{1jk}$. This setting of $\boldsymbol{\phi}_k^{(0)}$ is suggested by the assumptions made in N'Guessan et al. [73] in building the multinomial model.

**Remark 4.5.1.** The different initialization schemes used in this numerical experiments are similar to the ones used in Chapter 2. The first one is quite a logical way to initialize a vector of class probabilities whose sum is equal to 1. The second one corresponds to the starting point used by N'Guessan and Truffier [78]. The fourth one corresponds to a natural initialization of $\boldsymbol{\phi}_k$ by construction of the model (4.4). At last, the third one corresponds to the general case where the vector $\boldsymbol{\phi}_k^{(0)}$ is randomly chosen.

### 4.5.2 Analysis of the results

Table 4.2. Results for Scenario 1 ($s = 5$ and $r = 3$) and $n_k = 50$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 186 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 11 | 3 | 9 | 8 | 2 | 19 |
| Max iter. | 4 | 35 | 15 | 12 | 20 | 4 | 62 |
| Mean iter. | 3,0 | 24,0 | 5,6 | 10,6 | 13,6 | 3,1 | 28,0 |
| Time | 9E-03 | 2E-02 | 2E-02 | 1E+00 | 9E-01 | 7E-03 | 3E-01 |
| Time ratio | 1,0 | 2,3 | 2,4 | 126,9 | 103,6 | 1,0 | 44,6 |
| MSE | 4,2E-03 | 4,2E-03 | 4,1E-03 | 4,2E-03 | 1,2E-01 | 4,2E-03 | 4,2E-03 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 191 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 16 | 3 | 9 | 9 | 2 | 19 |
| Max iter. | 4 | 32 | 15 | 11 | 20 | 4 | 65 |
| Mean iter. | 3,1 | 23,2 | 5,6 | 10,6 | 13,9 | 3,1 | 27,2 |
| Time | 9E-03 | 2E-02 | 2E-02 | 1E+00 | 9E-01 | 7E-03 | 3E-01 |
| Time ratio | 1,0 | 2,3 | 2,3 | 122,9 | 100,9 | 1,0 | 49,4 |
| MSE | 4,1E-03 | 4,1E-03 | 4,3E-03 | 4,1E-03 | 1,2E-01 | 4,0E-03 | 4,0E-03 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 202 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 14 | 4 | 1 | 1 | 2 | 21 |
| Max iter. | 4 | 36 | 13 | 11 | 20 | 4 | 60 |
| Mean iter. | 3,1 | 24,8 | 5,7 | 6,4 | 8,2 | 3,1 | 30,2 |
| Time | 9E-03 | 2E-02 | 2E-02 | 7E-01 | 6E-01 | 7E-03 | 3E-01 |
| Time ratio | 1,0 | 2,5 | 2,2 | 72,3 | 60,8 | 1,0 | 41,2 |
| MSE | 4,0E-03 | 4,0E-03 | 4,0E-03 | 6,2E-02 | 1,2E-01 | 3,9E-03 | 3,9E-03 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 210 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 7 | 3 | 1 | 1 | 2 | 17 |
| Max iter. | 3 | 32 | 17 | 12 | 22 | 3 | 66 |
| Mean iter. | 2,8 | 21,3 | 5,3 | 10,4 | 13,3 | 2,8 | 27,3 |
| Time | 9E-03 | 2E-02 | 2E-02 | 1E+00 | 9E-01 | 6E-03 | 3E-01 |
| Time ratio | 1,0 | 2,3 | 2,3 | 122,4 | 98,6 | 1,0 | 50,9 |
| MSE | 4,2E-03 | 4,2E-03 | 4,3E-03 | 5,1E-03 | 1,1E-01 | 4,0E-03 | 4,0E-03 |

The results presented below correspond to the average parameters estimates calculated on 250 replications. Tables 4.2 to 4.9 show the results obtained for the different scenarios. In these tables, CPU times are given in seconds and are linked to CPU time ratios calculated as the ratio between the mean CPU time of a given algorithm and the mean duration of the cyclic algorithm. Thus, the CPU time ratio of the cyclic algorithm is always equal

Table 4.3. Results for Scenario 1 ($s = 5$ and $r = 3$) and $n_k = 5000$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 190 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 17 | 3 | 13 | 11 | 2 | 24 |
| Max iter. | 5 | 40 | 20 | 15 | 23 | 5 | 55 |
| Mean iter. | 3,7 | 33,0 | 5,7 | 14,9 | 16,1 | 3,7 | 30,3 |
| Time | 1E-02 | 3E-02 | 2E-02 | 2E+00 | 1E+00 | 9E-03 | 4E-01 |
| Time ratio | 1,0 | 2,2 | 1,6 | 179,9 | 72,3 | 1,0 | 43,7 |
| MSE | 4,2E-05 | 4,2E-05 | 4,3E-05 | 4,2E-05 | 2,1E-01 | 4,1E-05 | 4,1E-05 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 196 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 24 | 3 | 1 | 1 | 3 | 18 |
| Max iter. | 5 | 38 | 14 | 15 | 23 | 5 | 47 |
| Mean iter. | 3,8 | 31,8 | 5,8 | 14,1 | 15,5 | 3,8 | 27,2 |
| Time | 1E-02 | 3E-02 | 2E-02 | 2E+00 | 9E-01 | 9E-03 | 4E-01 |
| Time ratio | 1,0 | 2,1 | 1,6 | 165,9 | 67,6 | 1,0 | 44,8 |
| MSE | 4,0E-05 | 4,0E-05 | 4,0E-05 | 5,1E-03 | 2,5E-01 | 3,9E-05 | 3,9E-05 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 196 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 22 | 4 | 1 | 1 | 2 | 25 |
| Max iter. | 5 | 40 | 24 | 15 | 22 | 5 | 56 |
| Mean iter. | 3,8 | 33,1 | 6,1 | 7,8 | 9,1 | 3,8 | 31,2 |
| Time | 1E-02 | 3E-02 | 2E-02 | 1E+00 | 6E-01 | 9E-03 | 4E-01 |
| Time ratio | 1,0 | 2,3 | 1,7 | 93,6 | 42,2 | 1,0 | 40,6 |
| MSE | 4,1E-05 | 4,1E-05 | 4,1E-05 | 7,1E-02 | 1,6E-01 | 4,1E-05 | 4,1E-05 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 185 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 11 | 3 | 1 | 1 | 2 | 16 |
| Max iter. | 3 | 34 | 21 | 15 | 21 | 4 | 69 |
| Mean iter. | 2,8 | 28,0 | 5,9 | 13,8 | 15,1 | 2,8 | 25,6 |
| Time | 1E-02 | 3E-02 | 2E-02 | 2E+00 | 9E-01 | 7E-03 | 4E-01 |
| Time ratio | 1,0 | 2,5 | 2,0 | 206,5 | 85,6 | 1,0 | 58,8 |
| MSE | 4,3E-05 | 4,3E-05 | 4,4E-05 | 9,5E-03 | 2,4E-01 | 4,1E-05 | 4,1E-05 |

to 1.

The first step of our analysis focuses on the influence of the initialization scheme through the number of iterations. First, it can be noticed that except NR algorithm all the others have always converged. On average, NR converged 166 to 210 times over 250 repetitions. This is a general trend observed for all the scenarios and this is not surprising as it is well known that NR may converge to bad values or may not converge at all if started far from the true parameter. In the results, the mean values are calculated only when there has been convergence of the NR algorithm. Secondly, it can be seen that on the overall 8000 repetitions of the ML estimation, the CA has only needed between 2 and 5 iterations. The BFGS, NM, NR and MM algorithms look much more sensible to the initial solution than the CA as their iterations number vary respectively between 1 and 29, 1 and 27, 3 and 31, 7 and 40. The IP algorithm's performance in terms of number of iterations is by far the worse since the number of iterations varies between 16 and 390. Thirdly, a look at the mean

Table 4.4. Results for Scenario 2 ($s = 5$ and $r = 10$) and $n_k = 50$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 178 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 9 | 3 | 9 | 4 | 2 | 29 |
| Max iter. | 4 | 27 | 22 | 13 | 23 | 4 | 101 |
| Mean iter. | 3,1 | 21,5 | 5,7 | 10,6 | 5,6 | 3,1 | 45,7 |
| Time | 1E-02 | 4E-02 | 7E-02 | 5E+00 | 2E+00 | 7E-03 | 1E+00 |
| Time ratio | 1,0 | 3,6 | 6,5 | 515,1 | 179,2 | 1,0 | 159,9 |
| MSE | 1,7E-03 | 1,7E-03 | 1,7E-03 | 1,7E-03 | 1,5E-02 | 1,7E-03 | 1,7E-03 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 177 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 18 | 3 | 9 | 3 | 3 | 26 |
| Max iter. | 4 | 29 | 17 | 13 | 18 | 4 | 149 |
| Mean iter. | 3,3 | 23,6 | 5,7 | 10,6 | 5,3 | 3,4 | 41,8 |
| Time | 1E-02 | 4E-02 | 7E-02 | 5E+00 | 2E+00 | 7E-03 | 1E+00 |
| Time ratio | 1,0 | 3,8 | 6,2 | 482,5 | 169,3 | 1,0 | 137,0 |
| MSE | 1,7E-03 | 1,7E-03 | 1,7E-03 | 1,7E-03 | 6,0E-03 | 1,7E-03 | 1,7E-03 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 187 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 13 | 3 | 1 | 1 | 2 | 34 |
| Max iter. | 4 | 28 | 25 | 12 | 22 | 4 | 112 |
| Mean iter. | 3,1 | 22,2 | 5,8 | 5,3 | 4,1 | 3,1 | 64,0 |
| Time | 1E-02 | 4E-02 | 7E-02 | 3E+00 | 1E+00 | 7E-03 | 1E+00 |
| Time ratio | 1,0 | 4,0 | 7,0 | 301,9 | 149,9 | 1,0 | 212,8 |
| MSE | 1,7E-03 | 1,7E-03 | 1,7E-03 | 1,8E-02 | 2,3E-02 | 1,7E-03 | 1,7E-03 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 194 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 12 | 3 | 9 | 3 | 2 | 25 |
| Max iter. | 4 | 28 | 15 | 13 | 19 | 4 | 118 |
| Mean iter. | 3,0 | 21,3 | 5,4 | 10,6 | 5,5 | 2,9 | 41,9 |
| Time | 9E-03 | 4E-02 | 6E-02 | 5E+00 | 2E+00 | 7E-03 | 1E+00 |
| Time ratio | 1,0 | 4,2 | 7,3 | 601,1 | 213,1 | 1,0 | 156,8 |
| MSE | 1,8E-03 | 1,8E-03 | 1,8E-03 | 1,8E-03 | 6,5E-03 | 1,7E-03 | 1,7E-03 |

number of iterations for different scenarios and initialization schemes shows that the mean number of iterations doesn't vary too much for CA, MM, NR. But this cannot be said of BFGS, NM and IP. The mean number of iterations of the latter three vary with the type of initialization of the solution. For example, in tables 4.2, 4.4, 4.6 and 4.8, BFGS took on average 10 to 11 iterations except for Initialization 3 where the number of iterations is significantly lower. The same remark on Initialization 3 applies to Nelder-Mead's algorithm.

Since the main goal of any estimation algorithm is to deliver a solution near the true value, we analyse, in this second step, the MSE. A first trend for all the algorithms is that the MSE decreases when the sample size increases. Now considering all the MSE, CA, MM, NR and IP are all competitive as they have the same MSE. But the MSE of BFGS algorithm are 10 times greater than those of the four algorithms (CA, MM, NR and IP) for initialization 3 ($n_k = 50$) and upto 1000 times greater for initialization 3 with $n_k = 5000$. The NM algorithm is the least competitive of the selected algorithms since its MSE are

Table 4.5. Results for Scenario 2 ($s = 5$ and $r = 10$) and $n_k = 5000$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 183 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 19 | 4 | 14 | 5 | 2 | 37 |
| Max iter. | 5 | 37 | 20 | 15 | 26 | 5 | 177 |
| Mean iter. | 3,8 | 30,7 | 6,0 | 14,9 | 7,4 | 3,8 | 59,7 |
| Time | 1E-02 | 5E-02 | 7E-02 | 8E+00 | 2E+00 | 9E-03 | 1E+00 |
| Time ratio | 1,0 | 4,2 | 5,6 | 610,4 | 149,8 | 1,0 | 159,5 |
| MSE | 1,9E-05 | 1,9E-05 | 1,9E-05 | 1,9E-05 | 8,1E-03 | 2,0E-05 | 2,0E-05 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 166 | 250 | 250 | 250 | 250 |
| Min iter. | 4 | 29 | 4 | 13 | 5 | 4 | 20 |
| Max iter. | 5 | 36 | 12 | 15 | 17 | 5 | 128 |
| Mean iter. | 4,0 | 32,2 | 5,7 | 14,9 | 6,4 | 4,0 | 47,8 |
| Time | 1E-02 | 6E-02 | 7E-02 | 8E+00 | 2E+00 | 1E-02 | 1E+00 |
| Time ratio | 1,0 | 4,3 | 5,2 | 597,8 | 148,2 | 1,0 | 125,6 |
| MSE | 2,0E-05 | 2,0E-05 | 2,0E-05 | 2,0E-05 | 2,0E-03 | 1,9E-05 | 1,9E-05 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 200 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 21 | 4 | 1 | 1 | 3 | 41 |
| Max iter. | 5 | 36 | 14 | 15 | 24 | 5 | 178 |
| Mean iter. | 3,8 | 30,8 | 5,6 | 6,9 | 5,1 | 3,9 | 89,8 |
| Time | 1E-02 | 5E-02 | 7E-02 | 4E+00 | 1E+00 | 9E-03 | 2E+00 |
| Time ratio | 1,0 | 4,2 | 5,3 | 301,2 | 115,1 | 1,0 | 220,7 |
| MSE | 1,9E-05 | 1,9E-05 | 1,9E-05 | 1,5E-02 | 1,9E-02 | 2,0E-05 | 2,0E-05 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 182 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 11 | 3 | 12 | 5 | 2 | 19 |
| Max iter. | 4 | 33 | 15 | 15 | 18 | 4 | 142 |
| Mean iter. | 2,9 | 26,9 | 5,6 | 14,9 | 6,7 | 2,9 | 39,1 |
| Time | 1E-02 | 5E-02 | 7E-02 | 8E+00 | 2E+00 | 8E-03 | 1E+00 |
| Time ratio | 1,0 | 4,2 | 5,5 | 639,9 | 159,4 | 1,0 | 137,7 |
| MSE | 2,0E-05 | 2,0E-05 | 2,0E-05 | 2,0E-05 | 1,5E-03 | 2,0E-05 | 2,0E-05 |

very often the greatest and upto 10 000 times greater (Table 4.3).

The third step of our analysis is dedicated to the CPU time. First, it is noticed that the CPU time of CA is always around $10^{-2}$ seconds during the 8000 repetitions. Now we focus on the CPU time ratios that allow to make a comparison with the CA. It is noticed that none of the CPU ratios is lower than 1 which means none of the other algorithms is faster than the CA. The MM and NR algorithms seem the most competitive with the CA as they rank respectively second and third as the computation time is concerned. The CA is on average 2 to 13 times faster than MM and 2 to 63 times quicker than NR. The other algorithms seem less competitive. Indeed, the CA is upto 1358 (resp. 2108, resp. 5300) times quicker than IP (resp. NM, resp. BFGS).

In a fourth and last step we analyse the impact of the dimension of the parameter space on the performance of the different algorithms. This is done by analysing the relative CPU

Table 4.6. Results for Scenario 3 ($s = 10$ and $r = 10$) and $n_k = 50$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 181 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 9 | 3 | 9 | 4 | 2 | 29 |
| Max iter. | 4 | 28 | 16 | 13 | 25 | 4 | 193 |
| Mean iter. | 3,1 | 22,3 | 5,4 | 11,0 | 5,9 | 3,1 | 63,6 |
| Time | 1E-02 | 7E-02 | 2E-01 | 2E+01 | 6E+00 | 8E-03 | 3E+00 |
| Time ratio | 1,0 | 6,3 | 18,8 | 1470,0 | 597,7 | 1,0 | 342,3 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,4E-02 | 1,6E-03 | 1,6E-03 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 173 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 21 | 3 | 9 | 3 | 3 | 27 |
| Max iter. | 4 | 28 | 16 | 13 | 18 | 4 | 229 |
| Mean iter. | 3,7 | 24,9 | 5,4 | 10,8 | 5,3 | 3,7 | 54,6 |
| Time | 1E-02 | 7E-02 | 2E-01 | 2E+01 | 6E+00 | 9E-03 | 2E+00 |
| Time ratio | 1,0 | 6,1 | 16,1 | 1258,4 | 499,1 | 1,0 | 263,5 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,3E-02 | 1,6E-03 | 1,6E-03 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 184 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 12 | 3 | 1 | 1 | 2 | 40 |
| Max iter. | 4 | 29 | 25 | 13 | 23 | 4 | 296 |
| Mean iter. | 3,1 | 22,8 | 5,7 | 2,9 | 3,5 | 3,1 | 93,3 |
| Time | 1E-02 | 7E-02 | 2E-01 | 6E+00 | 4E+00 | 8E-03 | 4E+00 |
| Time ratio | 1,0 | 6,8 | 20,9 | 605,7 | 447,2 | 1,0 | 508,5 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,5E-02 | 1,9E-02 | 1,6E-03 | 1,6E-03 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 186 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 13 | 3 | 1 | 3 | 2 | 29 |
| Max iter. | 4 | 27 | 24 | 13 | 17 | 4 | 209 |
| Mean iter. | 3,0 | 22,1 | 5,7 | 10,9 | 4,8 | 3,0 | 55,9 |
| Time | 9E-03 | 6E-02 | 2E-01 | 1E+01 | 6E+00 | 7E-03 | 2E+00 |
| Time ratio | 1,0 | 6,8 | 21,5 | 1583,3 | 635,4 | 1,0 | 323,8 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,4E-02 | 1,6E-03 | 1,6E-03 |

times computed as the ratios of CPU times divided by a reference CPU time. Here the dimension of the parameter space takes the four values $1 + sr \in \{16, 51, 101, 201\}$. For example, if we select the dimension 16 as reference, the relative CPU time of dimension $d$ is computed as:

$$\text{relative CPU time for dimension } d \; = \; \frac{\text{CPU time for dimension } d}{\text{CPU time for dimension } 16}.$$

The relative CPU times for Initialization 3 and $n_k = 50$ are given by Table 4.10 and displayed on Figure 4.2. It is seen that when the dimension of the parameter space increases, the CPU time of the CA remains almost the same. The MM algorithm is still very competitive with the CA but the CPU times of all the other algorithms increase heavily with the dimension of the parameter space. This is quite expected. For example, each step of the NR algorithm requires the inversion of the Jacobian matrix which belongs to $\mathbb{R}^{201\times 201}$ when $s = 20$ and $r = 10$.

Table 4.7. Results for Scenario 3 ($s = 10$ and $r = 10$) and $n_k = 5000$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 187 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 21 | 4 | 12 | 6 | 2 | 51 |
| Max iter. | 5 | 36 | 15 | 16 | 25 | 4 | 242 |
| Mean iter. | 3,8 | 31,9 | 5,9 | 15,1 | 8,0 | 3,8 | 85,8 |
| Time | 1E-02 | 9E-02 | 2E-01 | 2E+01 | 6E+00 | 1E-02 | 3E+00 |
| Time ratio | 1,0 | 7,3 | 16,5 | 1554,1 | 474,9 | 1,0 | 345,9 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,4E-02 | 1,8E-05 | 1,8E-05 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 179 | 250 | 250 | 250 | 250 |
| Min iter. | 4 | 31 | 4 | 1 | 1 | 4 | 32 |
| Max iter. | 5 | 36 | 15 | 16 | 19 | 5 | 201 |
| Mean iter. | 4,0 | 33,6 | 5,7 | 15,0 | 6,7 | 4,0 | 56,3 |
| Time | 1E-02 | 1E-01 | 2E-01 | 2E+01 | 6E+00 | 1E-02 | 2E+00 |
| Time ratio | 1,0 | 7,3 | 15,2 | 1464,9 | 445,2 | 1,0 | 231,3 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 5,8E-05 | 1,2E-02 | 1,8E-05 | 1,8E-05 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 188 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 17 | 4 | 1 | 1 | 3 | 51 |
| Max iter. | 5 | 38 | 31 | 16 | 22 | 5 | 227 |
| Mean iter. | 3,9 | 32,1 | 5,7 | 3,3 | 3,6 | 3,9 | 125,6 |
| Time | 1E-02 | 9E-02 | 2E-01 | 6E+00 | 4E+00 | 1E-02 | 5E+00 |
| Time ratio | 1,0 | 7,2 | 15,6 | 477,8 | 330,4 | 1,0 | 504,2 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,6E-02 | 1,8E-02 | 1,8E-05 | 1,8E-05 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 190 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 17 | 3 | 15 | 4 | 2 | 20 |
| Max iter. | 3 | 34 | 26 | 16 | 20 | 3 | 186 |
| Mean iter. | 2,9 | 28,9 | 5,8 | 15,0 | 7,2 | 2,9 | 42,7 |
| Time | 1E-02 | 8E-02 | 2E-01 | 2E+01 | 6E+00 | 8E-03 | 2E+00 |
| Time ratio | 1,0 | 8,2 | 20,0 | 1926,8 | 583,7 | 1,0 | 235,0 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,2E-02 | 1,8E-05 | 1,8E-05 |

## 4.6 Conclusion

In this chapter, we presented a generalization of the cyclic algorithm introduced in Chapter 2. The results of the numerical experiments performed in this chapter suggest that this generalization of the algorithm is robust towards the starting values, efficient and accurate. We proved that the proposed algorithm is an ascent algorithm that converges to the Maximum Likelihood Estimate from any starting point. Moreover, the comparison of the performance of the cyclic algorithm to some of the best available optimization algorithms like MM and Newton-Raphson's algorithms suggest that it is as accurate as the others and most importantly that it is much more faster as far as the convergence is concerned.

Table 4.8. Results for Scenario 4 ($s = 20$ and $r = 10$) and $n_k = 50$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 175 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 13 | 3 | 9 | 4 | 2 | 42 |
| Max iter. | 4 | 28 | 22 | 13 | 25 | 4 | 283 |
| Mean iter. | 3,2 | 23,2 | 5,6 | 11,2 | 6,3 | 3,2 | 94,5 |
| Time | 1E-02 | 1E-01 | 7E-01 | 4E+01 | 2E+01 | 8E-03 | 8E+00 |
| Time ratio | 1,0 | 10,7 | 61,2 | 3843,7 | 2052,2 | 1,0 | 955,9 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,0E-02 | 1,6E-03 | 1,6E-03 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 181 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 24 | 3 | 9 | 3 | 3 | 32 |
| Max iter. | 4 | 30 | 18 | 13 | 17 | 4 | 380 |
| Mean iter. | 4,0 | 26,4 | 5,4 | 11,2 | 5,2 | 4,0 | 70,5 |
| Time | 1E-02 | 1E-01 | 7E-01 | 4E+01 | 2E+01 | 1E-02 | 6E+00 |
| Time ratio | 1,0 | 10,1 | 49,1 | 3220,5 | 1678,4 | 1,0 | 615,0 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,6E-03 | 7,0E-03 | 1,6E-03 | 1,6E-03 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 190 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 14 | 3 | 1 | 1 | 2 | 51 |
| Max iter. | 4 | 29 | 12 | 12 | 19 | 4 | 390 |
| Mean iter. | 3,3 | 23,4 | 5,4 | 1,2 | 2,7 | 3,2 | 130,9 |
| Time | 1E-02 | 1E-01 | 7E-01 | 2E+01 | 2E+01 | 8E-03 | 1E+01 |
| Time ratio | 1,0 | 10,6 | 57,9 | 1454,3 | 1514,5 | 1,0 | 1358,3 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,3E-02 | 1,3E-02 | 1,6E-03 | 1,6E-03 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 182 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 14 | 3 | 1 | 1 | 2 | 34 |
| Max iter. | 4 | 28 | 21 | 13 | 17 | 4 | 315 |
| Mean iter. | 3,1 | 23,4 | 5,5 | 11,0 | 5,3 | 3,0 | 80,2 |
| Time | 1E-02 | 1E-01 | 7E-01 | 4E+01 | 2E+01 | 8E-03 | 7E+00 |
| Time ratio | 1,0 | 11,3 | 62,9 | 4020,0 | 2107,9 | 1,0 | 867,5 |
| MSE | 1,6E-03 | 1,6E-03 | 1,6E-03 | 1,6E-03 | 8,4E-03 | 1,6E-03 | 1,6E-03 |

Table 4.9. Results for Scenario 4 ($s = 20$ and $r = 10$) and $n_k = 5000$

| | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| **Initialisation 1** | | | | | | | |
| Nb. conver | 250 | 250 | 185 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 19 | 4 | 15 | 6 | 2 | 70 |
| Max iter. | 5 | 37 | 16 | 27 | 27 | 4 | 364 |
| Mean iter. | 3,8 | 33,5 | 5,9 | 17,0 | 7,7 | 3,8 | 127,7 |
| Time | 1E-02 | 2E-01 | 7E-01 | 6E+01 | 2E+01 | 1E-02 | 1E+01 |
| Time ratio | 1,0 | 11,8 | 49,4 | 4215,9 | 1607,1 | 1,0 | 974,5 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,4E-03 | 9,2E-03 | 1,8E-05 | 1,8E-05 |
| **Initialisation 2** | | | | | | | |
| Nb. conver | 250 | 250 | 179 | 250 | 250 | 250 | 250 |
| Min iter. | 4 | 33 | 4 | 13 | 5 | 4 | 40 |
| Max iter. | 5 | 37 | 22 | 28 | 19 | 5 | 208 |
| Mean iter. | 4,0 | 35,0 | 5,9 | 17,1 | 7,0 | 4,0 | 68,6 |
| Time | 2E-02 | 2E-01 | 7E-01 | 6E+01 | 2E+01 | 1E-02 | 6E+00 |
| Time ratio | 1,0 | 12,0 | 48,2 | 4174,1 | 1495,5 | 1,0 | 536,5 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 2,4E-03 | 6,3E-03 | 1,8E-05 | 1,8E-05 |
| **Initialisation 3** | | | | | | | |
| Nb. conver | 250 | 250 | 197 | 250 | 250 | 250 | 250 |
| Min iter. | 3 | 22 | 4 | 1 | 1 | 2 | 63 |
| Max iter. | 4 | 38 | 15 | 18 | 22 | 5 | 385 |
| Mean iter. | 3,9 | 33,8 | 5,8 | 1,6 | 2,9 | 3,8 | 172,4 |
| Time | 2E-02 | 2E-01 | 7E-01 | 2E+01 | 2E+01 | 1E-02 | 1E+01 |
| Time ratio | 1,0 | 11,6 | 46,9 | 1150,0 | 1149,9 | 1,0 | 1348,7 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,3E-02 | 1,3E-02 | 1,8E-05 | 1,8E-05 |
| **Initialisation 4** | | | | | | | |
| Nb. conver | 250 | 250 | 187 | 250 | 250 | 250 | 250 |
| Min iter. | 2 | 15 | 3 | 13 | 4 | 2 | 21 |
| Max iter. | 3 | 34 | 27 | 29 | 21 | 3 | 170 |
| Mean iter. | 2,9 | 30,7 | 5,9 | 17,1 | 7,3 | 2,9 | 49,4 |
| Time | 1E-02 | 2E-01 | 7E-01 | 6E+01 | 2E+01 | 9E-03 | 5E+00 |
| Time ratio | 1,0 | 13,3 | 60,3 | 5300,5 | 1902,2 | 1,0 | 523,1 |
| MSE | 1,8E-05 | 1,8E-05 | 1,8E-05 | 1,5E-03 | 6,4E-03 | 1,8E-05 | 1,8E-05 |

Table 4.10. Relative CPU Times for Initialization 3 and $n_k = 50$

| Dimension | R software | | | | | MATLAB software | |
|---|---|---|---|---|---|---|---|
| | CA | MM | NR | BFGS | NM | CA | IP |
| 16 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 51 | 1,0 | 1,6 | 3,3 | 4,3 | 2,6 | 1,0 | 5,2 |
| 101 | 1,0 | 2,8 | 9,6 | 8,6 | 7,6 | 1,1 | 13,3 |
| 201 | 1,2 | 5,2 | 32,3 | 24,9 | 30,9 | 1,2 | 39,4 |

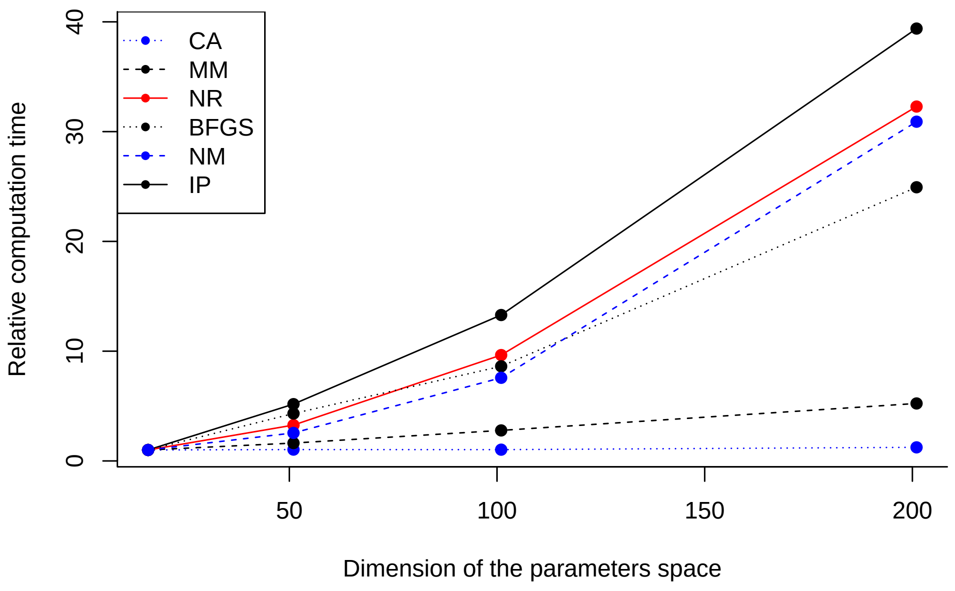

Figure 4.2. Relative CPU Time

# General conclusion and perspectives

## Conclusion

In this thesis, we studied the convergence properties of the constrained maximum likelihood estimator of a family of parameters of the form $(\theta, \boldsymbol{\phi})$ where $\theta > 0$ and $\boldsymbol{\phi}$ belongs to the multidimensional simplex. These parameters are derived from discrete probabilistic models aiming at modelling both road accident risks and the effect of changes in the road conditions. Instead of the classical Newton's method and its modifications which can be sensible to the choice of starting points and which involve matrix inversion, we considered a cyclic iterative algorithm (CA) that cycles through the estimators' components updating $\theta$ from $\boldsymbol{\phi}$ and $\boldsymbol{\phi}$ from $\theta$ and so on until a convergence criteria is satisfied. This CA is very simple to program without any matrix inversion. The theoretical and numerical convergence of this algorithm have been thoroughly studied in this thesis.

On the theoretical level, we proved that this algorithm has two of the properties most required for maximum likelihood estimation algorithms: (a) it converges to the maximum likelihood estimator (MLE) from any starting point and (b) it is an ascent algorithm, that is, the value of the log-likelihood is increased at each iteration. The results of the intensive numerical experiments performed in this thesis suggest that this algorithm is robust towards the starting values and that it converges to the desired value with few iterations and computation time. Moreover, the comparison of the performance of this cyclic algorithm with some of the best available optimization algorithms like Minorization-Maximization and Newton-Raphson's algorithms suggest that it is as accurate as the others and most importantly that it is the quickest as far as the convergence is concerned.

On the stochastic level, we proved that the MLE obtained from the CA is strongly consistent i.e. it converges to the true values of the unknown parameters when the sample size tends to infinity. This strong result was confirmed through numerical studies performed on some probability distributions such as the uniform, Gaussian and Dirichlet distributions.

Finally, we presented a generalization of the cyclic algorithm for maximum likelihood estimation of a family of parameters $(\theta, \boldsymbol{\phi})$ when $\boldsymbol{\phi}$ can be written in the form $(\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_s)$ with each $\boldsymbol{\phi}_k$ belonging to a multidimensional simplex. We proved that this generalization of the CA is also an ascent algorithm and that it also converges to the maximum likelihood estimate from any starting point. The results of the numerical experiments suggest that this generalization of the algorithm is robust towards the starting values, efficient, accurate and that it outperforms the best available optimization algorithms like MM and Newton-Raphson's algorithms. This generalization of the CA enables the estimation of the mean effect and the different accident risks when a road condition modification is applied to several areas, each with several accident types.

## Perspectives

A first natural extension of this work could be to generalize the strong consistency results to the multidimensional estimator proposed in [73] when we deal with the estimation of the effect of a road safety measure applied on $s$ ($s > 0$) different areas (sites), each with several accidents types.

It could also be interesting to extend our estimation algorithm to other models. One of them is the model proposed by N'Guessan et al. [74] in which the control coefficients per site are replaced by a mean control coefficient. Another possibility inspired from [77] would be to write the mean effect as a vector $\boldsymbol{\theta} = (\theta_1, \dots, \theta_s)$ where $\theta_k$, $k = 1, \dots, s$, represents the mean effect of the road measure at site $k$.

In the construction of the models considered in this thesis, it is assumed that the number of accidents per site denoted by $n_k$ is a non-random variable. We could modify this assumption by considering a random variable $N_k$ and then obtain conditional probabilities given $N_k = n_k$.

Finally, drawing our inspiration from the transformations proposed in Chapters 2 and 4 of this thesis, we could generalize the re-parametrization proposed by N'Guessan et al [70, 72] in order to introduce a logistic structure in their models and establish a link with logistic regression models.

# References

[1] Aitchison, J. and Silvey, S. D. (1958). Maximum-Likelihood Estimation of Parameters Subject to Restraints. *The Annals of Mathematical Statistics*, 29(3):813–828. [cited on pages 15 and 56].

[2] Allaire, G. (2005). *Analyse numérique et optimisation: une introduction à la modélisation mathématique et à la simulation numérique.* Editions Ecole Polytechnique. [cited on page 20].

[3] Badahur, R. R. (1958). Examples of Inconsistency of Maximum Likelihood Estimates. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 20(3/4):207–210. [cited on pages 85 and 90].

[4] Barvínek, E., Daler, I., and Francu, J. (1991). Convergence of sequences of inverse functions. *Archivum Mathematicum*, 27(3-4):201–204. [cited on page 92].

[5] Bonnans, J. F., Gilbert, Lemaréchal, C., and Sagastizábal, C. (2006). *Numerical Optimization – Theoretical and Practical Aspects.* Universitext. Springer Verlag, Berlin. [cited on pages 20 and 105].

[6] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press. [cited on pages 51 and 57].

[7] Broyden, C. G. (1970). The Convergence of a Class of Double-rank Minimization Algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90. [cited on pages 69 and 126].

[8] Chafai, D. and Concordet, D. (2007). On the strong consistency of asymptotic $m-$estimators. *Journal of Statistical Planning and Inference*, 137:2774–2783. [cited on pages 85 and 90].

[9] Chase, G. R. (1972). On the Chi-Square test when the parameters are estimated independently of the sample. *Journal of the American Statistical Association*, 67(339):609–611. [cited on page 43].

[10] Cheng, L., Geedipally, S., and Lord, D. (2013). The Poisson-Weibull generalized linear model for analyzing motor vehicle crash data. *Safety Science*, 54:38–42. [cited on page 42].

[11] Chiou, Y.-C. and Fu, C. (2013). Modeling crash frequency and severity using multinomial-generalized poisson model with error components. *Accident Analysis and Prevention*, 50:73–82. [cited on page 46].

[12] Chung, K. L. (2001). *A Course in Probability Theory.* Academic Press. [cited on page 95].

[13] Conn, A. R., Scheinberg, K., and Vicente, L. N. (2009). *Introduction to Derivative-Free Optimization.* MPS-SIAM Book Series on Optimization. SIAM, Philadelphia. [cited on pages 34 and 36].

[14] Cramer, H. (1946). *Mathematical methods of statistics.* Princeton University Press, Princeton. [cited on pages 85, 87, and 88].

[15] Davidon, W. (1959). Variable metric methods for minimization. Technical Report AEC Research and Development Report ANL-5990, Argonne National Laboratory, USA. [cited on page 26].

[16] de Leeuw, J. (1977). Applications of convex analysis to multidimensional scaling. In Barra, J., Brodeau, F., and Romier, G.and Van Cutsem, B., editors, *Recent Developments in Statistics*, pages 133–146. North Holland Publishing Company, Amsterdam. [cited on page 28].

[17] de Leeuw, J. (1994). Block-relaxation algorithms in statistics. In Bock, H.-H., Lenski, W., and Richter, M. M., editors, *Information Systems and Data Analysis*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 308–324. Springer Berlin Heidelberg. [cited on pages 30, 31, and 106].

[18] de Leeuw, J. and Michailidis, G. (2000). Discussion on the article "optimization transfer using surrogate objective functions". *Journal of Computational and Graphical Statistics*, 9(1):26–31. [cited on page 28].

[19] Demidovich, B. P. and Maron, I. A. (1981). *Computational Mathematics.* MIR Publishers, Moscow. [cited on pages 114 and 115].

[20] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38. [cited on pages 16, 29, and 50].

[21] Dennis, Jr, J. E. and Schnabel, R. B. (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM's Classics in Applied Mathematics. [cited on pages 20, 21, 24, 36, 50, 58, and 105].

[22] El Barmi, H. and Dykstra, R. L. (1998). Maximum likelihood estimates via duality for log-convex models when cell probabilities are subject to convex constraints. *The Annals of Statistics*, 26(5):1878–1893. [cited on page 56].

[23] Ferguson, T. S. (1982). An Inconsistent Maximum Likelihood Estimate. *Journal of the American Statistical Association*, 77(380):831–834. [cited on pages 85 and 90].

[24] Fiorin, S. (2000). The strong consistency for maximum likelihood estimates: a proof not based on the likelihood ratio. *C. R. Acad. Sci. Paris, Série I*, 331:721–726. [cited on pages 85 and 89].

[25] Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322. [cited on pages 69 and 126].

[26] Geraldo, I. C., N'Guessan, A., and Gneyou, K. E. (2015). A note on the strong consistency of a constrained maximum likelihood estimator used in crash data modelling. *Comptes Rendus Mathematique / C. R. Acad. Sci. Paris, Ser. I*, 353(12):1147–1152. [cited on page 85].

[27] Goldfarb, D. (1970). A Family of Variable Metric Updates Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26. [cited on pages 69 and 126].

[28] Golub, G. H. and Ortega, J. M. (2014). *Scientific Computing: An Introduction with Parallel Computing.* Elsevier. [cited on page 114].

[29] Griva, I., Nash, S. G., and Sofer, A. (2009). *Linear and Nonlinear Optimization: Second Edition.* Society for Industrial and Applied Mathematics. [cited on pages 20, 23, 26, 27, 32, and 105].

[30] Hadjicostas, P. (2003). Consistency of logistic regression coefficient estimates calculated from a training sample. *Statistics & Probability Letters*, 62(3):293–303. [cited on page 85].

[31] Hauer, E. (1997). *Observational Before After Studies in Road Safety.* Pergamon Press, Elsevier Science Ltd., Oxford, England. [cited on page 41].

[32] Hauer, E. (2010). Cause, effect and regression in road safety: A case study. *Accident Analysis and Prevention*, 42:1128–1135. [cited on page 54].

[33] Henningsen, A. and Toomet, O. (2011). maxlik: A package for maximum likelihood estimation in R. *Computational Statistics*, 26(3):443–458. [cited on page 70].

[34] Horn, R. A. and Johnson, C. R. (2013). *Matrix Analysis.* Cambridge University Press, New York, 2nd edition. [cited on page 51].

[35] Hunter, D. R. and Lange, K. (2000). Rejoinder to a discussion of "optimization transfer using surrogate objective functions". *Journal of Computational and Graphical Statistics*, 9(1):52–59. [cited on page 27].

[36] Hunter, D. R. and Lange, K. (2004). A Tutorial on MM Algorithms. *The American Statistician*, 58(1):30–37. [cited on pages 16, 28, 50, 105, and 106].

[37] Kelley, C. T. (1999). Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10:43–55. [cited on page 36].

[38] Kourouklis, S. (1987). On the strong consistency of a solution to the likelihood equation. *Statistics and Probability Letters*, 5:23–27. [cited on page 85].

[39] Kraft, C. and Le Cam, L. (1956). A remark on the roots of the maximum likelihood equation. *Ann. Math. Statist.*, 27(4):1174–1177. [cited on pages 85 and 90].

[40] Krantz, S. G. and Parks, H. R. (2013). *The implicit function theorem: history, theory, and applications*. Modern Birkhäuser Classics. Birkhäuser/Springer, New York. Reprint of the 2003 edition. [cited on page 120].

[41] Kumar, C. N., Parida, M., and Jain, S. (2013). Poisson family regression techniques for prediction of crash counts using bayesian inference. *Procedia - Social and Behavioral Sciences*, 104:982–991. 2nd Conference of Transportation Research Group of India (2nd CTRG). [cited on page 42].

[42] Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1998). Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization*, 9(1):112–147. [cited on page 35].

[43] Lange, K. (2010). *Numerical Analysis for Statisticians*. Springer. [cited on pages 16, 20, 21, 25, 28, 30, 33, 37, 50, 57, and 105].

[44] Lange, K. (2013). *Optimization*. Springer, New York, 2nd edition. [cited on pages 16, 20, 37, 39, 50, 57, 62, and 105].

[45] Lange, K., Chi, E. C., and Zhou, H. (2014). A Brief Survey of Modern Optimization for Statisticians. *International Statistical Review*, 82(1):46–70. [cited on pages 20, 27, 28, 30, 31, 50, and 105].

[46] Lange, K., Hunter, D. R., and Yang, I. (2000). Optimization transfer using surrogate objective functions (with discussion). *Journal of Computational and Graphical Statistics*, 9:1–21. [cited on page 105].

[47] Lassarre, S. (1977). A propos des tests statistiques, sur variables poissonniennes, utilisés dans le domaine de la sécurité routière. *Revue de Statistique Appliquée*, 25(3):55–74. [cited on page 43].

[48] Le Cam, L. (1990). Maximum Likelihood: An Introduction. *International Statistical Review*, 58(2):153–171. [cited on pages 85 and 90].

[49] Lee, S. and Huang, J. Z. (2013). A coordinate descent MM algorithm for fast computation of sparse logistic PCA. *Computational Statistics and Data Analysis*, 62:26–38. [cited on page 106].

[50] Lehmann, E. L. and Casella, G. (1998). *Theory of Point Estimation*. Springer, New York, 2nd edition. [cited on page 85].

[51] Liu, C. (2000). Estimation of Discrete Distributions with a Class of Simplex Constraints. *Journal of the American Statistical Association*, 95(449):109–120. [cited on page 56].

[52] Lord, D. and Mannering, F. (2010). The statistical analysis of crash-frequency data: A review and assessment of methodological alternatives. *Transportation Research Part A*, 44:291–305. [cited on pages 41 and 42].

[53] Lord, D. and Miranda-Moreno, L. (2008). Effects of low sample mean values and small sample size on the estimation of the fixed dispersion parameter of Poisson-gamma models for modeling motor vehicle crashes: a Bayesian perspective. *Safety Science*, 46(5):751–770. [cited on page 42].

[54] Luebeck, G. and Meza, R. (2013). *Bhat: General likelihood exploration*. R package version 0.9-10. [cited on page 70].

[55] Ma, J. and Kockelman, K. (2006). Bayesian multivariate Poisson regression for models of injury count by severity. *Transportation Research Record*, 1950:24–34. [cited on page 46].

[56] Ma, J., Kockelman, K., and Damien, P. (2008). A multivariate Poisson-lognormal regression model for prediction of crash counts by severity, using Bayesian methods. *Accident Analysis and Prevention*, 40(3):964–975. [cited on page 46].

[57] Mannering, F. L. and Bhat, C. R. (2014). Analytic methods in accident research: Methodological frontier and future directions. *Analytic Methods in Accident Research*, 1:1–22. [cited on pages 41 and 46].

[58] Martin, A. D., Quinn, K. M., and Park, J. H. (2011). MCMCpack: Markov chain monte carlo in R. *Journal of Statistical Software*, 42(9):1–21. [cited on page 101].

[59] McKinnon, K. (1998). Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization*, 9(1):148–158. [cited on page 36].

[60] McLachlan, G. and Krishnan, T. (2008). *The EM Algorithm and Extensions*. Wiley series in probability and statistics. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition. [cited on page 29].

[61] Meng, Q. and Qu, X. (2012). Estimation of rear-end vehicle crash frequencies in urban road tunnels. *Accident Analysis and Prevention*, 48:254–263. [cited on page 42].

[62] Meng, X. and Rubin, D. (1993). Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika*, 80:267–278. [cited on page 30].

[63] Meng, X.-L. and Van Dyk, D. (1997). The EM Algorithm – an Old Folk-song Sung to a Fast New Tune. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(3):511–567. [cited on page 30].

[64] Miao, W. and Hahn, M. G. (1996). Existence and strong consistency of maximum likelihood estimates for 1-dimensional exponential families. *Statistics & Probability Letters*, 28(1):9–21. [cited on page 85].

[65] Mkhadri, A., N'Guessan, A., and Hafidi, B. (2010). An MM algorithm for constrained estimation in a road safety measure modeling. *Communications in Statistics - Simulation and Computation*, 39(5):1057–1071. [cited on pages 50, 69, 70, 106, and 126].

[66] Monahan, J. F. (2011). *Numerical Methods of Statistics*. Cambridge University Press, 2nd edition. [cited on pages 20, 36, 66, 85, and 105].

[67] Nelder, J. A. and Mead, R. (1965). A simplex algorithm for function minimization. *Computer Journal*, 7(4):308–313. [cited on pages 35, 69, and 126].

[68] Ng, K. W., Tian, G.-L., and Tang, M.-L. (2011). *Dirichlet and Related Distributions: Theory, Methods and Applications*. Wiley Series in Probability and Statistics. John Wiley & Sons, 1st edition. [cited on pages 100 and 101].

[69] N'Guessan, A. (1993). *Contribution à l'analyse statistique d'une mesure de sécurité routière*. PhD thesis, Université Lille 1. [cited on page 43].

[70] N'Guessan, A. (2006). *Approches statistiques de l'évaluation d'une mesure : cas de la sécurité routière*. Habilitation à Diriger des Recherches (HDR), Université Lille 1. [cited on page 140].

[71] N'Guessan, A. (2010). Analytical Existence of solutions to a system of non-linear equations with application. *Journal of Computational and Applied Mathematics*, 234:297–304. [cited on pages 16, 17, 47, 50, 51, 53, 58, 61, 62, 85, 86, 91, and 110].

[72] N'Guessan, A. and Bellavance, F. (2005). A confidence interval estimation problem using schur complement approach. *C. R. Math. Rep. Acad. Sci. Canada*, 27(3):84–91. [cited on pages 43, 45, 55, and 140].

[73] N'Guessan, A., Essai, A., and Langrand, C. (2001). Estimation multidimensionnelle des contrôles et de l'effet moyen d'une mesure de sécurité routière. *Revue de statistique appliquée*, 49(2):85–102. [cited on pages 16, 20, 43, 45, 47, 53, 54, 56, 80, 90, 103, 107, 109, 129, and 140].

[74] N'Guessan, A., Essai, A., and N'zi, M. (2006). An estimation method of the average effect and the different accident risks when modelling a road safety measure: A simulation study. *Computational Statistics & Data Analysis*, 51:1260–1277. [cited on pages 43, 46, and 140].

[75] N'Guessan, A. and Geraldo, I. C. (2013). Comparison of a cyclic algorithm to some classical optimization algorithms for constrained estimation of a multinomial distribution. In *International conference of applied statistics for development in Africa SADA'13*, Cotonou, Benin. 4-8 march. [cited on page 49].

[76] N'Guessan, A. and Geraldo, I. C. (2015). A cyclic algorithm for maximum likelihood estimation using Schur complement. *Numerical Linear Algebra with Applications*, 22(6):1161–1179. [cited on pages 49 and 86].

[77] N'Guessan, A. and Langrand, C. (1993). Sur la distribution asymptotique de certaines statistiques utilisées dand le domaine de la sécurité routière. *C. R. Acad. Sci. Paris, Ser. I*, 317:401–404. [cited on pages 43, 46, and 140].

[78] N'Guessan, A. and Truffier, M. (2008). Impact d'un aménagement de sécurité routière sur la gravité des accidents de la route. *Journal de la Société Française de Statistique*, 149(3):23–41. [cited on pages 16, 50, 51, 53, 55, 58, 73, 85, 86, and 130].

[79] Nocedal, J. and Wright, S. J. (2006). *Numerical optimization*. Springer, second edition. [cited on pages 20, 21, 22, 26, 27, 32, 33, 34, 37, 39, 40, 41, 50, 58, and 105].

[80] Oh, J., Washington, S., and Nam, D. (2006). Accident prediction model for railway-highway interfaces. *Accident Analysis and Prevention*, 38(2). [cited on page 42].

[81] Oppe, S. (1979). The use of multiplicative models for analysis of road safety data. *Accidents Analysis & Prevention*, 11:101–115. [cited on page 44].

[82] Ortega, J. M. and Rheinbolt, W. C. (2000). *Iterative Solution of Nonlinear Equations in Several Variables*. Classics in Applied Mathematics. SIAM. republication of the work first published by Academic Press, New York and London, 1970. [cited on pages 28 and 114].

[83] Osborne, M. R. (1992). Fisher's method of scoring. *International Statistical Review / Revue Internationale de Statistique*, 60(1):99–117. [cited on page 24].

[84] Ouellette, D. V. (1981). Schur complement and statistics. *Linear Algebra and Its Applications*, 36:187–295. [cited on pages 51 and 52].

[85] Park, E.-S. and Lord, D. (2007). Multivariate Poisson-lognormal models for jointly modeling crash frequency by severity. *Transportation Research Record*, 2019:1–6. [cited on page 46].

[86] Phillips, G. M. M. and Taylor, P. J. (1996). *Theory and Applications of Numerical Analysis*. Elsevier Science & Technology Books, 2nd edition. [cited on page 114].

[87] R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. [cited on page 69].

[88] Rao, C. R. (1973). *Linear Statistical Inference and its Applications*. John Wiley & Sons, Inc. [cited on pages 15, 25, and 85].

[89] Rios, L. M. and Sahinidis, N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293. [cited on page 34].

[90] Rudin, W. (1976). *Principles of Mathematical Analysis.* International series in pure and applied mathematics. McGraw-Hill. [cited on pages 92, 120, and 121].

[91] Seo, B. and Lindsay, B. G. (2013). Nearly universal consistency of maximum likelihood in discrete models. *Statistics and Probability Letters*, 83:1699–1702. [cited on pages 85 and 90].

[92] Serfling, R. J. (1980). *Approximation Theorems of Mathematical Statistics.* Wiley. [cited on pages 88 and 92].

[93] Shanno, D. F. (1970). Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656. [cited on pages 69 and 126].

[94] Smyth, G. K. (1996). Partitioned algorithms for maximum likelihood and other nonlinear estimation. *Statistics and Computing*, 6. [cited on page 106].

[95] Smyth, G. K. (1998). Optimization and nonlinear equations. In Armitage, P. and Colton, T., editors, *Encyclopedia of Biostatistics*, pages 3174–3180. John Wiley & Sons, Ltd, Chichester, 2nd edition. [cited on page 25].

[96] Strichartz, R. S. (2000). *The Way of Analysis.* Jones and Bartlett books in mathematics. Jones and Bartlett Publishers. [cited on pages 92, 120, and 121].

[97] Tanner, J. C. (1958). A problem in the combination of accident frequencies. *Biometrika*, 45:331–342. [cited on pages 42, 43, 54, and 114].

[98] Tseng, P. (1999). Fortified-descent simplicial search method: a general approach. *SIAM Journal on Optimization*, 10(1):269–288. [cited on page 36].

[99] Van der Vaart, A. W. (1998). *Asymptotic Statistics.* Cambridge University Press. [cited on pages 85, 89, and 91].

[100] Varadhan, R. (2012). *alabama: Constrained nonlinear optimization.* R package version 2011.9-1, http://CRAN.R-project.org/package=alabama. [cited on pages 69 and 126].

[101] Wackerly, D. D., Mendenhall III, W., and Scheaffer, R. L. (2008). *Mathematical Statistics with Applications.* Thomson, 7ème edition. [cited on page 91].

[102] Wald, A. (1949). Note on the consistency of the maximum likelihood estimate. *The Annals of Mathematical Statistics*, 20(4):595–601. [cited on pages 85, 88, and 89].

[103] Waltz, R. A., Morales, J. L., Nocedal, J., and Orban, D. (2006). An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming*, 107(3):391–408. [cited on page 126].

[104] Wang, H. and Flournoy, N. (2015). On the consistency of the maximum likelihood estimator for the three parameter lognormal distribution. *Statistics & Probability Letters*, 105:57–64. [cited on page 85].

[105] Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103. [cited on pages 29 and 30].

[106] Ye, X., Pendyala, R., Shankar, V., and Konduri, K. (2013). A simultaneous model of crash frequency by severity level for freeway sections. *Accident Analysis and Prevention*, 57:140–149. [cited on page 42].

[107] Zhang, F. (2005). *The Schur Complement and Its Applications*. Springer US. [cited on pages 51, 52, and 53].

[108] Zhou, H. and Lange, K. (2010). MM algorithms for some discrete multivariate distributions. *Journal of Computational and Graphical Statistics*, 19(3):645–665. [cited on pages 16, 50, and 105].

# Appendix

# Appendix A

# Article extracted from Chapter 2

**A. N'Guessan and I. C. Geraldo (2015). A cyclic algorithm for maximum likelihood estimation using Schur complement, *Numerical Linear Algebra with Applications*, Vol. 22, No 6, pp. 1161-1179.**

NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
*Numer. Linear Algebra Appl.* 2015; **22**:1161–1179
Published online 18 June 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.1999

# A cyclic algorithm for maximum likelihood estimation using Schur complement

Assi N'Guessan[1,*,†] and Issa Cherif Geraldo[1,2]

[1]*Laboratoire Paul Painlevé (UMR CNRS 8524), Université de Lille 1, 59655 Villeneuve d'Ascq CEDEX, France*
[2]*Département de Mathématiques et Informatique, Université Catholique de l'Afrique de l'Ouest, Unité Universitaire du Togo (UCAO-UUT), 01 B.P. 1502 Lomé 01, Lomé, Togo*

SUMMARY

Using the Schur complement of a matrix, we propose a computational framework for performing constrained maximum likelihood estimation in which the unknown parameters can be partitioned into two sets. Under appropriate regularity conditions, the corresponding estimating equations form a non-linear system of equations with constraints. Solving this system is typically accomplished via methods which require computing or estimating a Hessian matrix. We present an alternative algorithm that solves the constrained non-linear system in block coordinate descent fashion. An explicit form for the solution is given. The overall algorithm is shown in numerical studies to be faster than standard methods that either compute or approximate the Hessian as well as the classical Nelder–Mead algorithm. We apply our approach to a motivating problem of evaluating the effectiveness of Road Safety Policies. This includes several numerical studies on simulated data. Copyright © 2015 John Wiley & Sons, Ltd.

Received 30 November 2013; Revised 20 May 2015; Accepted 28 May 2015

KEY WORDS: Schur complement; constrained optimization; maximum likelihood; cyclic algorithm; road safety measure; simulation

## 1. INTRODUCTION

The maximum likelihood method [1–3], very often quoted and used in statistics, is a numerical optimization method enabling, according to the problem data, to estimate the unknown parameters linked to a probability function. The approach consisting in maximizing this probability function is therefore called maximum likelihood method with or without constraints. This function is often obtained under the form of a product of probabilities under constraints. So it is equivalent to maximize its logarithm taking the constraints into account. We then talk about the constrained log likelihood method. One of the most popular and used probability functions is the one generally called multinomial law or distribution. The basic principle of this multinomial function consists in distributing a finite number of items in a finite number of categories or classes. The probability for an object to fall in a class is called class probability with the sum of all class probabilities equal to 1. Given a distribution, the class probability estimation is obtained through maximization of the class probability product under a linear constraint (sum of the class probabilities equal to 1) and under limit constraint (each class probability is between 0 and 1). The class probabilities maximizing this product are easily obtained thanks to the ratio between the number of objects fallen in a class and the total sum of the items to be distributed. In practice, unfortunately, each class probability depends not only on data to be distributed but also on unknown auxiliary parameters, which are very often under constraints.

*Correspondence to: Assi N'Guessan, Bât. Polytech'Lille, Université Lille 1, 59655 Villeneuve d'Ascq, France.
†E-mail: assi.nguessan@polytech-lille.fr

This constrained optimization problem can be solved thanks to different classic approaches. Most of these iterative methods need first and even second derivatives of the objective function. A complete review is given, for example, in [4] and [1]. The most often used and quoted basic iterative method is the Newton–Raphson one. It nevertheless implies the calculation and inversion of the Hessian matrix with each iteration. This Hessian matrix can be costly and difficult to obtain if the probability function (objective function) takes a complex form or if the dimension of the parameters increases along with the data dimension. In this case, the quasi-Newton methods represent an attractive alternative, due to the fact that they calculate an approximation of the inverse of the Hessian matrix using the expression for the gradient. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm described in [5–8] is one of the most popular quasi-Newton methods. There are also other methods which do not use derivatives, such as Nelder–Mead's algorithm [9].

All these classic methods have been adapted to the constrained maximum likelihood problem (see for example, [10–12]). Other authors (for example [13, 14]) also suggest Minorization–Maximization (MM) algorithms to solve particular situations. In spite of those significant contributions, the practical case studies still remain very sensitive, and several complications may compromise the performance of these traditional algorithms especially in the case of multivaried discrete data, which are as follows: (1) the Hessian matrix or an approximation can be costly in terms of calculation, (2) it may not be positively defined, that is, the inversion is not possible, (3) for data of important dimensions, solving the solution research linear system can be costly, (4) if parameter constraints or limit constraints appear, then the update itself needs adapted modifications, and (5) the choice of an initial solution vector enabling a rapid convergence remains an important key for all the iterative methods. Despite the many remedies and guarantees brought by scientific results, we are still facing the greater and greater complexity of numerical algorithms and the fact that they are not accessible to non-specialists.

In this particular context, N'Guessan and Truffier [15] and N'Guessan [16] have replaced the constrained maximum likelihood problem by the problem of solving a constrained non-linear equations systems. These authors have proved the existence of solutions but they only considered a few simulations and compared their results only to the Newton–Raphson method. In this paper, we generalise the latter authors' results and propose generic algorithms, one of which is N'Guessan's one [16]. Our results are thus organised as follows: Section 2 defines the problem, the associated conditions and the results obtained. The general structures of our algorithm are also found there. In Section 3, we rapidly describe classic constrained optimization algorithms competing with ours. In Section 4, we present a case study about the modelling of car crash data. Section 2's main results are used to obtain explicit forms of the mean effect estimation of a road safety measure and the associated seriousness risks. We also present the generic algorithm in a more accessible form in practice. In Section 5, we focus on simulations with different initial vectors and different values of the car crash number. The mean squared error is then used as a comparison criteria between our algorithm and other classic methods using or not the Hessian matrix. We finish with a conclusion in Section 6.

## 2. PROBLEM SETUP AND MAIN RESULTS

### 2.1. Problem setup

Let us consider $x = (x_1, \dots, x_R)^T$ a vector of dimension $R$ $(R > 2)$ made of observed data and $x_i (i = 1, \dots, R)$ being an integer or zero value such that $n = \sum_{i=1}^{R} x_i$, $(n > 0)$. We then focus on the maximization of a probability function noted $\ell(\Theta, x)$ where $\Theta \in \mathbb{R}^d$ is vector of dimension $d$ $(1 < d < R)$ made of unknown parameters and under constraint $h(\Theta) = 0$. This problem is equivalent to

$$\max_{\Theta \in \mathbb{R}^d} \ell(\Theta, x) \quad \text{under constraint } h(\Theta) = 0. \tag{1}$$

As the paper goes on, we suppose the following conditions:

$(H_1)$ $\Theta = (\theta, \phi^T)^T$, $\theta > 0$ and $\phi \in \mathbb{R}^{d-1}$ such that $\phi_j > 0$.

$(H_2)$ Functions $\ell : \mathbb{R}^d \to \mathbb{R}$, $\Theta \mapsto \ell(\Theta, x)$ and $h : \mathbb{R}^d \to \mathbb{R}$, $\Theta \mapsto h(\Theta)$ are continuously differentiable from $\Theta$.

$(H_3)$ $\frac{\partial h}{\partial \theta} = 0$, $\frac{\partial h}{\partial \phi_j} \neq 0\,(j = 1, \dots, d-1)$.

$(H_4)$ There is a non-zero constant $\eta$ such that $\langle \nabla_\phi h, \phi \rangle = \eta$ where $\nabla_\phi$ is the gradient operator, $\langle \cdot, \cdot \rangle$ is the classic scalar product (in relation to the identity matrix of $\mathbb{R}^{d-1}$).

$(H_5)$ Given $\hat{\Theta} = (\hat{\theta}, \hat{\phi}^T)^T$ the maximum of $\ell(\Theta, x)$ if it exists. Let us suppose that for a given $\hat{\theta}$, there is a $D_{\hat{\theta},x}$ non-singular matrix of dimension $(d-1) \times (d-1)$, a $B_x$ vector of dimension $(d-1) \times 1$ such that the non-linear system

$$(\nabla_\phi \ell)_{\hat{\Theta}} - \frac{1}{\eta} \langle (\nabla_\phi \ell)_{\hat{\Theta}}, \hat{\phi} \rangle (\nabla_\phi h)_{\hat{\Theta}} = 0_{d-1} \ ; \ h(\hat{\Theta}) = 0$$

is equivalent to the linear system in relation to $\hat{\phi}$

$$\begin{bmatrix} D_{\hat{\theta},x} & (\nabla_\phi h)_{\hat{\Theta}} \\ (\nabla_\phi h)^T_{\hat{\Theta}} & 0 \end{bmatrix} \begin{bmatrix} \hat{\phi} \\ 0 \end{bmatrix} = \begin{bmatrix} B_x \\ \eta \end{bmatrix}, \quad h(\hat{\Theta}) = 0,$$

where $0_{d-1} = (0, \dots, 0) \in \mathbb{R}^{d-1}$.

$(H_6)$ There are two functions $g_1 : \mathbb{R} \to \mathbb{R}$ invertible and $g_2 : \mathbb{R}^{d-1} \to \mathbb{R}$ such that equation $\left(\frac{\partial \ell}{\partial \theta}\right)_{\hat{\Theta}} = 0$ is equivalent to $g_1(\hat{\theta}) - g_2(\hat{\phi}; x) = 0$.

*Remark 1*

$(H_1)$ condition enables to specify a bit more precisely where vector $\Theta$ belongs. In particular, supposing that vector $\Theta$'s components are strictly positive is relative to the maximum likelihood principle where the used probability function logarithm is maximized. We are therefore led to take the logarithm of some components of the parameter vector as shown in the case study presented later. The separation of $\Theta$ components in two subsets is linked to the principle of alternate or cyclic estimation of components we present in our approach. Conditions $(H_2)$ to $(H_4)$ allow to characterise the structure of constraint $h$. In particular, we note that $h$ essentially depends on sub-vector $\phi$. Conditions $(H_5)$ and $(H_6)$ specify the estimation structuration by blocks of $\Theta$, which consists in linearly obtaining $\phi$, $\theta$ being set and vice versa. Thus, we can start the estimation procedure in initialising the first component. They also show that our method no longer uses second derivatives of $\ell(\Theta, x)$, which means neither the Hessian matrix nor an approximation.

*Lemma 1*

Under assumptions $(H_1)$–$(H_4)$, solution $\hat{\Theta}$ to problem (1), if existing, is also a solution to the following non-linear equation system:

$$\left(\frac{\partial \ell}{\partial \theta}\right)_{\hat{\Theta}} = 0 \quad \text{et} \quad (\nabla_\phi \ell)_{\hat{\Theta}} - \frac{1}{\eta} (\nabla_\phi \ell)^T_{\hat{\Theta}}\, \hat{\phi}\, (\nabla_\phi h)_{\hat{\Theta}} = 0_{d-1} \tag{2}$$

*Proof*

Problem (1) is equivalent to maximizing function

$$L(\Theta, x) = \ell(\Theta, x) - \lambda\, h(\Theta) \tag{3}$$

where $\lambda$ is the Lagrange multiplier. Let $\hat{\Theta}$ solution to $L(\Theta, x)$, if existing, such that $h(\hat{\Theta}) = 0$ then

$$(\nabla_\Theta L)_{\hat{\Theta}} = (\nabla_\Theta \ell)_{\hat{\Theta}} - \hat{\lambda}\, (\nabla_\Theta h)_{\hat{\Theta}} = 0_d \tag{4}$$

where $\hat{\lambda} = \lambda(\hat{\Theta})$. Considering $(H_1)$ to $(H_3)$, system (4) is equivalent to

$$\left(\frac{\partial \ell}{\partial \theta}\right)_{\hat{\Theta}} = 0 \quad \text{et} \quad \left(\frac{\partial \ell}{\partial \phi_j}\right)_{\hat{\theta}} - \hat{\lambda} \left(\frac{\partial h}{\partial \phi_j}\right)_{\hat{\Theta}} = 0,\ (j = 1, \dots, d-1).$$

Pre-multiplying the latter system by $\hat{\phi}_j$ and summing in relation to index $j$, we get

$$\sum_{j=1}^{d-1} \hat{\phi}_j \left(\frac{\partial \ell}{\partial \phi_j}\right)_{\hat{\Theta}} - \hat{\lambda} \sum_{j=1}^{d-1} \hat{\phi}_j \left(\frac{\partial h}{\partial \phi_j}\right)_{\hat{\Theta}} = 0.$$

This equation is equivalent to

$$\langle (\nabla_\phi \ell)_{\hat{\Theta}}, \hat{\phi}\rangle - \hat{\lambda}\langle (\nabla_\phi h)_{\hat{\Theta}}, \hat{\phi}\rangle = 0.$$

Whence equality $\hat{\lambda} = \lambda(\hat{\Theta}) = \frac{1}{\eta}\langle (\nabla_\phi \ell)_{\hat{\Theta}}, \hat{\phi}\rangle$ using $(H_4)$. We then obtain (2) by substitution of $\hat{\lambda}$ in (4). □

*Theorem 1*
Under assumptions $(H_1)$–$(H_6)$, constrained maximum likelihood $\hat{\Theta} = (\hat{\theta}, \hat{\phi}^T)^T$ of $\Theta$ is given by

$$\begin{aligned} \hat{\phi} &= D_{\hat{\theta},x}^{-1} B_x \\ \hat{\theta} &= g_1^{-1}(g_2(\hat{\phi}; x)). \end{aligned} \tag{5}$$

*Proof*
Using assumptions $(H_1)$–$(H_5)$ and Lemma 1, sub-vector $\hat{\phi}$ is the solution to system

$$\begin{bmatrix} D_{\hat{\theta},x} & (\nabla_\phi h)_{\hat{\Theta}} \\ (\nabla_\phi h)_{\hat{\Theta}}^T & 0 \end{bmatrix} \begin{bmatrix} \hat{\phi} \\ 0 \end{bmatrix} = \begin{bmatrix} B_x \\ \eta \end{bmatrix}, \quad \text{avec} \quad h(\hat{\Theta}) = 0.$$

We set

$$\Omega_{\hat{\theta},x} = \begin{bmatrix} D_{\hat{\theta},x} & (\nabla_\phi h)_{\hat{\Theta}} \\ (\nabla_\phi h)_{\hat{\Theta}}^T & 0 \end{bmatrix}.$$

Obtaining $\hat{\phi}$ consists in inverting matrix $\Omega_{\hat{\theta},x}$, using the general results in relation to the Schur complement [17–19]. Indeed, under assumptions $(H_5)$ and $(H_3)$, $D_{\hat{\theta},x}^{-1}$ exists and

$$(\nabla_\phi h)_{\hat{\Theta}}^T D_{\hat{\theta},x}^{-1} (\nabla_\phi h)_{\hat{\Theta}} = \|(\nabla_\phi h)_{\hat{\Theta}}\|_{D_{\hat{\theta},x}^{-1}}^2 > 0.$$

where $\|(\nabla_\phi h)_{\hat{\Theta}}\|_{D_{\hat{\theta},x}^{-1}}^2 = \langle (\nabla_\phi h)_{\hat{\Theta}}, (\nabla_\phi h)_{\hat{\Theta}}\rangle_{D_{\hat{\theta},x}^{-1}}$ is the norm of vector $(\nabla_\phi h)_{\hat{\Theta}}$ in relation to matrix $D_{\hat{\theta},x}^{-1}$. Consequently, the inversion of $\Omega_{\hat{\theta},x}$ is possible and we have

$$\Omega_{\hat{\theta},x}^{-1} = \begin{bmatrix} M_{\hat{\theta},x} & \|(\nabla_\phi h)_{\hat{\Theta}}\|_{D_{\hat{\theta},x}^{-1}}^{-2} D_{\hat{\theta},x}^{-1} (\nabla_\phi h)_{\hat{\Theta}} \\ \|(\nabla_\phi h)_{\hat{\Theta}}\|_{D_{\hat{\theta},x}^{-1}}^{-2} (\nabla_\phi h)_{\hat{\Theta}}^T D_{\hat{\theta},x}^{-1} & -\|(\nabla_\phi h)_{\hat{\Theta}}\|_{D_{\hat{\theta},x}^{-1}}^{-2} \end{bmatrix} \tag{6}$$

where

$$M_{\hat{\theta},x} = D_{\hat{\theta},x}^{-1} - \|(\nabla_\phi h)_{\hat{\Theta}}\|_{D_{\hat{\theta},x}^{-1}}^{-2} D_{\hat{\theta},x}^{-1} (\nabla_\phi h)_{\hat{\Theta}} (\nabla_\phi h)_{\hat{\Theta}}^T D_{\hat{\theta},x}^{-1}.$$

Technical details of this result are given as an appendix. We deduce $\hat{\phi}$ in multiplying $\Omega_{\hat{\theta},x}^{-1}$ by vector $\left(B_x^T, \eta\right)^T$. Condition $(H_6)$ enables to obtain $\hat{\theta}$ in function of $\hat{\phi}$. □

*Corollary 1*
In addition to the conditions of Theorem 1, we also suppose that

$(H_7)$ $R = 2r$, $d = r + 1$ ($r$ being a non-zero integer), $\theta > 0$, $\phi_j > 0$ such that $\sum_{j=1}^{r} \phi_j = 1$.

$(H_8)$ There is a vector $Z = (w_1, \dots, w_r)^T$, $w_j > 0$ such that

$$D_{\theta,x} = \Lambda_{\theta,Z} - \theta B_x Z^T$$

where $\Lambda_{\theta,Z} = \mathrm{Diag}(1+\theta w_1, \dots, 1+\theta w_r)$ is a diagonal matrix $r \times r$,

$$B_x = \left(\frac{x_1 + x_{1+r}}{n}, \dots, \frac{x_r + x_{2r}}{n}\right)^T \in \mathbb{R}^r$$

is a dimension $r$ vector thus defined with $n = \sum_{j=1}^r (x_j + x_{j+r})$.

Then

$$\hat{\phi}_j = \frac{1}{1 - \frac{1}{n}\sum_{m=1}^{r} \frac{\hat{\theta} w_m x_{\cdot m}}{1+\hat{\theta} w_m}} \times \frac{1}{1+\hat{\theta} w_j} \times \frac{x_j + x_{j+r}}{n}$$

$$\hat{\theta} = g_1^{-1}(g_2(\hat{\phi}, x)).$$

where $x_{\cdot m} = x_m + x_{m+r}$.

*Proof*
Using the fact that $\Lambda^{-1}_{\hat{\theta},Z}$ exists and $0 < \hat{\theta} Z^T \Lambda^{-1}_{\hat{\theta},Z} B_x < 1$, we show that $D^{-1}_{\hat{\theta},x}$ exists and that

$$D^{-1}_{\hat{\theta},x} = \Lambda^{-1}_{\hat{\theta},Z} + \hat{\theta}\left(1 - \hat{\theta}\langle Z, B_x\rangle_{\Lambda^{-1}_{\hat{\theta},Z}}\right)^{-1} \Lambda^{-1}_{\hat{\theta},Z} B_x Z^T \Lambda^{-1}_{\hat{\theta},Z}.$$

After some matrix manipulations, we show that

$$\hat{\phi} = D_{\hat{\theta},x} B_x = \frac{1}{1 - \hat{\theta}\langle Z, B_x\rangle_{\Lambda^{-1}_{\hat{\theta},Z}}} \Lambda^{-1}_{\hat{\theta},Z} B_x$$

where

$$1 - \hat{\theta}\langle Z, B_x\rangle_{\Lambda^{-1}_{\hat{\theta},Z}} = 1 - \frac{1}{n}\sum_{m=1}^{r} \frac{\hat{\theta} w_m x_{\cdot m}}{1+\hat{\theta} w_m}.$$

We then deduce the expression of $\hat{\phi}_j$ $(j = 1, \dots, r)$ of Corollary 1. A few lines of development will be found in the appendix. □

*Remark 2*
The corollary enables to have more explicit formulas for component $\hat{\phi}$ and thus generalises the results of [16]. We can then start the optimization procedures of $\ell(\Theta, x)$, working component after component. For example, when $\hat{\theta}^{(0)} = 0$ then $\hat{\phi}_j^{(1)} = (x_j + x_{j+r})/n$, $(j = 1, \dots, r)$ and $\hat{\theta}^{(1)} = g_1^{-1}(g_2(\hat{\phi}^{(1)}))$ and so on. We thus automate our estimation process of $\hat{\Theta}$ on focusing for example on the initialisation of its first component. So, looking for a constrained initial vector is brought to an initial point. We describe the general structure of our estimation method in relation to Theorem 1 in the succeeding sections. Then, we give a more convenient version within the framework of Corollary 1.

### *2.2. General framework of the cyclic algorithm*

This general approach allows to alternate the estimation of $\hat{\Theta}$ between its two components $\hat{\theta}$ and $\hat{\phi}$. To start the procedure, we initialise first component $\hat{\theta}^{(0)}$. Then, we compute $\hat{\phi}^{(0)} = D^{-1}_{\hat{\theta}^{(0)},x} B_x$ and define $\hat{\Theta}^{(0)} = (\hat{\theta}^{(0)}, (\hat{\phi}^{(0)})^T)^T$. Conversely, we can, if data allow, initialise $\hat{\phi}^{(0)}$ and get $\hat{\theta}^{(0)} = g_1^{-1}(g_2(\hat{\phi}^{(0)}, x))$. In step $k$ $(k > 0)$, we calculate $\hat{\theta}^{(k)} = g_1^{-1}(g_2(\hat{\phi}^{(k-1)}, x))$, then $\hat{\phi}^{(k)}$ thanks to $\hat{\theta}^{(k)}$, $B_x$ and $D^{-1}_{\hat{\theta}^{(k)},x}$. $\hat{\Theta}^{(k)}$ is updated. Maximization conditions of $\ell(\Theta, x)$ and assumptions are

**Algorithm 1** General scheme of the cyclic algorithm

```
Require: $x = (x_1, \ldots, x_R)^T$, $\epsilon_1 > 0$, $\epsilon_2 > 0$ two precisions, $\eta$.
Ensure: $\hat{\Theta}$ the MLE of $\Theta$, $k_0$ the number of iterations.

1:  Compute $B_x$, $\hat{\theta}^{(0)}$, $D^{-1}_{\hat{\theta}^{(0)},x}$, $\hat{\phi}^{(0)} = D^{-1}_{\hat{\theta}^{(0)},x} B_x$, $\hat{\Theta}^{(0)} = (\hat{\theta}^{(0)}, (\hat{\phi}^{(0)})^T)^T$, $\ell(\hat{\Theta}^{(0)}, x)$ and $(\nabla_\phi h)_{\hat{\Theta}^{(0)}}$.
2:  Set $k = 0$.
3:  Set STOP $= 0$.
4:  while STOP $\neq 1$ do
5:      Compute $\hat{\theta}^{(k+1)} = g_1^{-1}(g_2(\hat{\phi}^{(k)}), x)$ and $D^{-1}_{\hat{\theta}^{(k+1)},x}$.
6:      Compute $\hat{\phi}^{(k+1)} = D^{-1}_{\hat{\theta}^{(k+1)},x} B_x$.
7:      Set $\hat{\Theta}^{(k+1)} = \left(\hat{\theta}^{(k+1)}, (\hat{\phi}^{(k+1)})^T\right)^T$ and compute $\ell(\hat{\Theta}^{(k+1)}, x)$, $(\nabla_\phi h)_{\hat{\Theta}^{(k+1)}}$.
8:      if $|(\nabla_\phi h)_{\hat{\Theta}^{(k+1)}})^T \hat{\phi}^{(k+1)} - \eta| > \epsilon_1$ or $|\ell(\hat{\Theta}^{(k+1)}, x) - \ell(\hat{\Theta}^{(k)}, x)| > \epsilon_2$ then
9:          STOP $= 0$.
10:     else
11:         STOP $= 1$.
12:         Set $\hat{\Theta} = \Theta^{(k+1)}$.
13:     end if
14:     $k = k + 1$.
15: end while
16: Set $k_0 = k$.
17: Compute $h(\hat{\Theta})$, $\ell(\hat{\Theta}, x)$, $(\nabla_\phi h)_{\hat{\Theta}}$.
```

tested. The process is thus repeated until all conditions are satisfied. Using Corollary 1, we get Algorithm 1.

In this version, we note that we can start the procedure using the problem data via vector $B_x$. Thus, in practice, our algorithm is automated as soon as the problem data are entered, using $\hat{\phi}^{(0)} = B_x$ then $\hat{\theta}^{(1)} = g_1^{-1}(g_2(\hat{\phi}^{(0)}, x))$ and $\hat{\phi}_j^{(1)} = \left(1/\hat{\Delta}_n^{(1)}\right) \times \left(1 + \hat{\theta}^{(1)} w_j\right)^{-1} B_{x,j}$ and so on. On the whole, the choice of $\hat{\Theta}^{(0)}$ can be done on either component of $\hat{\Theta}$ according to the problem data. We can also note that the second partial derivatives of $\ell(\Theta, x)$ are no longer used in our algorithm.

The aim of this paper is not to carry out a theoretical study on the properties of the cyclic algorithm. We rather focus on the numerical properties of this algorithm through an application. Nevertheless, it can be noticed that the estimation of $\hat{\Theta}$ with the cyclic algorithm does not use any longer the computation and the inversion of the second derivative matrix of the objective function. This suggests that the estimation is improved, at least, in terms of computation time.

## 3. A BRIEF REVIEW OF SOME CLASSICAL OPTIMIZATION ALGORITHMS

### 3.1. *Newton–Raphson method*

Newton–Raphson method to maximize a function $\ell(\Theta)$ with $\Theta \in \mathbb{R}^d$, consists of an iterative plan taking the form

$$\hat{\Theta}^{(k+1)} = \hat{\Theta}^{(k)} - \left(\nabla^2_\Theta \ell\right)^{-1}_{\hat{\Theta}^{(k)}} (\nabla_\Theta \ell)_{\hat{\Theta}^{(k)}} \tag{7}$$

where $\left(\nabla^2_\Theta \ell\right)$ represents the Hessian matrix of $\ell$ and $\hat{\Theta}^{(0)}$ is the initial solution. This method's advantage is that it converges when initial solution $\hat{\Theta}^{(0)}$ is close to the true solution, which is unknown in practice. However, there are several problems which may arise when using Newton–Raphson's method. The first one being that evaluation and inversion of the Hessian matrix can appear very

**Algorithm 2**

```
Require: $x = (x_1, \dots, x_{2r})^T$, $\epsilon_1 > 0$, $\epsilon_2 > 0$, $\eta = 1$, $Z = (w_1, \dots, w_r)^T$, $1_r = (1, \dots, 1)^T \in \mathbb{R}^r$.
Ensure: $\hat{\Theta}$ the MLE of $\Theta$, $k_0$ the number of iterations.

1: Compute $n = \sum_{j=1}^{r}(x_j + x_{j+r})$, $B_x = n^{-1}(x_1 + x_{1+r}, \dots, x_r + x_{2r})^T$
2: Set $\hat{\theta}^{(0)} = 0$
3: For $j = 1, \dots, r$, Compute $B_{x,j} = n^{-1}(x_j + x_{j+r})$
4: Set $\hat{\Theta}^{(0)} = \left(0, B_x^T\right)^T$
5: Set $k = 0$.
6: Set STOP $= 0$.
7: while STOP $\neq 1$ do
8:     Compute $\hat{\theta}^{(k+1)} = g_1^{-1}(g_2(\hat{\phi}^{(k)}), x)$
9:     For $m = 1, \dots, r$, $\hat{\Delta}_{n,m}^{(k+1)} = \dfrac{\hat{\theta}^{(k+1)} w_m (x_m + x_{m+r})}{n(1 + \hat{\theta}^{(k+1)} w_m)}$ and $\hat{\Delta}_n^{(k+1)} = 1 - \sum_{m=1}^{r} \hat{\Delta}_{n,m}^{(k+1)}$.
10:    For $j = 1, \dots, r$, compute $\hat{\phi}_j^{(k+1)} = \frac{1}{\hat{\Delta}_n^{(k+1)}} \times \frac{B_{x,j}}{1+\hat{\theta}^{(k+1)} w_j}$.
11:    Compute $\hat{\phi}^{(k+1)} = (\hat{\phi}_1^{(k+1)}, \dots, \hat{\phi}_r^{(k+1)})^T$.
12:    Set $\hat{\Theta}^{(k+1)} = \left(\hat{\theta}^{(k+1)}, (\hat{\phi}^{(k+1)})^T\right)^T$.
13:    if $|1_r^T \phi^{(k+1)} - \eta| > \epsilon_1$ or $|\ell(\hat{\Theta}^{(k+1)}, x) - \ell(\hat{\Theta}^{(k)}, x)| > \epsilon_2$ then
14:        STOP $= 0$.
15:    else
16:        STOP $= 1$.
17:        Set $\hat{\Theta} = \Theta^{(k+1)}$.
18:    end if
19:    $k = k + 1$.
20: end while
21: Set $k_0 = k$.
22: Compute $h(\hat{\Theta})$, $\ell(\hat{\Theta}, x)$.
```

difficult and costly, numerically speaking, if $d$, the parameters' space dimension, is high or if the expression of $\ell(\Theta)$ is complex. Another possibility consists in proceeding in two steps at each iteration. The linear system

$$\left(\nabla_\Theta^2 \ell\right)_{\hat{\Theta}^{(k)}} s = (\nabla_\Theta \ell)_{\hat{\Theta}^{(k)}}$$

is solved and then we proceed to updating $\hat{\Theta}^{(k+1)} = \hat{\Theta}^{(k)} + s$. But this does not fundamentally change the situation because nothing can prove that the Hessian matrix is positive and therefore can be inverted. The second possible problem comes from the fact that when $\hat{\Theta}^{(k)}$ is far from the true solution, Newton's method is not an ascent method, that is, we do not necessarily have $\ell(\hat{\Theta}^{(k+1)}) > \ell(\hat{\Theta}^{(k)})$.

## 3.2. *The minorization-maximization (MM) algorithms*

In maximization problems, the first M of the acronym MM stands for minorize and the second M for maximize. The MM philosophy consists in substituting a simple optimization problem for a difficult optimization problem. It can be summarized as follows (for example [20]).

The first M step of a minorize-maximize MM algorithm consists in defining a function $g_{\hat{\Theta}^{(k)}}(\Theta)$ that minorizes $\ell(\Theta)$ at the point $\hat{\Theta}^{(k)}$, that is

$$g_{\hat{\Theta}^{(k)}}(\Theta) \leqslant \ell(\Theta), \quad \text{for all } \Theta,$$
$$g_{\hat{\Theta}^{(k)}}(\hat{\Theta}^{(k)}) = \ell(\hat{\Theta}^{(k)}).$$

where $\hat{\Theta}^{(k)}$ represents the current iterate. In the second M step, the next iterate $\hat{\Theta}^{(k+1)}$ is produced by maximizing the minorizing function $g_{\hat{\Theta}^{(k)}}(\Theta)$ rather than the actual function $\ell(\Theta)$.

### 3.3. The quasi-Newton methods

The main characteristic of the quasi-Newton methods is that they use an inverse approximation of the Hessian matrix $\left(\nabla^2_\Theta \ell\right)_{\hat{\Theta}^{(k)}}$ at each iteration. They, thus, enable to avoid the calculation (and inversion) of the Hessian matrix. Starting from the following first-order approximation:

$$(\nabla_\Theta \ell)_{\hat{\Theta}^{(k)}} - (\nabla_\Theta \ell)_{\hat{\Theta}^{(k+1)}} \approx \left(\nabla^2_\Theta \ell\right)_{\hat{\Theta}^{(k)}} (\hat{\Theta}^{(k)} - \hat{\Theta}^{(k+1)}). \tag{8}$$

and noting $J_{k+1}$ the approximation the inverse of Hessian matrix $\left(\nabla^2_\Theta \ell\right)_{\hat{\Theta}^{(k)}}$,

$$\begin{aligned} s_k &= \hat{\Theta}^{(k+1)} - \hat{\Theta}^{(k)} \\ y_k &= (\nabla_\Theta \ell)_{\hat{\Theta}^{(k+1)}} - (\nabla_\Theta \ell)_{\hat{\Theta}^{(k)}}, \end{aligned}$$

condition (8) is equivalent to $J_{k+1} y_k = s_k$ and called equation of the secant. To determine $J_{k+1}$, we use the BFGS formula, which is considered as the most efficient of the quasi-Newton formulas [3, 4]. Its name coming from its authors' names (Broyden, Fletcher, Goldfarb, Shanno), it consists of an update of the form

$$J_{k+1} = \left(I - \gamma_k s_k y_k^T\right) J_k \left(I - \gamma_k y_k s_k^T\right) + \gamma_k s_k s_k^T \tag{9}$$

with

$$\gamma_k = \frac{1}{y_k^T s_k}. \tag{10}$$

The quasi-Newton methods have several advantages : they do not need first-order derivatives, they save the Hessian matrix inversion, they remain 'close' to the Newton method and matrix $J_k$ is positively defined, which enables to guarantee their 'success' (convergence).

### 3.4. The derivative-free optimization algorithms

The derivative-free optimization algorithms (DFO), as its name shows, do not use the derivative of the function to be optimised. These algorithms therefore do not approximate the gradient but determine the successive iterations from the function values on a finite set of points.

Among the DFO algorithms, the Nelder–Mead method is particularly used. It is a heuristic method (rapidly giving a workable solution, not necessarily optimal or exact) proposed by Nelder and Mead [9] in 1965. It consists in maximizing a function $\ell(\Theta)$, $\Theta \in \mathbb{R}^d$, starting from a simplex (a set of $d+1$ points) of $\mathbb{R}^d$. With each iteration, the vertex point with the smallest value by $\ell$ is replaced by another point. Iterations are repeated until the images of the vertices by $\ell$ are sufficiently close.

## 4. A CASE STUDY

Our results and algorithms are applied to model a road accident data when a road safety measure (crossroad lay-out, surface of a motorway section, etc.) is applied to an experimental site presenting several mutually exclusive accident types (fatal accidents, seriously injured people, slightly injured people, material damage, etc.) over a fixed period of time.

### 4.1. Statistical model

Let $r > 1$ the number of different accident types occurring on the experimental site, $X_1 = (X_{11}, \ldots, X_{1r})^T$ (respectively $X_2 = (X_{21}, \ldots, X_{2r})^T$) the random vector giving the number of accident of each type on the experimental site in the period before (respectively after) the application of the measure. The vectors $X_1$ and $X_2$ are such that $X_{1j}$ (respectively $X_{2j}$), $j = 1, \ldots, r$

represents the number of crashes of type $j$ occurred in the 'before' (respectively 'after') period. In order to take into account some external factors (traffic flow, speed limit variation, weather conditions, etc.), the experimental site is associated to a control area where the safety measure was not directly applied. Let $C = (c_1, \dots, c_r)^T$ a $r \times 1$ vector such that $c_j$ denotes the ratio of the number of accidents of type $j$ for the period after to the period before in the control area over the same time period. The vector $C$ is assumed to be fixed and known and its components are called control coefficients. We adopt the before–after multinomial modelling proposed by N'Guessan *et al.* [21] when the number of sites is equal to 1. We therefore assume that $(X_1, X_2)$ has the multinomial distribution

$$(X_1, X_2) \sim \mathcal{M}(n; p_1(\Theta), p_2(\Theta))$$

where $n$ is the total number of accidents recorded on the experimental site in both periods, and the components $p_{tj}(\Theta)$ of $p_t(\Theta)$; $(t = 1, 2)$ are given by

$$p_{1j}(\Theta) = \frac{\phi_j}{1 + \theta C^T \phi} \quad \text{and} \quad p_{2j}(\Theta) = \frac{\theta c_j \phi_j}{1 + \theta C^T \phi} \qquad \forall j = 1, \dots, r \tag{11}$$

where $\theta > 0$, $0 < \phi_j < 1$ with $\sum_{j=1}^{r} \phi_j = 1$. The scalar $\theta$ represents the unknown average effect of the road safety measure, while each $\phi_j\,(j = 1, 2, \dots, r)$ denotes the global accident risk of type $j$ before and after the application of the road safety measure. A value of $\theta$ significantly lower than 1 means that the introduction of changes has diminished the number of crashes. The parameter vector $\Theta = (\theta, \phi^T)^T$ is subjected to the following constraints $\theta > 0$, $0 < \phi_j < 1$ and the linear constraint

$$h(\Theta) = 0, \quad \text{with} \quad h(\Theta) = \phi^T 1_r - 1. \tag{12}$$

Given an observed data $x = (x_1; x_2)$ where $x_1 = (x_{11}, \dots, x_{1r})$ and $x_2 = (x_{21}, \dots, x_{2r})$ and using the cell probabilities expression of (11), the probability function related to the random vector $(X_1, X_2)$ is given by

$$L(\Theta, x) = \frac{n!}{\prod_{j=1}^{r} x_{1j}!x_{2j}!} \left(\frac{\phi_j}{1 + \theta C^T \phi}\right)^{x_{1j}} \left(\frac{\theta c_j \phi_j}{1 + \theta C^T \phi}\right)^{x_{2j}}. \tag{13}$$

Unknown vector $\Theta$ under the (12) constraints is estimated in maximizing $L(\Theta, x)$ when we know vector $x$. It is equivalent to maximize $\ell(\Theta, x) = \log(L(\Theta, x))$ given to one additive constant as

$$\ell(\Theta, x) = \sum_{j=1}^{r} \left[ x_{\cdot j} \log(\phi_j) + x_{2j} \log(\theta) - x_{\cdot j} \log\left(1 + \theta \sum_{m=1}^{r} c_m \phi_m\right)\right] \tag{14}$$

with $x_{\cdot j} = x_{1j} + x_{2j}$. Different iterative methods [13, 22] may be used to solve the constrained maximum likelihood estimation problem of $\Theta$. Newton–Raphson method or Fisher scoring method for the computation of a solution are probably the most widely used iterative methods. Each of them computes the second-order derivatives of $\ell(\Theta, x)$ and need a starting vector $\hat{\Theta}^{(0)}$. However, such methods can be unsuccessful if $\hat{\Theta}^{(0)}$ is not close enough to the true value, which in practice is unknown.

*Remark 3*
Expression $\ell(\Theta, x)$ given in formula (14) and the constraints relative to parameters $\theta$ and $\phi$ of model (11) justify assumption $(H_1)$. Assumption $(H_2)$ is naturally verified along with $(H_3)$ with $\nabla_\phi h = (1, \dots, 1)^T \in \mathbb{R}^r$. Similarly, we show that $(\nabla_\phi h)^T \phi = 1 = \eta$.

## 4.2. *Cyclic algorithm for the estimation of the average effect and the different risks*

Deriving $\ell(\Theta, x) - \lambda h(\Theta)$ in relation to $\theta$ and $\phi_j$ $(j = 1, \dots, r)$, we show that

$$\begin{aligned}\frac{\partial \ell}{\partial \theta} &= \frac{\sum_{j=1}^{r} x_{2j}}{\theta} - \frac{n c^T \phi}{1+\theta c^T \phi}\\ \frac{\partial \ell}{\partial \phi_j} &= \frac{x_{.j}}{\phi_j} - \frac{\theta c_j n}{1+\theta c^T \phi} - \lambda, \quad j = 1,\dots,r.\end{aligned} \tag{15}$$

where $\lambda$ is the Lagrange multiplier. When setting the partial derivatives of (15) to zero, we show that $\hat{\Theta}$, if existing, is the solution to the following non-linear equation system:

$$\begin{aligned}\frac{\sum_{j=1}^{r} x_{2j}}{\hat{\theta}} - \frac{n c^T \hat{\phi}}{1+\hat{\theta} c^T \hat{\phi}} &= 0\\ \frac{x_{.j}}{\hat{\phi}_j} - \frac{\hat{\theta} c_j n}{1+\hat{\theta} c^T \hat{\phi}} - \hat{\lambda} &= 0, \quad j = 1,\dots,r.\\ 1_r^T \hat{\phi} &= 1\end{aligned} \tag{16}$$

where $\hat{\lambda} = n(1+\hat{\theta} c^T \hat{\phi})^{-1}$.

*Corollary 2*
The components of $\hat{\Theta}$ solution to (16) are obtained with the following expressions:

$$\begin{cases}\hat{\theta} = x_{2.}(x_{1.})^{-1}(c^T\hat{\phi})^{-1}\\ \hat{\phi}_j = \left(1 - \frac{1}{n}\sum_{m=1}^{r} \frac{\hat{\theta} c_m x_{.m}}{1+\hat{\theta} c_m}\right)^{-1} \times \frac{x_{.j}}{n(1+\hat{\theta} c_j)} \qquad (j = 1,\dots,r),\end{cases} \tag{17}$$

where $x_{.m} = x_{1m} + x_{2m}$ et $x_{t\cdot} = \sum_{m=1}^{r} x_{tm}$, $(t = 1, 2)$.

*Remark 4*
Corollary 2's proof is similar to that of Corollary 1, taking $Z = C$, function $g_1$ equal to the identity and function $g_2(\phi, x) = x_{2.}(x_{1.})^{-1}(c^T\hat{\phi})^{-1}$. Moreover, replacing $\hat{\lambda} = \lambda(\hat{\Theta}) = n(1 + c^T\hat{\phi})^{-1}$ in the second equation of (16) and by multiplying this latter equation in relation to ratio $\hat{\phi}_j/n$, we obtain the non-linear equation system in relation to $\hat{\phi}_j$

$$(1 + c^T\hat{\phi})^{-1}(1+\hat{\theta} c_j)\hat{\phi}_j = \frac{x_{.j}}{n}.$$

We then show (see [16] for technical details) that the latter system is equivalent to

$$D_{\hat{\theta},x}\hat{\phi} = B_x,$$

where $D_{\hat{\theta},x} = \Lambda_{\hat{\theta},x} - \hat{\theta} B_x c^T$.

Thanks to the explicit formulas of the components of $\hat{\Theta}$ of Corollary 2, we propose Algorithm 3, which is a convenient form of Algorithm 2.

*Remark 5*
As in Algorithm 2, we could have started Algorithm 3 at $\hat{\phi}^{(0)} = B_x$ because $B_x$ is available and moreover $1_r^T B_x = 1$. Other initial solutions can also be used as the numerical studies in the following section show.

## 5. NUMERICAL STUDIES

This section focuses on the numerical study of the cyclic algorithm (CA) applied to the multinomial model presented in Section 4. To the authors knowledge, three criteria are usually investigated on iterative algorithms, which are as follows: robustness (the algorithm should perform well for all

**Algorithm 3**

```
Require: $x = (x_{11}, \dots, x_{1r}, x_{21}, \dots, x_{2r})^T$, $\epsilon_1 > 0$, $\epsilon_2 > 0$.
Ensure: $\hat{\Theta}$ the MLE of $\Theta$, $k_0$ the number of iterations.

 1: Compute $n$, $B_x = n^{-1}(x_{.1}, \dots, x_{.r})^T$, $x_{1.}$, $x_{2.}$.
 2: Set $\hat{\phi}^{(0)} = B_x$, $\hat{\Theta}^{(0)} = \left(0, B_x^T\right)^T$.
 3: Set $k = 0$.
 4: Set STOP $= 0$.
 5: while STOP $\neq 1$ do
 6:     Compute $\hat{\theta}^{(k+1)} = x_{2.}(x_{1.})^{-1}(c^T\hat{\phi}^{(k)})^{-1}$
 7:     Compute $\hat{\Delta}_{n,m}^{(k+1)} = (x_{.m}/n) \times c_m\hat{\theta}^{(k+1)}/(1 + c_m\hat{\theta}^{(k+1)})$ for $m = 1, \dots, r$.
 8:     Compute $\hat{\Delta}_n^{(k+1)} = 1 - \sum_{m=1}^{r} \hat{\Delta}_{n,m}^{(k+1)}$.
 9:     Compute $\hat{\phi}_j^{(k+1)} = \dfrac{1}{\hat{\Delta}_n(k+1)} \times \dfrac{x_{.j}}{n(1 + \hat{\theta}^{(k+1)}c_j)}$ for $j = 1, \dots, r$.
10:     Set $\hat{\phi}^{(k+1)} = (\hat{\phi}_1^{(k+1)}, \dots, \hat{\phi}_r^{(k+1)})^T$.
11:     Set $\hat{\Theta}^{(k+1)} = \left(\hat{\theta}^{(k+1)}, (\hat{\phi}^{(k+1)})^T\right)^T$.
12:     if $|1_r^T\hat{\phi}| > \epsilon_1$ or $|\ell(\hat{\Theta}^{(k+1)}, x) - \ell(\hat{\Theta}^{(k)}, x)| > \epsilon_2$ then
13:         STOP $= 0$
14:     else
15:         STOP $= 1$
16:         Set $\hat{\Theta} = \Theta^{(k+1)}$.
17:     end if
18:     $k = k + 1$.
19: end while
20: Set $k_0 = k$.
```

reasonable choices of the initial guess), accuracy (the algorithm should be able to identify a solution near the true values with precision) and efficiency (the algorithm should not require too much computation time or storage).

In our experiment, the robustness will be checked trough the number of iterations. If the number of iterations is approximately the same for different starting values $\hat{\Theta}^{(0)}$, then it can be said that the algorithm is robust. In order to evaluate the accuracy of the algorithm, we compute the mean squared error (MSE)

$$\text{MSE}(\hat{\Theta}) = \frac{1}{1+r}\left((\hat{\theta} - \theta^0)^2 + \sum_{j=1}^{r}\left(\hat{\phi}_j - \phi_j^0\right)^2\right) \tag{18}$$

with $\Theta^0 = \left(\theta^0, (\phi^0)^T\right)^T$ the true parameter vector. Efficiency will be monitored by the central process unit (CPU) time computed in seconds.

We also compare our cyclic algorithm to some of the best available optimization algorithms. The methods selected for this comparison are described in Section 3. That is, the Newton–Raphson algorithm, the quasi-Newton BFGS algorithm [5–8], the Nelder–Mead algorithm [9] and Minorization-Maximization (MM) algorithm proposed by [14]. The performances of the different algorithms in terms of computation time are compared with their respective CPU time ratios calculated as the ratio between their mean duration and the mean duration of the cyclic algorithm. Thus, the CPU time ratio of the cyclic algorithm is always equal to 1.

The computations presented in this section were conducted in R software [23] (version 3.0.2) and executed on a PC with an AMD E-350 Processor 1.6 GHz CPU. Any execution time given below is the user CPU time reported by the R function `system.time()`. The BFGS and Nelder–Mead algorithm are implemented using the function `constrOptim.nl` of the `alabama` R package [24]. The Newton–Raphson algorithm is implemented using the description given in Section 3 while the MM algorithm is implemented using the algorithm proposed in [14]. As shown in Algorithm 3, for example, the usual way of calculating the number of iterations consists in counting the number of times when an algorithm is repeated until all the stopping criteria are satisfied.

The R codes generating our numerical results can be downloaded from the web-site http://www.assi-nguessan.fr.

### 5.1. Data generation principle

Given $r$ (the number of accident types) and $n$ (the total number of crashes), we generate the components of vector $C = (c_1, \dots, c_r)^T$ from a uniform random variable $U[0.5; 2.5]$. We generate the true value of parameter $\theta$, denoted $\theta^0$, as the mean of a uniform random variable $U]0;1[$ and the true value of vector $\phi$, denoted $\phi^0 = \left(\phi_1^0, \dots, \phi_r^0\right)^T$ from a Dirichlet distribution. Afterwards, we compute the true values

$$p_{1j}^0(\Theta) = \frac{\phi_j^0}{1 + \theta^0 \sum_{m=1}^r c_m \phi_m^0} \quad \text{and} \quad p_{2j}^0(\Theta) = \frac{\theta^0 c_j \phi_j^0}{1 + \theta^0 \sum_{m=1}^r c_m \phi_m^0} \qquad \forall j = 1, \dots, r$$

linked to the multinomial distribution of $(X_1, X_2)$. Finally, the data $x = (x_1, x_2)$ is randomly generated from the multinomial distribution $\mathcal{M}\left(n; p_1^0(\Theta^0), p_2^0(\Theta^0)\right)$.

### 5.2. Results

The results presented in this paper correspond to $r \in \{3; 5\}$ and two values of $n$: a small value ($n = 50$) and a great value ($n = 5000$). In order to explore all the possible starting positions, we have considered four different ways of setting the starting parameter vector $\hat{\Theta}^{(0)} = \left(\hat{\theta}^{(0)}, (\hat{\phi}^{(0)})^T\right)^T$. The parameter $\hat{\theta}^{(0)}$ is randomly generated and the parameter vector $\hat{\phi}^{(0)}$ is randomly generated in four different ways:

($I_1$) Uniform: $\hat{\phi}^{(0)} = \left(\frac{1}{r}, \cdots, \frac{1}{r}\right)^T$.

($I_2$) Proportional I: $\hat{\phi}^{(0)} = \dfrac{1}{n}(x_1 + x_2)$.

($I_3$) Random: $\hat{\phi}^{(0)} = \frac{1}{s}U$ where $U = (u_1, \dots, u_r)^T$ is an $r-$dimensional vector whose components are randomly generated from a uniform distribution $\mathcal{U}[0.05; 0.95]$ and $s = \sum_{j=1}^r u_j$.

($I_4$) Proportional II: $\hat{\phi}^{(0)} = \frac{1}{n_1}x_1$ where $n_1 = \sum_{j=1}^r x_{1j}$. This setting of $\hat{\phi}^{(0)}$ is suggested by some basic assumptions made in [21].

By combining these different values of $r$, $n$ and $\hat{\Theta}^{(0)}$ we get 16 different scenarios. The results presented in Tables II–IX correspond to the mean values obtained for 1000 simulations for all the scenarios.

*5.2.1. Numerical study of the cyclic algorithm.* The overall performance of the cyclic algorithm is presented in Table I. The efficiency will be studied later on when we compare the CA with the other algorithms.

As mentioned earlier, we use the MSE as an indicator of the accuracy of the algorithm. It is noticed that the MSE has an order of $10^{-2}$ when $n$ is small and $10^{-4}$ when $n$ is great. The results presented in Tables II–IX show that the estimates produced by the CA are quite near to the true values, and when $n$ is great, the estimates match the true values.

The results also suggest that the cyclic algorithm is robust towards the starting value (or initial guess). For each value of $r$, 8000 starting values (corresponding to four initialization schemes for $\hat{\phi}^{(0)}$, two values of $n$ and 1000 simulations) are randomly selected in the parameter space. For all these values, the number of iterations lies between 3 and 4 on average. It can be noticed that for a given value of $n$, there is no significant increase of the number of iterations when the dimension of the parameter space ($r + 1$) increases. When $n$ increases, a slight increase of the mean number

Table I. Results of the cyclic algorithm for all the 16 000 simulations.

| | | $n = 50$ | | | $n = 5000$ | | |
|---|---|---|---|---|---|---|---|
| $r$ | Init. | MSE | Iterations | CPU time | MSE | Iterations | CPU time |
| 3 | $I_1$ | 9,65E-03 | 3,5 | 9,24E-04 | 8,21E-05 | 4,1 | 1,04E-03 |
| | $I_2$ | 9,19E-03 | 3,2 | 9,08E-04 | 8,09E-05 | 3,7 | 9,65E-04 |
| | $I_3$ | 8,33E-03 | 3,5 | 9,67E-04 | 8,12E-05 | 4,0 | 1,02E-03 |
| | $I_4$ | 9,40E-03 | 3,1 | 8,86E-04 | 8,24E-05 | 3,1 | 1,03E-03 |
| 5 | $I_1$ | 6,68E-03 | 3,6 | 9,31E-04 | 6,04E-05 | 4,2 | 1,18E-03 |
| | $I_2$ | 6,67E-03 | 3,5 | 9,14E-04 | 6,16E-05 | 4,1 | 1,06E-03 |
| | $I_3$ | 6,51E-03 | 3,6 | 9,59E-04 | 5,99E-05 | 4,2 | 1,03E-03 |
| | $I_4$ | 7,00E-03 | 3,2 | 8,11E-04 | 6,09E-05 | 3,2 | 8,75E-04 |

MSE, mean squared error; CPU, central process unit.

Table II. Results for scenario $r = 3$ and $\hat{\phi}^{(0)} = (1/r, \ldots, 1/r)^T$.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,532 | 0,532 | 0,532 | 0,532 | 0,532 |
| | 0,019 | 0,051 | 0,051 | 0,051 | 0,051 | 0,051 |
| | 0,513 | 0,494 | 0,494 | 0,494 | 0,494 | 0,494 |
| | 0,468 | 0,455 | 0,455 | 0,455 | 0,455 | 0,455 |
| Number of iterations | | 3,5 | 3,8 | 16,6 | 10,0 | 10,0 |
| CPU time | | 9,24E-04 | 4,96E-03 | 4,73E-03 | 1,93E-01 | 5,34E-01 |
| CPU time ratio | | 1 | 5 | 5 | 205 | 567 |
| MSE | | 9,65E-03 | 9,65E-03 | 9,65E-03 | 9,66E-03 | 9,65E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 |
| | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 |
| | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 |
| | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 |
| Number of iterations | | 4,1 | 4,2 | 21,4 | 14,0 | 14,0 |
| CPU time | | 1,04E-03 | 5,32E-03 | 6,01E-03 | 4,98E-01 | 8,25E-01 |
| CPU time ratio | | 1 | 5 | 6 | 477 | 791 |
| MSE | | 8,21E-05 | 8,21E-05 | 8,21E-05 | 8,21E-05 | 8,21E-05 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error; CPU, central process unit.

of iterations can be noticed. For a fixed value of $n$, 8000 starting values (corresponding to four initialization schemes for $\hat{\phi}^{(0)}$, 2 values of $r$ and 1000 simulations) are randomly selected in the parameter space, and the MSE are quite the same even when $r$ varies from 3 to 5. It can also be noticed that the computation time does not vary too much (all the CPU times are near $10^{-3}$) and this strengthens the suggestion that the CA is robust.

*5.2.2. Comparison with other algorithms.* It can be seen that for all the scenarios, the CA, Newton-Raphson and MM are accurate. For a small value of $n$, the Nelder–Mead algorithm is as accurate as CA except in scenario where $r = 5$, $n = 5000$ and $\hat{\phi}^{(0)}$ is randomly chosen. In general, the BFGS algorithm has a higher MSE than the other methods. For example, for an initialization scheme $(I_2)$, $n = 5000$ and $r = 3$, the order of the MSE corresponding to BFGS is $10^{-1}$ while others are $10^{-5}$. This is also seen for initialization scheme $(I_3)$, $n = 5000$ and $r = 5$.

As far as the robustness is concerned, it can be seen that the CA and the NR almost have the same number of iterations (approximatively between 3 and 5), which are the lowest. The MM, BFGS and NM algorithms use approximatively three to four times more iterations than the CA.

Table III. Results for scenario $r = 3$ and $\hat{\phi}^{(0)} = (x_1 + x_2)/n$.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,527 | 0,527 | 0,527 | 0,527 | 0,527 |
| | 0,019 | 0,051 | 0,051 | 0,051 | 0,051 | 0,051 |
| | 0,513 | 0,495 | 0,495 | 0,495 | 0,495 | 0,495 |
| | 0,468 | 0,455 | 0,455 | 0,455 | 0,455 | 0,455 |
| Number of iterations | | 3,2 | 3,3 | 13,6 | 10,0 | 10,0 |
| CPU time | | 9,08E-04 | 4,26E-03 | 4,06E-03 | 1,88E-01 | 5,48E-01 |
| CPU time ratio | | 1 | 5 | 5 | 207 | 603 |
| MSE | | 9,19E-03 | 9,19E-03 | 9,19E-03 | 9,19E-03 | 9,19E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,500 | 0,500 | 0,500 | 0,639 | 0,500 |
| | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 |
| | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 |
| | 0,468 | 0,468 | 0,468 | 0,468 | 0,467 | 0,468 |
| Number of iterations | | 3,7 | 3,6 | 19 | 13,2 | 13,4 |
| CPU time | | 9,56E-04 | 4,67E-03 | 5,56E-03 | 4,75E-01 | 8,01E-01 |
| CPU time ratio | | 1 | 5 | 6 | 496 | 842 |
| MSE | | 8,09E-05 | 8,09E-05 | 8,09E-05 | 1,05E-01 | 8,11E-05 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error; CPU, central process unit.

Table IV. Results for scenario $r = 3$ and $\hat{\phi}^{(0)}$ randomly chosen.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,517 | 0,517 | 0,517 | 0,818 | 0,517 |
| | 0,019 | 0,051 | 0,051 | 0,051 | 0,088 | 0,051 |
| | 0,513 | 0,498 | 0,498 | 0,498 | 0,477 | 0,498 |
| | 0,468 | 0,451 | 0,451 | 0,451 | 0,435 | 0,451 |
| Number of iterations | | 3,5 | 3,8 | 16,4 | 8,7 | 9,1 |
| CPU time | | 9,67E-04 | 4,99E-03 | 5,08E-03 | 1,69E-01 | 5,36E-01 |
| CPU time ratio | | 1 | 5 | 5 | 174 | 554 |
| MSE | | 8,33E-03 | 8,33E-03 | 8,33E-03 | 2,54E-01 | 8,33E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,500 | 0,500 | 0,500 | 0,735 | 0,500 |
| | 0,019 | 0,019 | 0,019 | 0,019 | 0,056 | 0,019 |
| | 0,513 | 0,513 | 0,513 | 0,513 | 0,494 | 0,513 |
| | 0,468 | 0,468 | 0,468 | 0,468 | 0,450 | 0,468 |
| Number of iterations | | 4,0 | 4,3 | 21,4 | 12,5 | 12,8 |
| CPU time | | 1,02E-03 | 5,32E-03 | 5,89E-03 | 4,31E-01 | 7,59E-01 |
| CPU time ratio | | 1 | 5 | 6 | 422 | 744 |
| MSE | | 8,12E-05 | 8,12E-05 | 8,12E-05 | 1,79E-01 | 8,13E-05 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error.

Most importantly, the CA algorithm is efficient. It needs much less time than the other algorithms. Indeed, none of the CPU time ratios is lower than 1, which means that none of the algorithm needs less computation time than CA. On average and in all the cases, the CA is five to six times quicker than MM algorithm and despite having the same number of iterations, the CA is five to eight times quicker than the NR. This is understandable because at each iteration of the NR algorithm a matrix inversion is performed. The CA is approximately 174 (respectively 554) to 512 (respectively 2598) times quicker than BFGS (respectively NM).

Table V. Results for $r = 3$ and $\hat{\phi}^{(0)} = x_1 / \sum_{m=1}^{r} x_{1m}$.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,525 | 0,525 | 0,525 | 0,571 | 0,525 |
| | 0,019 | 0,051 | 0,051 | 0,051 | 0,050 | 0,051 |
| | 0,513 | 0,497 | 0,497 | 0,497 | 0,497 | 0,497 |
| | 0,468 | 0,452 | 0,452 | 0,452 | 0,452 | 0,452 |
| Number of iterations | | 3,1 | 3,2 | 14,5 | 9,8 | 9,8 |
| CPU time | | 8,86E-04 | 4,21E-03 | 4,41E-03 | 1,86E-01 | 5,55E-01 |
| CPU time ratio | | 1 | 5 | 5 | 210 | 626 |
| MSE | | 9,40E-03 | 9,40E-03 | 9,41E-03 | 3,82E-02 | 9,41E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,500 | 0,500 | 0,500 | 0,617 | 0,500 |
| | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 | 0,019 |
| | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 | 0,513 |
| | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 | 0,468 |
| Number of iterations | | 3,1 | 3,0 | 16,1 | 13,3 | 13,6 |
| CPU time | | 1,03E-03 | 4,42E-03 | 5,20E-03 | 5,26E-01 | 8,98E-01 |
| CPU time ratio | | 1 | 4 | 5 | 512 | 875 |
| MSE | | 8,24E-05 | 8,24E-05 | 8,24E-05 | 9,05E-02 | 8,24E-05 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error; CPU, central process unit.

Table VI. Results for scenario $r = 5$ and $\hat{\phi}^{(0)} = (1/r, \ldots, 1/r)^T$.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,5 | 0,533 | 0,533 | 0,533 | 0,533 | 0,533 |
| | 0,142 | 0,141 | 0,141 | 0,141 | 0,141 | 0,141 |
| | 0,003 | 0,043 | 0,043 | 0,043 | 0,043 | 0,043 |
| | 0,222 | 0,212 | 0,212 | 0,212 | 0,212 | 0,212 |
| | 0,238 | 0,230 | 0,230 | 0,230 | 0,230 | 0,230 |
| | 0,395 | 0,375 | 0,375 | 0,375 | 0,375 | 0,375 |
| Number of iterations | | 3,6 | 3,7 | 16,5 | 10,0 | 10,0 |
| CPU time | | 9,31E-04 | 7,53E-03 | 4,87E-03 | 2,46E-01 | 1,47E+00 |
| CPU time ratio | | 1 | 8 | 5 | 264 | 1578 |
| MSE | | 6,68E-03 | 6,68E-03 | 6,68E-03 | 6,68E-03 | 6,68E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 | 0,500 |
| | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 |
| | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 |
| | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 |
| | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 |
| | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 |
| Number of iterations | | 4,2 | 4,1 | 21,3 | 14,0 | 14,0 |
| CPU time | | 1,18E-03 | 8,61E-03 | 6,86E-03 | 5,32E-01 | 2,73E+00 |
| CPU time ratio | | 1 | 7 | 6 | 451 | 2311 |
| MSE | | 6,04E-05 | 6,04E-05 | 6,04E-05 | 6,04E-05 | 6,04E-05 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error; CPU, central process unit.

Table VII. Results for scenario $r = 5$ and $\hat{\phi}^{(0)} = (x_1 + x_2)/n$.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,5 | 0,533 | 0,533 | 0,533 | 0,533 | 0,533 |
| | 0,142 | 0,140 | 0,140 | 0,140 | 0,140 | 0,140 |
| | 0,003 | 0,043 | 0,043 | 0,043 | 0,043 | 0,043 |
| | 0,222 | 0,213 | 0,213 | 0,213 | 0,213 | 0,213 |
| | 0,238 | 0,227 | 0,227 | 0,227 | 0,227 | 0,227 |
| | 0,395 | 0,378 | 0,378 | 0,378 | 0,378 | 0,378 |
| Number of iterations | | 3,5 | 3,4 | 14,7 | 10,0 | 10,0 |
| CPU time | | 9,14E-04 | 6,97E-03 | 4,34E-03 | 2,41E-01 | 1,45E+00 |
| CPU time ratio | | 1 | 8 | 5 | 264 | 1589 |
| MSE | | 6,67E-03 | 6,67E-03 | 6,67E-03 | 6,67E-03 | 6,67E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,5 | 0,499 | 0,499 | 0,499 | 0,513 | 0,499 |
| | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 |
| | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 |
| | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 | 0,222 |
| | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 |
| | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 |
| Number of iterations | | 4,1 | 3,8 | 20,4 | 13,9 | 13,9 |
| CPU time | | 1,06E-03 | 7,62E-03 | 5,78E-03 | 4,81E-01 | 2,46E+00 |
| CPU time ratio | | 1 | 7 | 5 | 456 | 2336 |
| MSE | | 6,16E-05 | 6,16E-05 | 6,16E-05 | 6,27E-03 | 6,69E-05 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error; CPU, central process unit.

Table VIII. Results for scenario $r = 5$ and $\hat{\phi}^{(0)}$ randomly chosen.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,5 | 0,534 | 0,534 | 0,534 | 0,847 | 0,533 |
| | 0,142 | 0,142 | 0,142 | 0,142 | 0,150 | 0,143 |
| | 0,003 | 0,042 | 0,042 | 0,042 | 0,064 | 0,043 |
| | 0,222 | 0,215 | 0,215 | 0,215 | 0,211 | 0,217 |
| | 0,238 | 0,228 | 0,228 | 0,228 | 0,226 | 0,229 |
| | 0,395 | 0,373 | 0,373 | 0,373 | 0,349 | 0,368 |
| Number of iterations | | 3,6 | 3,8 | 16,6 | 8,7 | 9,7 |
| CPU time | | 9,59E-04 | 7,77E-03 | 4,92E-03 | 2,13E-01 | 1,46E+00 |
| CPU time ratio | | 1 | 8 | 5 | 222 | 1521 |
| MSE | | 6,51E-03 | 6,51E-03 | 6,51E-03 | 1,76E-01 | 6,62E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,500 | 0,500 | 0,500 | 0,778 | 0,500 |
| | 0,142 | 0,142 | 0,142 | 0,142 | 0,150 | 0,145 |
| | 0,003 | 0,003 | 0,003 | 0,003 | 0,027 | 0,004 |
| | 0,222 | 0,222 | 0,222 | 0,222 | 0,219 | 0,223 |
| | 0,238 | 0,238 | 0,238 | 0,238 | 0,233 | 0,239 |
| | 0,395 | 0,395 | 0,395 | 0,395 | 0,371 | 0,390 |
| Number of iterations | | 4,2 | 4,1 | 21,5 | 12,3 | 13,2 |
| CPU time | | 1,03E-03 | 8,13E-03 | 6,33E-03 | 4,33E-01 | 2,41E+00 |
| CPU time ratio | | 1 | 8 | 6 | 421 | 2340 |
| MSE | | 5,99E-05 | 5,99E-05 | 5,99E-05 | 1,41E-01 | 2,23E-04 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error; CPU, central process unit.

Table IX. Results for $r = 5$ and $\hat{\phi}^{(0)} = x_1 / \sum_{m=1}^{r} x_{1m}$.

| | | CA | NR | MM | BFGS | NM |
|---|---|---|---|---|---|---|
| $n = 50$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,537 | 0,537 | 0,537 | 0,542 | 0,537 |
| | 0,142 | 0,141 | 0,141 | 0,141 | 0,141 | 0,141 |
| | 0,003 | 0,042 | 0,042 | 0,042 | 0,042 | 0,042 |
| | 0,222 | 0,214 | 0,214 | 0,214 | 0,214 | 0,214 |
| | 0,238 | 0,228 | 0,228 | 0,228 | 0,228 | 0,228 |
| | 0,395 | 0,374 | 0,374 | 0,374 | 0,375 | 0,374 |
| Number of iterations | | 3,2 | 3,3 | 15,3 | 10,0 | 10,0 |
| CPU time | | 8,11E-04 | 6,43E-03 | 4,40E-03 | 2,32E-01 | 1,41E+00 |
| CPU time ratio | | 1 | 8 | 5 | 286 | 1739 |
| MSE | | 7,00E-03 | 7,00E-03 | 7,00E-03 | 8,86E-03 | 7,00E-03 |
| $n = 5000$ | $\Theta^0$ | | | $\hat{\Theta}$ | | |
| | 0,500 | 0,501 | 0,501 | 0,501 | 0,501 | 0,501 |
| | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 | 0,142 |
| | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 |
| | 0,222 | 0,221 | 0,221 | 0,221 | 0,221 | 0,221 |
| | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 | 0,238 |
| | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 | 0,395 |
| Number of iterations | | 3,2 | 3,0 | 16,8 | 14,0 | 14,0 |
| CPU time | | 8,75E-04 | 5,67E-03 | 4,56E-03 | 4,39E-01 | 2,27E+00 |
| CPU time ratio | | 1 | 6 | 5 | 502 | 2598 |
| MSE | | 6,09E-05 | 6,09E-05 | 6,09E-05 | 6,09E-05 | 6,09E-05 |

CA, cyclic algorithm; NR, Newton–Raphson method; BFGS, Broyden–Fletcher–Goldfarb–Shanno algorithm; NM, Nelder–Mead algorithm; MSE, mean squared error; CPU, central process unit.

## 6. CONCLUSION

In statistics, when the maximum likelihood estimation with or without constraints is considered, the Newton method and/or the Fisher score one immediately spring to mind. If, moreover, the complete expression of the likelihood function and the constraints are available, then any user will immediately use commercial or free software to try and solve the problem they meet.

In theory, any package dedicated to constrained optimization enables to find solutions. The statistical model used in this paper does not escape this rule provided the package is implemented, that is, having access to it, giving appropriate initial solutions and being able to have the Hessian matrix or an approximation. For all these reasons, we propose, under certain regularity conditions, an estimation method that generalizes and completes the results of [16].

We present some numerical properties of our cyclic iterative algorithm for the estimation of the parameters of a multinomial model. Then, we applied it to the modelling of the mean effect of a road safety measure on accident risk and crash data via a multinomial distribution. This algorithm is very simple to program without any matrix inversion and it integrates the inequality or equality constraints easily. The results presented in this paper suggest that this algorithm is robust towards the starting values, efficient and accurate. Moreover, the comparison of the performance of the cyclic algorithm with some of the best available optimization algorithms suggest that it is as accurate as the others and most importantly that it is much faster as far as the convergence is concerned.

## APPENDIX A. PROOFS

### *A.1. Details of Theorem 1's proof*

$D_{\hat{\theta},x}$ exists by assumption and $\frac{\partial h}{\partial \phi_j} \neq 0$. Consequently, the inversion of matrix $\Omega_{\theta,x}$ of Theorem 1 is possible because the Schur complement of $D_{\hat{\theta},x}$ in $\Omega_{\hat{\theta},x}$ noted $(\Omega_{\hat{\theta},x}|D_{\hat{\theta},x})$ exists and its inversion is possible with $(\Omega_{\hat{\theta},x}|D_{\hat{\theta},x}) = (\nabla_\phi h)^T_{\hat{\Theta}} D^{-1}_{\hat{\theta},x} (\nabla_\phi h)_{\hat{\Theta}} \neq 0$. We then know that (for example, [17, 18, 25])

$$\Omega^{-1}_{\hat{\theta},x} = \begin{pmatrix} M_{\hat{\theta},x} & -D^{-1}_{\hat{\theta},x}(\nabla_\phi h)_{\hat{\Theta}}(\Omega_{\hat{\theta},x}|D_{\hat{\theta},x})^{-1} \\ -(\Omega_{\hat{\theta},x}|D_{\hat{\theta},x})^{-1}(\nabla_\phi h)^T_{\hat{\Theta}} D^{-1}_{\hat{\theta},x} & (\Omega_{\hat{\theta},x}|D_{\hat{\theta},x})^{-1} \end{pmatrix}.$$

By multiplying $\Omega_{\hat{\theta},x}$ with $\left(B_x^T, \eta\right)^T$, and using $\eta = (\nabla_\phi h)^T_{\hat{\Theta}} \hat{\phi}$, we show that

$$0 = \|(\nabla_\phi h)_{\hat{\Theta}}\|^{-2}_{D^{-1}_{\hat{\theta},x}} (\nabla_\phi h)^T_{\hat{\Theta}} D^{-1}_{\hat{\theta},x} B_x - \|(\nabla_\phi h)_{\hat{\Theta}}\|^{-2}_{D^{-1}_{\hat{\theta},x}} \eta$$
$$\hat{\phi} = D^{-1}_{\hat{\theta},x} B_x.$$

*A.2. Details of Corollary 1's proof*

Let us set

$$\Sigma_\theta = \begin{pmatrix} \Lambda_{\theta,Z} & \sqrt{\theta} B_x \\ \sqrt{\theta} Z^T & 1 \end{pmatrix}.$$

As $\Lambda^{-1}_{\theta,Z}$ exists, then the Schur complement of $\Lambda_{\theta,Z}$ in $\Sigma_\theta$ exists as well as the Schur complement of 1 in $\Sigma_\theta$. We then have $D_{\theta,x} = (\Sigma_\theta|1) = \Lambda_{\theta,Z} - \theta B_x Z^T$ and

$$(\Sigma_\theta|1)^{-1} = \Lambda^{-1}_{\theta,Z} + \Lambda^{-1}_{\theta,Z}\theta^{1/2} B_x (\Sigma_\theta|\Lambda_{\theta,Z})^{-1}\theta^{1/2} Z^T \Lambda^{-1}_{\theta,Z}$$

where $(\Sigma_\theta|\Lambda_{\theta,Z})^{-1} = \left(1 - \theta Z^T \Lambda^{-1}_{\theta,Z} B_x\right)^{-1} > 0$. We then deduce that

$$\hat{\phi} = D^{-1}_{\hat{\theta},x} B_x = \left(1 - \hat{\theta}\langle Z, B_x\rangle_{\Lambda^{-1}_{\hat{\theta},Z}}\right)^{-1} \Lambda^{-1}_{\theta,Z} B_x.$$

ACKNOWLEDGEMENTS

We thank the editor and the anonymous referees for their remarks and suggestions, which enabled a substantial improvement of this paper. The article was partially written while the first author was visiting CIRRELT of University of Montreal, Canada. The second author was partially supported by the "Groupe d'Intérêt Economique" (GIE) of PSA-RENAULT of France under agreement No. USTL-GIE PSA-RENAULT 248 02 515.

REFERENCES

1. Lange K. *Optimization* (2nd edn). Springer: New York, 2013.
2. Lange K. *Numerical Analysis for Statisticians* (2nd edn). Springer: New York, 2010.
3. Lange K, Chi EC, Zhou H. A brief survey of modern optimization for statisticians. *International Statistical Review* 2014; **82**(1):46–70.
4. Nocedal J, Wright SJ. *Numerical Optimization* (2nd edn). Springer: New York, 2006.
5. Broyden CG. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications* 1970; **6**:76–90.
6. Fletcher R. A new approach to variable metric algorithms. *The Computer Journal* 1970; **13**(3):317–322.
7. Goldfarb D. A family of variable metric updates derived by variational means. *Mathematics of Computation* 1970; **24**(109):23–26.
8. Shanno DF. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation* 1970; **24**(111):647–656.
9. Nelder JA, Mead R. A simplex algorithm for function minimization. *Computer Journal* 1965; **7**(4):308–313.
10. El Barmi H, Dykstra RL. Maximum likelihood estimates via duality for log-convex models when cell probabilities are subject to convex constraints. *The Annals of Statistics* 1998; **26**(5):1878–1893.

11. Matthews GB, Crowther NAS. A maximum likelihood estimation procedure when modelling in terms of constraints. *South African Statistical Journal* 1995; **29**(1):29–51.
12. Liu C. Estimation of discrete distributions with a class of simplex constraints. *Journal of the American Statistical Association* 2000; **95**(449):109–120.
13. Zhou H, Lange K. MM algorithms for some discrete multivariate distributions. *Journal of Computational and Graphical Statistics* 2010; **19**(3):645–665.
14. Mkhadri A, N'Guessan A, Hafidi B. An MM algorithm for constrained estimation in a road safety measure modeling. *Communications in Statistics - Simulation and Computation* 2010; **39**(5):1057–1071.
15. N'Guessan A, Truffier M. Impact d'un aménagement de sécurité routière sur la gravité des accidents de la route. *Journal de la Société Française de Statistiques* 2008; **149**(3):23–41.
16. N'Guessan A. Analytical existence of solutions to a system of non-linear equations with application. *Journal of Computational and Applied Mathematics* 2010; **234**:297–304.
17. Ouellette DV. Schur complement and statistics. *Linear Algebra and Its Applications* 1981; **36**:187–295.
18. Zhang F. *The Schur Complement and its Applications*. Springer: US, 2005.
19. N'Guessan A, Langrand C. A Schur complement approach for computing subcovariance matrices arising in a road safety measure modelling. *Journal of Computational and Applied Mathematics* 2005; **177**(2):331–345.
20. Hunter DR, Lange K. A tutorial on MM algorithms. *The American Statistician* 2004; **58**(1):30–37.
21. N'Guessan A, Essai A, Langrand C. Estimation multidimensionnelle des contrôles et de l'effet moyen d'une mesure de sécurité routière. *Revue de Statistique Appliquée* 2001; **49**(2):85–102.
22. Gokhale DV. Iterative maximum likelihood for discrete distributions. *Sankhya: The Indian Journal of Statistics, Series B (1960-2002)* 1973; **35**(3):293–298.
23. R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, 2013. (Available from: http://www.R-project.org/) [Accessed on 20 May 2015].
24. Varadhan R. alabama: Constrained nonlinear optimization. 2010. (Available from: http://R-Forge.R-project.org/projects/optimizer/) [Accessed on 20 May 2015], R package version 2010.7-1/r376.
25. N'Guessan A, Langrand C. A covariance components estimation procedure when modelling a road safety measure in terms of linear constraints. *Statistics: A Journal of Theoretical and Applied Statistics* 2005; **39**(4):303–314.

# Appendix B

# Article extracted from Chapter 3

**I. C. Geraldo, A. N'Guessan and K. E. Gneyou (2015). A note on the strong consistency of a constrained maximum likelihood estimator used in crash data modeling, *Comptes Rendus Mathematique*, Vol. 353, No 12, pp. 1147-1152.**

C. R. Acad. Sci. Paris, Ser. I 353 (2015) 1147–1152

Contents lists available at ScienceDirect

C. R. Acad. Sci. Paris, Ser. I

www.sciencedirect.com

Probability theory/Statistics

# A note on the strong consistency of a constrained maximum likelihood estimator used in crash data modeling

*À propos de la consistance forte d'un estimateur du maximum de vraisemblance sous contraintes utilisé dans la modélisation des données d'accidents*

Issa Cherif Geraldo [a,b], Assi N'Guessan [b], Kossi Essona Gneyou [a,c]

[a] *Département de mathématiques et informatique, Université catholique de l'Afrique de l'Ouest, Unité universitaire du Togo (UCAO–UUT), 01 B.P. 1502 Lomé 01, Lomé, Togo*
[b] *Laboratoire Paul-Painlevé, UMR CNRS 8524, Université de Lille-1, 59655 Villeneuve d'Ascq cedex, France*
[c] *Département de mathématiques, Faculté des sciences, Université de Lomé, B.P. 1515 Lomé, Togo*

A R T I C L E I N F O

*Article history:*
Received 1 May 2015
Accepted after revision 28 September 2015
Available online 30 October 2015

Presented by the Editorial Board

A B S T R A C T

In this note, we consider the Maximum Likelihood Estimator (MLE) of the vector parameter $\Phi = (\theta, \phi^{\mathrm{T}})^{\mathrm{T}}$ of dimension $R$ $(R > 1)$ used in crash-data modeling where $\theta > 0$ and $\phi$ belongs to the simplex of order $R - 1$. We prove the strong consistency of this constrained estimator making capital out of the cyclic form between the components of the MLE.

© 2015 Académie des sciences. Published by Elsevier Masson SAS. All rights reserved.

R É S U M É

Dans cette note, nous considérons l'estimateur du maximum de vraisemblance (EMV) du vecteur paramètre $\Phi = (\theta, \phi^{\mathrm{T}})^{\mathrm{T}}$ de dimension $R$ $(R > 1)$ utilisé dans la modélisation des données d'accidents où $\theta > 0$ et $\phi$ appartient au simplexe d'ordre $R - 1$. Nous démontrons la consistance forte de cet estimateur sous contraintes en exploitant la forme cyclique entre les composantes de cet estimateur.

© 2015 Académie des sciences. Published by Elsevier Masson SAS. All rights reserved.

## 1. Introduction

Let $X$ be a $\mathbb{R}^d$ valued random variable defined on a probability space $(\Omega, \mathcal{A}, \mathrm{P})$ with a probability density function depending on a vector parameter $\Phi$. The Maximum Likelihood Estimator (MLE) $\hat{\Phi}_n$ of $\Phi$ can be obtained by solving the optimization problem

$$\hat{\Phi}_n = \arg\max_{\Phi \in S} L(\Phi)$$

where $L$ is the log-likelihood function calculated on a sample of $n$ i.i.d. observations of $X$ and $S$ is the parameter space (the set of all possible values of $\Phi$).

*E-mail addresses:* cherifgera@gmail.com (I.C. Geraldo), assi.nguessan@polytech-lille.fr (A. N'Guessan), kossi_gneyou@yahoo.fr (K.E. Gneyou).

http://dx.doi.org/10.1016/j.crma.2015.09.025
1631-073X/© 2015 Académie des sciences. Published by Elsevier Masson SAS. All rights reserved.

One of the most desired properties of the estimator $\hat{\Phi}_n$ is its consistency, i.e. its asymptotic convergence to the true value $\Phi^0$. This property was studied in the literature by many authors (see, e.g., [2,6,7,14,15]). The main result on the strong consistency was established by Wald [15], who gave regularity conditions under which the MLE is strongly consistent. However, all these conditions may be hard to check in practice if the dimension of the parameter space is large and the probability density function (or the likelihood) takes some complex forms. Nevertheless, introducing modifications in Wald's work, some authors, among which [5,6,12,14], obtained useful results on the consistency under less restrictive conditions. Van der Vaart [14] established general consistency properties of $M$-estimators presenting the MLE as a special case of $M$-estimators. But it is still possible that the MLE is not consistent even when it exists, as shown by the examples given in [4].

The present work is motivated by our need to provide a proof of the strong convergence property of the MLE of $\Phi$ proposed in [10,11] for statistical analysis of accident data on an experimental site where observed accidents can be classified into $R$ mutually exclusive categories, $R \in \mathbf{N}^*$. In their before–after study in order to assess the impact of a measure on the occurrence of accidents, N'Guessan et al. [10,11] considered a random vector $X = (X_{11}, \ldots, X_{1R}, X_{21}, \ldots, X_{2R})^{\mathrm{T}}$ whose components are positive non-zero discrete random variables such that $X_{1r}$ (resp. $X_{2r}$), $r = 1, \ldots, R$, represents the number of crashes of type $r$ that have occurred in the "before" (resp. "after") period. This model also integrates a vector of known non-random components denoted by $C = (c_1, \ldots, c_R)^{\mathrm{T}}$. It is assumed that $X$ follows the multinomial distribution $X \sim \mathcal{M}(n; \pi_1(\Phi), \pi_2(\Phi))$ where $n$ denotes a positive integer representing the total number of independent accidents in both before and after periods, that is $\sum_{t=1}^{2}\sum_{r=1}^{R} X_{\mathrm{tr}} = n$. Here $\pi_1(\Phi) = (\pi_{11}(\Phi), \ldots, \pi_{1R}(\Phi))^{\mathrm{T}}$, $\pi_2(\Phi) = (\pi_{21}(\Phi), \ldots, \pi_{2R}(\Phi))^{\mathrm{T}}$ with

$$\pi_{1r}(\Phi) = \frac{\phi_r}{1 + \theta \sum_{j=1}^{R} c_j\phi_j}, \quad \pi_{2r}(\Phi) = \frac{\theta\, c_r\phi_r}{1 + \theta \sum_{j=1}^{R} c_j\phi_j}, \qquad \forall r = 1, \ldots, R \tag{1}$$

and $\sum_{t=1}^{2}\sum_{r=1}^{R} \pi_{\mathrm{tr}}(\Phi) = 1$. The random vector $X$ has a probability density function depending on a multidimensional parameter $\Phi = (\theta, \phi^{\mathrm{T}})^{\mathrm{T}}$, where $\theta \in \mathbb{R}_+^*$ and $\phi = (\phi_1, \ldots, \phi_R)^{\mathrm{T}}$ satisfies $\sum_{r=1}^{R} \phi_r = 1$ and belongs to the simplex of dimension $R-1$. The scalar $\theta$ represents the unknown average effect of the road safety measure, while each $\phi_r$ ($r = 1, 2, \ldots, R$) denotes the global accident risk of type $r$ before and after the application of the road safety measure. The coefficients $c_1, \ldots, c_R$ are given positive real numbers.

The existence of the constrained MLE $\hat{\Phi}_n$ of model (1) has been studied by [8] and an application is given in [11]. The numerical convergence properties of $\hat{\Phi}_n$ to the true values were recently studied by N'Guessan and Geraldo [10] using intensive simulation studies. They found that the MLE $\hat{\Phi}_n$ given by (2) converges numerically to the true value of the parameter whenever $n$ tends to $+\infty$. We then make up their results by showing the strong convergence of $\hat{\Phi}_n$ given by (2) below.

So the aim of this note is to give a theoretical proof of the strong convergence of the estimator $\hat{\Phi}_n$ when $n$ tends to infinity. The remainder of the note is organized as follows. In Section 2, we give some preliminary results. The main results on the strong convergence of the MLE in the crash control model are presented in Section 3. These main results are divided into three theorems. In Section 4, we present some concluding remarks.

## 2. Preliminary results

Throughout the paper, the subscript $n$ is used to indicate that the estimators depend on the sample size $n$. It is proven in [9] that the log-likelihood associated with an observed data $x = (x_{11}, \ldots, x_{1r}, x_{21}, \ldots, x_{2r})$ satisfying $\sum_{t=1}^{2}\sum_{r=1}^{R} x_{\mathrm{tr}} = n$, is defined up to an additive constant by $\ell(\Phi) = \sum_{r=1}^{R}[\underline{x}_r \log(\phi_r) + x_{2r}\log(\theta) - \underline{x}_r \log(1 + \theta \sum_{m=1}^{R} c_m\phi_m)]$ where $\underline{x}_r = x_{1r} + x_{2r}$, $r = 1, \ldots, R$. The MLE $\hat{\Phi}_n$ of $\Phi$ is then given by the following lemma.

**Lemma 2.1.** *(See [8].) The components $\hat{\theta}_n$ and $\hat{\phi}_n$ of the MLE $\hat{\Phi}_n$ satisfy*

$$\begin{cases} \hat{\theta}_n &= \dfrac{\sum_{m=1}^{R} X_{2m}}{\left(\sum_{m=1}^{R} c_m\,\hat{\phi}_{n,m}\right)\times\left(\sum_{m=1}^{R} X_{1m}\right)} \\ \hat{\phi}_{n,r} &= \dfrac{1}{1 - \frac{1}{n}\sum_{m=1}^{R}\frac{\hat{\theta}_n c_m \underline{X}_m}{1+\hat{\theta}_n c_m}} \times \dfrac{\underline{X}_r}{n(1+\hat{\theta}_n\, c_r)}, \qquad r = 1, 2, \ldots, R \end{cases} \tag{2}$$

*with $\underline{X}_r = X_{1r} + X_{2r}$, $r = 1, \ldots, R$.*

**Proof.** Introducing a Lagrange multiplier in order to cope with the linear constraint $\sum_{r=1}^{R} \phi_r = 1$, $\hat{\Phi}_n$ is obtained as a solution of the non-linear system:

$$\begin{cases} \displaystyle\sum_{r=1}^{R}\left[x_{2r} - \underline{x}_r \frac{\hat{\theta}_n \sum_{m=1}^{R} c_m\hat{\phi}_{n,m}}{1 + \hat{\theta}_n \sum_{m=1}^{R} c_m\hat{\phi}_{n,m}}\right] = 0 \\ \underline{x}_r - n\dfrac{\hat{\phi}_{n,r}(1 + c_r\hat{\theta}_n)}{1 + \hat{\theta}_n \sum_{m=1}^{R} c_m\hat{\phi}_{n,m}} = 0, \quad r = 1, 2, \ldots, R. \end{cases} \tag{3}$$

The first line of System (2) follows from the first line of (3). The expression of $\hat{\phi}_n$ is obtained by transforming the second line of (3) into a linear system whose unique vector solution is $\hat{\phi}_n$. For further details, refer to [8,11] and the references therein. □

Let us recall some important lemmas that will be the key for establishing our strong convergence results. The first lemma is provided by the continuous mapping theorem of [14, p. 7]. The second is due to the strong law of large numbers. The third lemma states conditions under which the convergence of a sequence of injective functions $(f_n)$ implies that of their inverses $(f_n^{-1})$ [1, Theorem 2].

**Lemma 2.2.** *(See [14].) Let $g : \mathbb{R}^k \to \mathbb{R}^m$ be continuous at every point of a Borel set $A$ such that $\mathrm{P}(X \in A) = 1$. If $X_n$ converges almost surely (a.s.) to $X$, then $g(X_n)$ converges almost surely to $g(X)$.*

**Lemma 2.3.** *If the random vector $X = (X_{11}, \ldots, X_{1R}, X_{21}, \ldots, X_{2R})$ has the multinomial distribution $\mathcal{M}(n; \pi)$ with $\pi = (\pi_{11}, \ldots, \pi_{1R}, \pi_{21}, \ldots, \pi_{2R})$ then, as $n \to +\infty$: $\frac{1}{n} X \xrightarrow{a.s.} (\pi_{11}, \ldots, \pi_{1R}, \pi_{21}, \ldots, \pi_{2R})$.*

**Proof.** The vector $X$ can be thought of as the sum of $n$ independent multinomial vectors $Y_1, \ldots, Y_n$ with parameters 1 and $\pi$. Then, the mathematical expectation of $Y_i$ is $E(Y_i) = \pi$. By the strong law of large numbers, the random sequence $n^{-1} X$ converges almost surely to $\pi$. □

The following lemmas are related to the uniform convergence of sequences of functions on a metric space.

**Lemma 2.4.** *(See [1].) If $(f_n)$ is a sequence of injection mappings on a metric space $E$, taking values in a locally compact metric space $G$ and converging uniformly to $f$ on $E$ and if $f^{-1}$ is a continuous mapping on $G_1 \subset G$, then $f_n^{-1}$ converges uniformly to $f^{-1}$ on every compact set contained in $\mathrm{int}(G_1) \cap (\cap_n f_n(E))$ where $\mathrm{int}(G_1)$ denotes the interior of $G_1$.*

**Lemma 2.5.** *(See [13].) Let $f_n$ be a sequence of continuous functions on a set $D$. If $f_n$ converges uniformly to $f$, then $f_n(u_n)$ converges to $f(u)$ for all sequences $u_n$ in $D$ convergent to $u \in D$.*

## 3. Main results

**Theorem 3.1.** *As $n$ tends to $+\infty$, the random variable $\hat{\theta}_n$ converges a.s. to $\theta^0$ if and only if the random vector $\hat{\phi}_n$ converges a.s. to $\phi^0$.*

**Proof.** We know that $\hat{\phi}_n = (\hat{\phi}_{n,1}, \ldots, \hat{\phi}_{n,R}) \in \mathbb{R}^R$ converges a.s. to $\phi^0 = (\phi_1^0, \ldots, \phi_R^0) \in \mathbb{R}^R$ if and only if, for all $r = 1, \ldots, R$, $\hat{\phi}_{n,r} \to \phi_r^0$ a.s. Thus, it is sufficient to prove that for all $r = 1, \ldots, R$, $\hat{\theta}_n \longrightarrow \theta^0$ *a.s.* implies that $\hat{\phi}_{n,r} \longrightarrow \phi_r^0$ *a.s.* Now let us suppose that $\hat{\theta}_n \to \theta^0$ a.s. Observing that $\sum_{m=1}^R \underline{X}_m = n$, we get

$$\hat{\phi}_{n,r} = \frac{(X_{1r} + X_{2r})/(1 + \hat{\theta}_n c_r)}{\sum_{m=1}^R (X_{1m} + X_{2m})/(1 + \hat{\theta}_n c_m)}. \tag{4}$$

Moreover, we can write $\hat{\phi}_{n,r} = g_r(\frac{X_{11}}{n}, \ldots, \frac{X_{1R}}{n}, \frac{X_{21}}{n}, \ldots, \frac{X_{2R}}{n}, \hat{\theta}_n)$ where $g_r$ is the continuous function from $\mathbb{R}^{2R+1}$ to $\mathbb{R}$ defined by $(b_1, \ldots, b_R, a_1, \ldots, a_R, \theta) \mapsto \frac{(b_r + a_r)/(1+\theta c_r)}{\sum_{m=1}^R (b_m + a_m)/(1+\theta c_m)}$. Using Lemma 2.3, we have almost surely

$$\left(\frac{X_{11}}{n}, \ldots, \frac{X_{1R}}{n}, \frac{X_{21}}{n}, \ldots, \frac{X_{2R}}{n}, \hat{\theta}_n\right) \to (\pi_{11}^0, \ldots, \pi_{1R}^0, \pi_{21}^0, \ldots, \pi_{2R}^0, \theta^0)$$

as $n \to \infty$. Applying the continuous mapping theorem (Lemma 2.2) and relations (1), we get as $n \to +\infty$,

$$\hat{\phi}_{n,r} \quad \to \quad \frac{(\pi_{1r}^0 + \pi_{2r}^0)/(1 + \theta^0 c_r)}{\sum_{m=1}^R (\pi_{1m}^0 + \pi_{2m}^0)/(1 + \theta^0 c_m)} = \phi_r^0 \quad \text{a.s.}$$

This proves that $\hat{\phi}_{n,r}$ converges to $\phi_r^0$ a.s.

Now let us assume that $\hat{\phi}_n \longrightarrow \phi^0$ a.s. or equivalently $\hat{\phi}_{n,r} \longrightarrow \phi_r^0$ a.s. for all $r = 1, \ldots, R$. From Lemma 2.1, we get

$$\hat{\theta}_n = \frac{\sum_{m=1}^R (X_{2m}/n)}{\sum_{m=1}^R (X_{1m}/n)} \times \frac{1}{\sum_{m=1}^R c_m \hat{\phi}_{n,m}} = g\left(\frac{X_{11}}{n}, \ldots, \frac{X_{1R}}{n}, \frac{X_{21}}{n}, \ldots, \frac{X_{2R}}{n}, \hat{\phi}_{n,1}, \ldots, \hat{\phi}_{n,R}\right)$$

where $g$ is the continuous function defined from $\mathbb{R}^{3R}$ to $\mathbb{R}$ by

$$(b_1, \ldots, b_R, a_1, \ldots, a_R, \phi_1, \ldots, \phi_R) \quad \mapsto \quad \frac{\sum_{m=1}^R a_m}{\sum_{m=1}^R b_m} \times \frac{1}{\sum_{m=1}^R c_m \phi_m}.$$

We again apply Lemmas 2.2 and 2.3 and get

$$\hat{\theta}_n \xrightarrow[n\to+\infty]{a.s.} g(\pi^0_{11},\ldots\pi^0_{1R},\pi^0_{21},\ldots,\pi^0_{2R},\phi^0_1,\ldots,\phi^0_R)=\theta^0. \qquad \square$$

Theorem 3.1 shows that the almost sure convergence of $\hat{\phi}_n$ to $\phi^0$ is equivalent to the almost sure convergence of $\hat{\theta}_n$ to $\theta^0$. To prove that the MLE $\hat{\Phi}_n=(\hat{\theta}_n,\hat{\phi}_n^{\mathrm{T}})^{\mathrm{T}}$ converges almost surely, it is then sufficient by Theorem 3.1 to prove for example that $\hat{\theta}_n$ converges almost surely to $\theta^0$. With that in mind, we first prove that the a.s. limit of $\hat{\theta}_n$ exists and then show that this a.s. limit is equal to $\theta^0$.

The following result shows the almost sure convergence of $\hat{\theta}_n$.

**Theorem 3.2.** *There exists a constant $\mu>0$ and a subset $N\subset\Omega$ such that $\mathrm{P}(N)=0$ and*

$$\forall\omega\in\Omega\setminus N,\quad \lim_{n\to\infty}\hat{\theta}_n(\omega)=\mu. \tag{5}$$

**Proof.** Set $\varphi_n(u)=\sum_{m=1}^R\frac{\underline{X}_m/n}{1+u\,c_m}$, $u\in]0,+\infty[$ and for all $t=1,2$, denote $X_{t+}=\sum_{m=1}^R X_{tm}$.

We first show that for all $\omega\in\Omega$, the real valued function $\varphi_{\omega,n}(u)=\varphi_n(u)(\omega)=\displaystyle\sum_{m=1}^R\frac{\underline{X}_m(\omega)/n}{1+uc_m}$ is a continuous bijective mapping from $]0,+\infty[$ to $]0,1[$ and that $\hat{\theta}_n(\omega)=\varphi^{-1}_{\omega,n}(X_{1+}(\omega)/n)$.

By Equation (4), we have the relationship $\hat{\phi}_{n,r}=\frac{\underline{X}_r/(1+\hat{\theta}_nc_r)}{\sum_{m=1}^R\underline{X}_m/(1+\hat{\theta}_nc_m)}$ which enables to write

$$\sum_{r=1}^R c_r\hat{\phi}_{n,r}=\frac{\sum_{r=1}^R\left(c_r\underline{X}_r/(1+\hat{\theta}_nc_r)\right)}{\sum_{m=1}^R\left(\underline{X}_m/(1+\hat{\theta}_nc_m)\right)}.$$

By the first line of System (2) in Lemma 2.1, we have

$$\hat{\theta}_n=\frac{X_{2+}}{X_{1+}}\frac{1}{\sum_{m=1}^R c_m\hat{\phi}_{n,m}}=\frac{X_{2+}}{X_{1+}}\frac{\sum_{m=1}^R\left(\underline{X}_m/(1+\hat{\theta}_nc_m)\right)}{\sum_{r=1}^R\left(c_r\underline{X}_r/(1+\hat{\theta}_nc_r)\right)}.$$

This is equivalent to

$$\frac{X_{2+}}{X_{1+}}\sum_{m=1}^R\frac{\underline{X}_m}{1+\hat{\theta}_nc_m}=\sum_{m=1}^R\frac{\hat{\theta}_nc_m\underline{X}_m}{1+\hat{\theta}_nc_m}.$$

We then deduce that $\sum_{m=1}^R\frac{\underline{X}_m}{1+\hat{\theta}_nc_m}=X_{1+}$. Divide the last equality by the sample size $n$ and get

$$\sum_{m=1}^R\frac{\underline{X}_m(\omega)/n}{1+\hat{\theta}_n(\omega)\,c_m}=\frac{X_{1+}(\omega)}{n},\qquad\forall\omega\in\Omega. \tag{6}$$

It is obvious that the random real function $\varphi_{\omega,n}(u)$ has a strictly negative derivative with respect to $u$ and satisfies for all $\omega\in\Omega$:

$$1=\lim_{u\to0}\varphi_{\omega,n}(u)=\sum_{m=1}^R\underline{X}_m(\omega)/n\qquad 0=\lim_{u\to+\infty}\varphi_{\omega,n}(u).$$

Hence $\varphi_{\omega,n}(u)$ is a continuous and bijective mapping from $]0,+\infty[$ to $]0,1[$ and since $X_{1+}(w)/n\in\ ]0,1[$, Equation (6) yields $\hat{\theta}_n(\omega)=\varphi^{-1}_{\omega,n}(X_{1+}(\omega)/n)$.

Let us now prove that there exists a subset $N\subset\Omega$ with $\mathrm{P}(N)=0$ such that for all $\omega\in\Omega\setminus N$, the sequence of real functions $\varphi_{\omega,n}(u)$ converges uniformly to some function $\varphi(u)$ on $]0,+\infty[$. The almost sure convergence of the statistic $\varphi_n(u)=\displaystyle\sum_{m=1}^R\frac{\underline{X}_m/n}{1+uc_m}$ to $\varphi(u)$ will then follow.

For all $m=1,\ldots,R$, write $\frac{\underline{X}_m}{n}=g_m(\frac{X_{11}}{n},\ldots,\frac{X_{1R}}{n},\frac{X_{21}}{n},\ldots,\frac{X_{2R}}{n})$ where $g_m$ is the continuous mapping defined from $\mathbb{R}^{2R}$ to $\mathbb{R}$ by $g_m(b_1,\ldots,b_R,a_1,\ldots,a_R)=b_m+a_m$. Applying Lemmas 2.2 and 2.3, we get

$$\frac{\underline{X}_m}{n}\xrightarrow{a.s.}\alpha^0_m=g_m(\pi^0_{11},\ldots,\pi^0_{1R},\pi^0_{21},\ldots,\pi^0_{2R})=\frac{(1+\theta^0c_m)\phi^0_m}{1+\theta^0\sum_{k=1}^Rc_k\phi^0_k}.$$

Equivalently [3, p. 68], there exists a null set $N_m$ such that

$$\forall \omega \in \Omega \setminus N_m, \quad \lim_{n\to\infty} \frac{\underline{X}_m(\omega)}{n} = \alpha_m^0. \tag{7}$$

The set $E_1 = \cup_{m=1}^R N_m$ satisfies $\mathrm{P}(E_1) = 0$ and

$$\forall \omega \in \Omega \setminus E_1, \quad \lim_{n\to\infty} \varphi_{\omega,n}(u) = \lim_{n\to\infty} \sum_{m=1}^R \frac{\underline{X}_m(\omega)/n}{1+u\,c_m} = \sum_{m=1}^R \frac{\alpha_m^0}{1+u\,c_m} = \varphi(u). \tag{8}$$

Thus we have proved that for all $\omega \in \Omega \setminus E_1$, the sequence of functions $\varphi_{\omega,n}$ converges simply to $\varphi$ on $]0,+\infty[$. Moreover,

$$\begin{aligned}\sup_{u\in]0,+\infty[} |\varphi_{\omega,n}(u) - \varphi(u)| &= \sup_{u\in]0,+\infty[} \left| \sum_{m=1}^R \frac{\underline{X}_m(\omega)/n}{1+u\,c_m} - \sum_{m=1}^R \frac{\alpha_m^0}{1+u\,c_m} \right| \leqslant \sup_{u\in]0;+\infty[} \sum_{m=1}^R \left| \frac{\underline{X}_m(\omega)/n - \alpha_m^0}{1+u\,c_m} \right| \\ &= \sup_{u\in]0,+\infty[} \sum_{m=1}^R \frac{|\underline{X}_m(\omega)/n - \alpha_m^0|}{1+u\,c_m} \leqslant \sum_{m=1}^R |\underline{X}_m(\omega)/n - \alpha_m^0|\end{aligned}$$

because $\forall u \in ]0,+\infty[$, $\forall c_m > 0$, $\frac{1}{1+u\,c_m} \leqslant 1$. It follows by (7) that

$$\sup_{u\in]0,+\infty[} |\varphi_{\omega,n}(u) - \varphi(u)| \leqslant \sum_{m=1}^R |\underline{X}_m(\omega)/n - \alpha_m^0| \longrightarrow 0 \text{ as } n \to +\infty.$$

This proves the uniform convergence of the sequence $\varphi_{\omega,n}$ to $\varphi$ on $]0,+\infty[$. □

**Remark 1.** In summary, we have proved that $\varphi_{\omega,n}$ is a sequence of bijective functions taking values in $]0,1[$ that is locally compact. Moreover, $\varphi_{\omega,n}$ converges uniformly to $\varphi$ and $\varphi^{-1}$ is continuous (as the inverse of a non-zero continuous function). Thus the conditions of Lemma 2.4 are satisfied and hence the sequence $\varphi_{\omega,n}^{-1}$ converges uniformly to $\varphi^{-1}$. Moreover, the sequence $X_{1+}/n$ satisfies

$$\frac{X_{1+}}{n} = \tilde{g}\left(\frac{X_{11}}{n}, \dots, \frac{X_{1R}}{n}, \frac{X_{21}}{n}, \dots, \frac{X_{2R}}{n}\right)$$

where $\tilde{g}$ is the continuous mapping defined from $\mathbb{R}^{2R}$ to $\mathbb{R}$ by $\tilde{g}(b_1,\dots,b_R,a_1,\dots,a_R) = \sum_{m=1}^R b_m$. Apply again Lemmas 2.2 and 2.3 and get

$$\frac{X_{1+}}{n} \xrightarrow{a.s.} \tilde{g}(\pi_{11}^0,\dots,\pi_{1R}^0,\pi_{21}^0,\dots,\pi_{2R}^0) = \frac{1}{1+\theta^0 \sum_{k=1}^R c_k \phi_k^0}.$$

That is, there exists a null set $E_2$ such that

$$\forall \omega \in \Omega \setminus E_2, \lim_{n\to\infty} \frac{X_{1+}(\omega)}{n} = \gamma^0 = \frac{1}{1+\theta^0 \sum_{k=1}^R c_k \phi_k^0}.$$

The set $N = E_1 \cup E_2$ satisfies $\mathrm{P}(N) = 0$ and for all $\omega \in \Omega \setminus N$, the sequence $X_{1+}(\omega)/n$ is convergent and hence is bounded. That is, there exists a compact set $D \subset ]0,1[$ such that $X_{1+}(\omega)/n \in D, \forall n > 0$. Moreover, since

$$\hat{\theta}_n(\omega) = \varphi_{\omega,n}^{-1}\left(\frac{X_{1+}(\omega)}{n}\right)$$

and $\varphi_{\omega,n}^{-1}$ converges uniformly to $\varphi$, we apply Lemma 2.5 to conclude that

$$\forall \omega \in \Omega \setminus N, \ \hat{\theta}_n(\omega) \longrightarrow \mu = \varphi^{-1}(\gamma^0) \text{ as } n \to \infty \tag{9}$$

where $\gamma^0$ is given above. This ends the proof of Theorem 3.2.

**Theorem 3.3.** *Set $\beta^0 = \sum_{r=1}^R c_r \phi_r^0$. Let $F_{\theta^0}$ be the function from $\mathbb{R}_+$ to $\mathbb{R}$ defined by:*

$$F_{\theta^0}(u) = u\left(\sum_{m=1}^R \frac{c_m(1+\theta^0 c_m)\phi_m^0}{1+uc_m}\right) - \theta^0\beta^0\left(\sum_{m=1}^R \frac{(1+\theta^0 c_m)\phi_m^0}{1+uc_m}\right).$$

*Then,*

*i) the function $F_{\theta^0}$ has $\theta^0$ as unique root on $\mathbb{R}_+$.*
*ii) the almost sure limit $\mu$ of $\hat{\theta}_n$ is equal to the unique root $\theta^0$ of $F_{\theta^0}$ on $\mathbb{R}_+$.*

**Proof.** i) We have $F_{\theta^0}(\theta^0) = 0$ and $F'_{\theta^0}(u) > 0$. So $F_{\theta^0}$ is strictly monotone and satisfies

$$\lim_{u\to 0} F_{\theta^0}(u) < 0 \quad \text{and} \quad \lim_{u\to +\infty} F_{\theta^0}(u) > 0.$$

So we get the first assertion of the theorem.

ii) Let us assume that as $n \to +\infty$, $\hat{\theta}_n \to \mu$ a.s. Let us then divide the numerator and the denominator of $\hat{\theta}_n$ given by Lemma 2.1 by $n^2$ and get

$$\hat{\theta}_n = \frac{\sum_{m=1}^R (X_{2m}/n) \times \sum_{m=1}^R \frac{X_m/n}{1+\hat{\theta}_n c_m}}{\sum_{m=1}^R (X_{1m}/n) \times \sum_{m=1}^R \frac{c_m X_m/n}{1+\hat{\theta}_n c_m}}. \tag{10}$$

By Lemma 2.3,

$$\frac{\sum_{m=1}^R X_{2m}/n}{\sum_{m=1}^R X_{1m}/n} \xrightarrow[n\to+\infty]{a.s.} \frac{\sum_{m=1}^R \pi_{2m}^0}{\sum_{m=1}^R \pi_{1m}^0} = \theta^0\beta^0$$

and

$$\frac{X_r/n}{1+\hat{\theta}_n c_r} = \frac{X_{1r}/n + X_{2r}/n}{1+\hat{\theta}_n c_r} \xrightarrow[n\to+\infty]{a.s.} \frac{\pi_{1r}^0 + \pi_{2r}^0}{1+\mu c_r} = \frac{(1+\theta^0 c_r)\phi_r^0}{(1+\theta^0 \sum_{m=1}^R c_m \phi_m^0)(1+\mu c_r)}.$$

Thus, as $n \to +\infty$, the first and the second hands of equation (10) yield, almost surely,

$$\mu = \theta^0\beta^0 \times \left( \sum_{r=1}^R \frac{(1+\theta^0 c_r)\phi_r^0}{(1+\mu c_r)} \right) \Big/ \left( \sum_{r=1}^R \frac{c_r(1+\theta^0 c_r)\phi_r^0}{(1+\mu c_r)} \right).$$

That is,

$$\mu \sum_{r=1}^R \frac{c_r(1+\theta^0 c_r)\phi_r^0}{(1+\mu c_r)} = \theta^0\beta^0 \times \sum_{r=1}^R \frac{(1+\theta^0 c_r)\phi_r^0}{(1+\mu c_r)},$$

which means that $F_{\theta^0}(\mu) = 0$ and hence $\mu = \theta^0$ by i). This completes the proof of Theorem 3.3. □

**Theorem 3.4.** *The MLE $\hat{\Phi}_n = (\hat{\theta}_n, \hat{\phi}_n)^\mathrm{T}$ converges a.s. to $\Phi^0 = (\theta^0, \phi^0)^\mathrm{T}$ with $\phi^0 = (\phi_1^0, \ldots, \phi_R^0)^\mathrm{T}$.*

**Proof.** This is a consequence of Theorem 3.1 and Theorem 3.3. Indeed, $\hat{\theta}_n$ converges a.s. to $\theta^0$ and since by Theorem 3.1, the consistency of $\hat{\theta}_n$ is equivalent to that of $\hat{\phi}_n$, then $\hat{\phi}_n$ converges also almost surely to $\phi^0$. Thus the vector $\hat{\Phi}_n = (\hat{\theta}_n, \hat{\phi}_n)^\mathrm{T}$ converges a.s. to the vector $\Phi^0 = (\theta^0, \phi^0)^\mathrm{T}$. □

## 4. Concluding remarks

We study the asymptotic strong consistency of a constrained maximum likelihood estimator of a vector parameter when a road safety measure has been applied to a target site. We intend to generalize our results to the multidimensional estimator proposed in [9] when we deal with the estimation of the effect of a road-safety measure applied on different target sites. Each target site counts $R$ ($R > 1$) mutually exclusive accidents types and is linked to a specific control area where the measure is not directly applied.

## References

[1] E. Barvínek, I. Daler, J. Francu, Convergence of sequences of inverse functions, Arch. Math. 27 (3–4) (1991) 201–204.
[2] D. Chafai, D. Concordet, On the strong consistency of asymptotic $M$-estimators, J. Stat. Plan. Inference 137 (2007) 2774–2783.
[3] K.L. Chung, A Course in Probability Theory, Academic Press, 2001.
[4] T.S. Ferguson, An inconsistent maximum likelihood estimate, J. Am. Stat. Assoc. 77 (380) (1982) 831–834.
[5] S. Fiorin, The strong consistency for maximum likelihood estimates: a proof not based on the likelihood ratio, C. R. Acad. Sci. Paris, Ser. I 331 (2000) 721–726.
[6] S. Kourouklis, On the strong consistency of a solution to the likelihood equation, Stat. Probab. Lett. 5 (1987) 23–27.
[7] J.F. Monahan, Numerical Methods of Statistics, 2nd edition, Cambridge University Press, 2011.
[8] A. N'Guessan, Analytical existence of solutions to a system of non-linear equations with application, J. Comput. Appl. Math. 234 (2010) 297–304.
[9] A. N'Guessan, A. Essai, C. Langrand, Estimation multidimensionnelle des contrôles et de l'effet moyen d'une mesure de sécurité routière, Rev. Stat. Appl. 49 (2) (2001) 85–102.
[10] A. N'Guessan, I.C. Geraldo, A cyclic algorithm for maximum likelihood estimation using Schur complement, Numer. Linear Algebra Appl. (2015), http://dx.doi.org/10.1002/nla.1999.
[11] A. N'Guessan, M. Truffier, Impact d'un aménagement de sécurité routière sur la gravité des accidents de la route, J. Soc. Fr. Stat. 149 (3) (2008) 23–41.
[12] B. Seo, B.G. Lindsay, Nearly universal consistency of maximum likelihood in discrete models, Stat. Probab. Lett. 83 (2013) 1699–1702.
[13] R.S. Strichartz, The Way of Analysis, Jones and Bartlett Books in Mathematics, Jones and Bartlett Publishers, 2000.
[14] A.W. Van der Vaart, Asymptotic Statistics, Cambridge University Press, 1998.
[15] A. Wald, Note on the consistency of the maximum likelihood estimate, Ann. Math. Stat. 20 (4) (1949) 595–601.

# Appendix C

# Some R codes

## C.1 Case $s = 1$

### C.1.1 The cyclic algorithm

```
#-------------------------------------------------------------#
# CA Algorithm R code                                         #
# Last update: 30/03/2015                                     #
# Authors: Assi N'GUESSAN, Issa Cherif GERALDO                #
# Correspondence: assi.nguessan@polytech-lille.fr             #
#-------------------------------------------------------------#
# vparam --> vector of initial solutions
# The following elements must be declared before calling the function:
# x1,x2 --> before/after accident data
# c --> control coefficients

CA_algorithm = function(vparam,eps=1e-8,maxiter=100)
{

R=length(vparam)-1 # Number of types of accidents
n=sum(x1,x2) # The total number of accidents

# Initialisation
theta=vparam[1]
phi=vparam[2:(R+1)]
loglik.new=loglik(vparam)

# Number of iterations
iter=0
# Stopping criteria
stop=0

# The loop
while (iter<maxiter && stop==0) #CA
{
# Updating theta for phi
theta=sum(x2)/(sum(c*phi)*sum(x1))

# Updating phi from theta
psr=1-sum((theta*c*(x1+x2))/(1+theta*c))/n
phi=(x1+x2)/(psr*n*(1+theta*c))
# Evaluation of the stopping criteria
loglik.old=loglik.new
loglik.new=loglik(c(theta,phi))
diff_l=abs(loglik.new-loglik.old)
if(diff_l < eps)
{ stop=1 } else { stop=0 }

# Updating the number of iterations
iter=iter+1

}

# Result
list(param=c(theta,phi), nb_iter=iter, loglik.value=loglik.new)
}
```

### C.1.2 MM algorithm

```
MM_algorithm1 = function(param,data,w=1,maxiter=100,eps=1e-8)
{

# Data
y=matrix(data$y,1,length(data$y))
z=matrix(data$z,1,length(data$z))

# Number of types of accidents
R=ncol(y)/2
n=sum(y)
n2=sum(y[(R+1):(2*R)])

# Initialisation
alpha_0=param[1]
v_alpha=param[2:(R+1)]

# Number of iterations
iter=0

# Stopping criterion
stop=0

while(iter<maxiter && stop==0)
{
old_alpha_0=alpha_0
old_v_alpha=v_alpha

# Update of alpha_0
a=1/(1+alpha_0*sum(z*v_alpha))
num1=n2+w*alpha_0
denom1=w+(a*n*sum(z*v_alpha))
alpha_0=num1/denom1

# Update of v_alpha
v_num2=y[1:R]+y[(R+1):(2*R)]+w*old_v_alpha
v_denom2=w+n+a*n*alpha_0*(z-sum(z*old_v_alpha))
v_alpha=v_num2/v_denom2

# Update of the number of iterations
iter=iter+1

# Update of the stopping criterion
old_alpha=c(old_alpha_0,old_v_alpha)
alpha=c(alpha_0,v_alpha)
dL=abs(loglik(alpha)-loglik(old_alpha))
stop=1*(dL<eps) #MM
}

res=list(param=alpha,nbiter=iter)
return(res)

}
```

### C.1.3 Functions for NR algorithm

```
#-------------------------------------------------------------#
# Implementation of the Newton algorithm                      #
#-------------------------------------------------------------#

#-------------------------------------------------------------#
# Re-parametrization and its inverse                          #
#-------------------------------------------------------------#

reparam2 = function(beta)
{
# Extraction des parametres
R=length(beta)-1

theta=beta[1]
phi=beta[2:(R+1)]

alpha=log(theta)
eta=log(phi)

return(c(alpha,eta))
}


reparam2.inv = function(beta)
{
# Extraction des parametres
R=length(beta)-1

alpha=beta[1]
eta=beta[2:(R+1)]
```

```
theta=exp(alpha)
phi=exp(eta)

return(c(theta,phi))
}


#-------------------------------------------------------------#
#  Log-likelihood under re-parametrization 2                 #
#-------------------------------------------------------------#

loglik2 = function(beta2)
{
return(loglik(reparam2.inv(beta2)))
}

#-------------------------------------------------------------#
# Function F such that the non-linear system is F(Theta)=0   #
#-------------------------------------------------------------#

F = function(theta)
{
np=length(theta)  # Number of parameters
nabla=1:np

# Parameters
a0=theta[1]
pi=theta[2:np]

K=sum(c*pi)

nabla[1]=n2-a0*n1*K
nabla[2:np]=(x1+x2)*(1+a0*K)-n*pi*(1+c*a0)

return(nabla)
}


F2 = function(beta2)
{
return(F(reparam2.inv(beta2)))
}


#-------------------------------------------------------------#
# Jacobian matrix of F = Hessian of the log-likelihood       #
#-------------------------------------------------------------#

F.jac = function(theta)
{
# Number of parametres
np=length(theta)
```

```
# Parameters
a0=theta[1]
phi=theta[2:np]

# Initialisation of J
J=matrix(0,np,np)

K=sum(c*phi)

# First line of J
J[1,1]=-n1*K
J[1,2:np]=-a0*n1*c

# First column of J
J[2:np,1]=(x1+x2)*K-n*phi*c

# The remaining lines and columns
J[2:np,2:np]=a0*(x1+x2)%*%t(c)-n*diag(1+a0*c)

return(J)
}


F2.jac = function(beta2)
{
D=diag(exp(beta2))
J=F.jac(reparam2.inv(beta2))%*%D
return(J)
}


#-------------------------------------------------------------#
#   Gauss algorithm for linear systems                        #
#-------------------------------------------------------------#

gauss = function(A,b,print=0)
{
n=length(b)

# Initialisation
x=1:n

# Triangularisation
A1=A
b1=b
for(k in 1:(n-1))
{
A0=A1
b0=b1
for(i in (k+1):n)
{
A1[i,]=A0[i,]-(A0[i,k]/A0[k,k])*A0[k,]
b1[i]=b0[i]-(A0[i,k]/A0[k,k])*b0[k]
}

# A message is printed if print equals 1
if(print==1)
{
cat("Triangularisation k =",k)
print(list(A1=A1,b1=b1))
}

}

# Resolution
x[n]=b1[n]/A1[n,n]
for(i in (n-1):1)
{
x[i]=(b1[i]-t(A1[i,(i+1):n])%*%x[(i+1):n])/A1[i,i]
}

return(x)

}


#------------------------------------------------------------#
# Newton method to solve F(x)=0 (x, F(x) in R^n)            #
#------------------------------------------------------------#

# Modification of the stopping criteria of the Newton's method

newton2 = function(x0,fun,jac,maxiter=100,eps=10^{-5},print=0)
{
# Initialisation
x1=x0

# First iteration
matJ=jac(x1); f=-fun(x1)
d=gauss(matJ,f)
x2=x1+d
nbiter=1

if(print){print(x2)}

while(nbiter<maxiter && abs(loglik2(x2)-loglik2(x1))>=eps)
{
x1=x2
matJ=jac(x1); f=-fun(x1)
d=gauss(matJ,f)
x2=x1+d
nbiter=nbiter+1

if(print){print(x2)}
}
```

```
if(abs(loglik2(x2)-loglik2(x1))<eps)
{
convergence=1
msg="The method has converged."
}
else
{
convergence=0
msg="The maximum number of iterations is reached."
}

if(print!=0) cat(msg," \n",sep="")
res=list(sol=x2,nbiter=nbiter,convergence=convergence,msg=msg)

return(res)
}


#-------------------------------------------------------------#
# Norm of a vector                                            #
#-------------------------------------------------------------#

norme = function(x)
{
return(sqrt(sum(x^2)))
}
```

### C.1.4 Code for running simulations

```
#--------------------------------------------------------------#
# Comparison of the CA algorithm with                          #
# (1) Newton-Raphson                                           #
# (2) MM algorithm (Mkhadri et al., 2010)                      #
# (3) BFGS (quasi-Newton)                                      #
# (4) Nelder-Mead                                              #
#                                           #
# Last update: 19/08/2015                                      #
# Authors: Assi N'GUESSAN, Issa Cherif GERALDO                 #
# Correspondence: assi.nguessan@polytech-lille.fr              #
#--------------------------------------------------------------#

#--------------------------------------------------------------#
# Loading external files                                       #
#--------------------------------------------------------------#

rm(list=ls())

library(alabama)
library(MCMCpack)

source("CA_algorithm.R")
source("functions_NR.R")
source("MM_algorithm.R")


#--------------------------------------------------------------#
# Generation of the true parameters                            #
#--------------------------------------------------------------#

# Number of simulated datasets
nb_sim=1e3
EPS=1e-6
MAXITER=1000

# Number of accidents types
R=5

# String vector for the names of the different methods
meth_names=c("CA","NR","MM","BFGS","NM")
nb_meth=length(meth_names)

# True parameters
theta0=0.5;
if(R==3){Phi0=c(0.019,0.513,0.468)}
if(R==5){Phi0=c(0.142,0.003,0.222,0.238,0.395)}
param0=c(theta0,Phi0)




#---------------------------------------------------#
# Functions needed to use the alabama package      #
#---------------------------------------------------#

loglik = function(phi)
{
L=sum((x1+x2)*log(phi[2:(R+1)])) + log(phi[1])*n2 - (log(1 + phi[1]*sum(c*phi[2:(R+1)])))*sum(x1+x2)
return(L)
}


negloglik = function(phi)
{
return(-loglik(phi))
}


loglik_grad = function(Alpha)
{
# Parameters extraction
R=length(Alpha)-1
a0=Alpha[1]
Pi=Alpha[2:(R+1)]

# Parameters adjustement
if(a0==0){a0=0.00001}
Pi[Pi==0]=0.00001

nabla=rep(0,(R+1))
nabla[1]=(1/a0)*(n2-a0*n1*sum(c*Pi))/(1+a0*sum(c*Pi))
nabla[2:(R+1)]=(x1+x2)/Pi - (n*a0*c)/(1+a0*sum(c*Pi))

return(nabla)
}


negloglik_grad = function(phi)
{
return(-loglik_grad(phi))
}


heq = function(phi)
{
R=length(phi)-1
return(sum(phi[2:length(phi)])-1)
}


heq.jac = function(phi)
{
R=length(phi)-1
return(matrix(c(0,rep(1,R)),1,(R+1)))
}


hin = function(phi)
{
phi
}

hin.jac = function(phi)
{
return(diag(rep(1,length(phi))))
}


#-------------------------------------------------------------#
# simulation                                                  #
#-------------------------------------------------------------#

# n --> Total number of accidents
# init_phi --> Type of initialisation scheme
values_n=c(50,5000)
values_init_phi=c(1:4)

#--------------------------------------------------
```

```
for(n in values_n)
{

for(init_phi in values_init_phi)
{


#-------------------- Results --------------------#
vtimes=matrix(0,nb_sim,nb_meth)
nb_iter=matrix(0,nb_sim,nb_meth)
M_loglik=matrix(0,nb_sim,nb_meth)
err_quadr=matrix(0,nb_sim,nb_meth)
true_loglik=rep(0,nb_sim)

colnames(vtimes)=meth_names
colnames(nb_iter)=meth_names
colnames(M_loglik)=meth_names
colnames(err_quadr)=meth_names

param_CA=matrix(0,nb_sim,(R+1))
param_NR=matrix(0,nb_sim,(R+1))
param_MM=matrix(0,nb_sim,(R+1))
param_BFGS=matrix(0,nb_sim,(R+1))
param_NM=matrix(0,nb_sim,(R+1))


for(i in 1:nb_sim)
{
# Printing
cat("R_",R,"_n_",n,"_init_phi_",init_phi,": ",i,"/",nb_sim," \n",sep="")

#--- Data generation --------------------------

c=runif(R,0.5,2.5)
p1=Phi0/(1+theta0*sum(c*Phi0))
p2=theta0*c*Phi0/(1+theta0*sum(c*Phi0))
p=c(p1,p2)
x=rmultinom(1,size=n,prob=p)
while(any(x==0)){x=rmultinom(1,size=n,prob=p)}
x1=x[1:R]
x2=x[(R+1):(2*R)]
n1=sum(x1)
n2=sum(x2)

#-----------------------------------------------------
true_loglik[i]=loglik(param0)
#-----------------------------------------------------
# Initial parameters
if(init_phi==1){phi_0=rep(1/R,R)}
if(init_phi==2){phi_0=(x1+x2)/n}
if(init_phi==3){u=runif(R,min=0.05,max=0.95);phi_0=u/sum(u)}
if(init_phi==4){phi_0=x1/n1}
```

```
theta_0=runif(1,0.1,5)
initpar=c(theta_0,phi_0)

#------------------------ CA Algorithm ----------------------#
t1=Sys.time()
res1=CA_algorithm(vparam=initpar,eps=EPS,maxiter=MAXITER)
t2=Sys.time()
param_CA[i,]=res1$param
nb_iter[i,1]=res1$nb_iter
M_loglik[i,1]=loglik(res1$param)
vtimes[i,1]=difftime(t2,t1,units="sec")
err_quadr[i,1]=mean((param_CA[i,]-param0)^2)


#----------------------- Newton-Raphson----------------------#
t1=Sys.time()
res2=newton2(x0=reparam2(initpar),fun=F2,jac=F2.jac,maxiter=MAXITER,eps=EPS)
t2=Sys.time()
res2.sol=reparam2.inv(res2$sol)
param_NR[i,]=res2.sol
nb_iter[i,2]=(res2$convergence)*(res2$nbiter)
M_loglik[i,2]=loglik(res2.sol)
vtimes[i,2]=difftime(t2,t1,units="sec")
err_quadr[i,2]=mean((param_NR[i,]-param0)^2)

#------------------------ MM algorithm ----------------------#
data=list(y=t(x),z=t(c))
t1=Sys.time()
res3=MM_algorithm1(initpar,data,w=1,eps=EPS,maxiter=MAXITER)
t2=Sys.time()
param_MM[i,]=res3$param
nb_iter[i,3]=res3$nbiter
M_loglik[i,3]=loglik(res3$param)
vtimes[i,3]=difftime(t2,t1,units="sec")
err_quadr[i,3]=mean((param_MM[i,]-param0)^2)


#------------------------ BFGS and NM ------------------------#

outer.ctrl=list(trace=FALSE,itmax=MAXITER,eps=EPS)
optim.ctrl=list(maxit=1e7)

outer.ctrl$method="BFGS"
t1=Sys.time()
res4=constrOptim.nl(par=initpar, fn=negloglik, gr=negloglik_grad,
  heq=heq, heq.jac=heq.jac,
       hin.jac=hin.jac, hin=hin,
    control.outer=outer.ctrl,control.optim=optim.ctrl)
t2=Sys.time()
param_BFGS[i,]=res4$par
nb_iter[i,4]=(res4$convergence==0)*(res4$outer.iterations)
M_loglik[i,4]=-res4$value
vtimes[i,4]=difftime(t2,t1,units="sec")
```

```
err_quadr[i,4]=mean((param_BFGS[i,]-param0)^2)

outer.ctrl$method="Nelder-Mead"
t1=Sys.time()
res5=constrOptim.nl(par=initpar, fn=negloglik, gr=negloglik_grad,
  heq=heq, heq.jac=heq.jac,
       hin.jac=hin.jac, hin=hin,
    control.outer=outer.ctrl,control.optim=optim.ctrl)
t2=Sys.time()
param_NM[i,]=res5$par
#nb_iter[i,5]=res5$outer.iterations
nb_iter[i,5]=(res5$convergence==0)*(res5$outer.iterations)
M_loglik[i,5]=-res5$value
vtimes[i,5]=difftime(t2,t1,units="sec")
err_quadr[i,5]=mean((param_NM[i,]-param0)^2)

}

#--------------------------------------------------------------#
#                           Stats                              #
#--------------------------------------------------------------#

# Number of iterations : min, max et average
nb_min_iter=apply(nb_iter,2,min)
nb_max_iter=apply(nb_iter,2,max)
nb_mean_iter=apply(nb_iter,2,mean)

# CPU time
mean_times=apply(vtimes,2,mean)
mean_times[1]=max(mean_times[1],1e-323)
time_ratio=mean_times/mean_times[1]

# Estimated parameters
mean_par = t(rbind(colMeans(param_CA),
      colMeans(param_NR),
    colMeans(param_MM),
    colMeans(param_BFGS),
                colMeans(param_NM)))


# Loglik: mean, min and max
mean_loglik=apply(M_loglik,2,mean)
min_loglik=apply(M_loglik,2,min)
max_loglik=apply(M_loglik,2,max)

# MSE
MSE=apply(err_quadr,2,mean)


# Results tables

stats_res_2=rbind(round(t(cbind(t(mean_par),nb_min_iter,nb_max_iter,
nb_mean_iter,mean_times,time_ratio,min_loglik,
max_loglik,mean_loglik)),7),round(MSE,7))
true_values=round(c(param0,rep(0,5),c(min(true_loglik),max(true_loglik),
mean(true_loglik)),0),3)
stats_res=cbind(true_values,stats_res_2)
colnames(stats_res)=c("TRUE",meth_names)
rownames(stats_res)=c("theta",paste("phi_",1:R,sep=""),
"Min iter.","Max iter.","Mean iter.","CPU time",
"CPU time ratio","Min loglik.","Max loglik.",
"Mean loglik.","MSE")

print(stats_res)


}

}
```

## C.2 Case $s > 1$

### C.2.1 The generalized cyclic algorithm

```
#-------------------------------------------------------#
#  Newton-Raphson's algorithm for updating theta        #
#-------------------------------------------------------#


NR_for_theta = function(theta,P,X,c.matrix,eps=10^(-5))
{
# Initialization
theta1=theta
S=nrow(c.matrix)
R=ncol(c.matrix)
X1=X[,1:R]
X2=X[,(R+1):(2*R)]

f1=f.theta(theta1,P,X,S,R,c.matrix)
fp1=fprime.theta(theta1,P,X,S,R,c.matrix)
theta2=theta1 - f1/fp1
f2=f.theta(theta2,P,X,S,R,c.matrix)

# Iterations
while(abs(f2-f1)>=eps)
{
theta1=theta2
f1=f.theta(theta1,P,X,S,R,c.matrix)
fp1=fprime.theta(theta1,P,X,S,R,c.matrix)
theta2=theta1 - f1/fp1
f2=f.theta(theta2,P,X,S,R,c.matrix)
}
```

```
return(theta2)

}


f.theta = function(theta,P,X,S,R,c.matrix)
{
# Initialization
X1=X[,1:R]
X2=X[,(R+1):(2*R)]

Prs=matrix(P,R,S)
Ns=rowSums(X)
Zs=rowSums(t(Prs)*c.matrix)

return(sum(Ns/(1+theta*Zs)) - sum(X1))

}


fprime.theta = function(theta,P,X,S,R,c.matrix)
{
Prs=matrix(P,R,S)
Zrs=t(c.matrix)
Ns=rowSums(X)
Zs=colSums(Prs*Zrs)

return(-sum(Ns*Zs/((1+theta*Zs)^2)))
}

#----------------------------------------------------------------------#
#                         Generalized CA                               #
#----------------------------------------------------------------------#

#------- Inputs
# Theta --> starting solution
# S --> Number of sites
# X --> matrix of dimension S*(2R)
# c.matrix --> matrix of dimension S*(2R)

CA_algorithm_multi_site = function(Theta,S,X,c.matrix,maxiter=100,eps=10^(-5),returntabpar=TRUE)
{
# Number of accident types
R=(length(Theta)-1)/S


# Number of sites
n_s=rowSums(X)

X1=X[,1:R]
X2=X[,(R+1):(2*R)]
Xrs=t(X1+X2)
```

```
# Initialization
P.estim=Theta[2:length(Theta)]

# Automatic computation of the first value of theta
theta.estim=NR_for_theta(0,P.estim,X,c.matrix)
Theta2=c(theta.estim,P.estim)

# Iterations
iter=0;
# Stopping criterion
stop=0;

# The loop
while (stop==0)
{

Theta1=Theta2

# Updating P
for(s in 1:S)
{
idx=((s-1)*R+1):(s*R)
K=theta.estim*sum((c.matrix[s,]*(Xrs[,s]))/(1+theta.estim*c.matrix[s,]))
KK=1/(1-(K/n_s[s]))
P.estim[idx]=KK*(Xrs[,s])/(n_s[s]*(1+theta.estim*c.matrix[s,]))
}

# Updating theta
theta.estim=NR_for_theta(0,P.estim,X,c.matrix)

iter=iter+1

# Evaluation of the stopping criterion
Theta2=c(theta.estim,P.estim)
LL1 = loglik(Theta1)
LL2 = loglik(Theta2)

if (iter>=maxiter || abs(LL2-LL1)<eps)
{ stop=1 } else { stop=0 }

}

# Results
conver=TRUE
if(iter>=maxiter){conver=FALSE}
return(list(Theta.estim=c(theta.estim,P.estim),
      nbiter=iter,convergence=conver))
}
```

### C.2.2 MM algorithm

```
MM_algorithm = function(param,data,w=1,maxiter=100,eps=1e-5)
{
```

```
# param$effet -> mean effect
# param$proba -> matrix S x R
# data$y -> accident data per site, one line per site
# data$z -> control coefficients, one line per site

# Number of sites and accidents types
if(is.matrix(data$y))
{
S=nrow(data$y); R=ncol(data$y)/2
} else {
S=1; R=length(data$y)
}

Y1sr=matrix(data$y[,1:R],S,R)
Y2sr=matrix(data$y[,(R+1):(2*R)],S,R)
Y=cbind(Y1sr,Y2sr)
Ysr=Y1sr+Y2sr
Zsr=matrix(data$z,S,R)


# Initialization
alpha_0=param$effet # mean effect
alpha=t(matrix(param$proba,R,S)) # matrix S x R

# Number of iterations
iter=0

# Stopping criterion
stop=0

while(iter<maxiter && stop==0)
{
old_alpha_0=alpha_0
old_alpha=alpha

# update of alpha_0
a=1/(1+old_alpha_0*diag(Zsr%*%t(old_alpha)))

num1=sum(Y2sr)+w*S*old_alpha_0
denom1=w*S+sum(a*rowSums(Ysr)*diag(Zsr%*%t(alpha)))

alpha_0=num1/denom1

# update of alpha
for(s in 1:S)
{
ys=sum(Ysr[s,])
for(r in 1:R)
{
num2=Ysr[s,r]+w*old_alpha[s,r]
denom2=w+ys+a[s]*ys*alpha_0*(Zsr[s,r]-sum(Zsr[s,]*old_alpha[s,]));
alpha[s,r]=num2/denom2
}
}

iter=iter+1

# loglik
old_theta=c(old_alpha_0,t(old_alpha))
theta=c(alpha_0,t(alpha))

stop=1*(abs(loglik(theta)-loglik(old_theta))<eps)

}

convergence=T
if(iter==maxiter){convergence=F}

res=list(alpha_0=alpha_0,alpha=alpha,nbiter=iter,convergence=convergence)
return(res)

}
```

### C.2.3 NR algorithm

```
systfun = function(param)
{
theta=param[1]
P=matrix(param[2:((S*R)+1)],S,R,byrow=T)

X1=X[,1:R]
X2=X[,(R+1):(2*R)]
Z=c.matrix
Cs=rowSums(Z*P)
Ns=rowSums(X)

F=rep(0,length(param))

F[1]=sum(Ns/(1+theta*Cs))-sum(X1)

for(s in 1:S)
{
idx=((s-1)*R+1):(s*R)

# I add 1 to idx since F[1] is for theta
F[idx+1]=(X1[s,]+X2[s,])*(1+theta*Cs[s])-Ns[s]*P[s,]*(1+theta*Z[s,])
}

return(F)

}


systfun.jac = function(param)
```

```
{
theta=param[1]
P=t(matrix(param[2:((S*R)+1)],R,S))

X1=X[,1:R]
X2=X[,(R+1):(2*R)]
Z=c.matrix
Cs=rowSums(Z*P)
Ns=rowSums(X)

J=matrix(0,length(param),length(param))

# Let g(theta,P)=0 the first equation
# dg/dtheta
J[1,1]=-sum((Ns*Cs)/((1+theta*Cs)^2))

for(s in 1:S)
{
# dg/dP : remainder of first line of J
idx=((s-1)*R+1):(s*R)
J[1,idx+1]=-Ns[s]*theta*Z[s,]/((1+theta*Cs[s])^2)

# remainder of first column
J[idx+1,1]=(X1[s,]+X2[s,])*Cs[s]-Ns[s]*P[s,]*Z[s,]

# remainder i.e. J[2:(R*S),2:(R*S)]
J[idx+1,idx+1]=theta*(X1[s,]+X2[s,])%*%t(Z[s,])-Ns[s]*diag(1+theta*Z[s,])
}

return(J)
}


reparam2 = function(beta)
{
return(log(beta))
}


reparam2.inv = function(beta)
{
return(exp(beta))
}


systfun2 = function(beta2)
{
return(systfun(reparam2.inv(beta2)))
}


systfun2.jac = function(beta2)
{
```

```
D=diag(exp(beta2))
J=systfun.jac(reparam2.inv(beta2))%*%D
return(J)
}
```

### C.2.4 Code for running the simulations

```
#--------------------------------------------------------------#
# Comparison of the GCA with:                                  #
# (1) Newton-Raphson                                           #
# (2) MM algorithm (Mkhadri et al., 2010)                      #
# (3) BFGS                                                     #
# (4) Nelder-Mead                                              #
#                         #
# Last update: 05/01/2016                                      #
#--------------------------------------------------------------#


#--------------------------------------------------------------#
# Loading external files and packages                          #
#--------------------------------------------------------------#

rm(list=ls())

library(alabama)
library(nleqslv)

REPERTOIRE="D:/Etudes/These/Codes/R/CA/multi-site"
setwd(REPERTOIRE)

source("CA_algorithm_multi_site.R")
source("MM.R")

#--------------------------------------------------------------#
#                        Functions                             #
#--------------------------------------------------------------#

loglik = function(Theta)
{
S=nrow(c.matrix)
R=ncol(c.matrix)

theta=Theta[1]
P=Theta[2:length(Theta)]

# Befaore/after accident data
X1=X[,1:R]
X2=X[,(R+1):(2*R)]

# P.matrix = Matrix R*S
P.matrix=matrix(P,R,S)
Xrs=t(X1+X2)
Xs=rowSums(X1+X2)
z=t(c.matrix)

L1=sum(Xrs*log(P.matrix))
L2=log(theta)*sum(X2)
L3=sum(Xs*log(1+theta*colSums(z*P.matrix)))

return(L1+L2-L3)
}


negloglik = function(Theta)
{
return(-loglik(Theta))
}


loglik.grad = function(Theta)
{
# Extraction
theta=Theta[1]
P=t(matrix(Theta[2:length(Theta)],R,S))
Z=c.matrix
Cs=rowSums(Z*P)
Ns=rowSums(X)
X1=X[,1:R]
X2=X[,(R+1):(2*R)]
N2=sum(X2)


# Correction of parameters
if(theta==0){theta=1e-5}

nabla=rep(0,length(Theta))

nabla[1]=(N2/theta)-sum(Ns*Cs/(1+theta*Cs))

for(s in 1:S)
{
idx=((s-1)*R+1):(s*R)
# I add 1 to idx since nabla[1] is for theta
nabla[idx+1]=((X1[s,]+X2[s,])/P[s,])-(Ns[s]*theta*Z[s,])/(1+theta*Cs[s])
}

return(nabla)
}


negloglik.grad = function(Theta)
{
return(-loglik.grad(Theta))
}
```

```
heq = function(Theta)
{
h=rep(0,S)
for(s in 1:S)
{
idx=((s-1)*R+1):(s*R)
h[s]=sum(Theta[idx+1])-1
}
return(h)
}


heq.jac = function(Theta)
{
Jh=matrix(0,S,length(Theta))
for(s in 1:S)
{
idx=((s-1)*R+1):(s*R)
Jh[s,idx+1]=1
}

return(Jh)
}


hin = function(Theta)
{
Theta
}


hin.jac = function(Theta)
{
return(diag(rep(1,length(Theta))))
}


#-------------------------------------------------------------#
# Generation of true parameters                               #
#-------------------------------------------------------------#

# Number of replications
nb_sim=5
EPS=1e-5
MAXITER=100

for(scen in c(1:4))
{

theta.vrai=0.8
```

```
#----------
if(scen==1)
{
S=5; R=3
      P.vrai=c(0.80,0.15,0.05,
               0.10,0.30,0.60,
               0.35,0.30,0.35,
               0.70,0.20,0.10,
   0.30,0.40,0.30)
}
if(scen==2)
{
S=5; R=10
      P.vrai=c(0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05)
}
if(scen==3)
{
S=10; R=10
      P.vrai=c(0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05)
}
if(scen==4)
{
S=20; R=10
      P.vrai=c(0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
   0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.05,0.05,0.10,0.25,0.05,0.05,0.15,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
               0.40,0.10,0.05,0.10,0.10,0.05,0.05,0.05,0.05,0.05)
}

#----------
Theta.vrai=c(theta.vrai,P.vrai)

#-------------------------------------------------------------#
# Simulation process                                          #
#-------------------------------------------------------------#

# Methods names
noms_methodes=c("CA","MM","NR","BFGS","NM")
nb_meth=length(noms_methodes)

# Results
temps=matrix(0,nb_sim,nb_meth)
nb_iter=matrix(0,nb_sim,nb_meth)
err_quadr=matrix(0,nb_sim,nb_meth)
converg=matrix(0,nb_sim,nb_meth)

colnames(temps)=colnames(nb_iter)=noms_methodes
colnames(converg)=colnames(err_quadr)=noms_methodes

param_CA=param_MM=param_NR=param_BFGS=param_NM=matrix(0,nb_sim,(S*R+1))

for(Nps in c(5000))
{
for(initialisation in c(3))
{
for(i in 1:nb_sim)
{

#----------------------------------------------------
# Printing
cat("S-",S,"-R-",R,"-I-",initialisation,"-Nps-",Nps," : ",i,"/",nb_sim," \n",sep="")

#----------------------------------------------------
# Accident data simulation

vectN=rep(Nps,S)

c.matrix=matrix(runif(S*R,0.5,2.5),S,R)

X=matrix(0,S,2*R)
for(s in 1:S)
{
idx=((s-1)*R+1):(s*R)
c_s=sum(c.matrix[s,]*P.vrai[idx])
q1=P.vrai[idx]/(1+theta.vrai*c_s)
q2=theta.vrai*c.matrix[s,]*P.vrai[idx]/(1+theta.vrai*c_s)
q=c(q1,q2)

X[s,]=rmultinom(1,size=vectN[s],prob=q)
while(any(X[s,]==0)){X[s,]=rmultinom(1,size=vectN[s],prob=q)}
}
X1=X[,1:R]
X2=X[,(R+1):(2*R)]
#-----------------------------------------------------

# Starting values
theta.init=runif(1,0.1,2.9)

if(initialisation==1){P.init=rep(rep(1/R,R),S)}
if(initialisation==2)
{
P.init=c(t((X1+X2)/apply(X,1,sum)))
}
if(initialisation==3)
{
U=matrix(runif(R*S,min=0.05,max=0.95),S,R)
P.init=c(t(U/rowSums(U)))
}
if(initialisation==4)
{
P.init=c(t(X1/apply(X1,1,sum)))
}


Theta.init=c(theta.init,P.init)

#-----------------------------------------------------
#----------------------- CA Algorithm -----------------------#

t1=Sys.time()
res1=CA_algorithm_multi_site(Theta.init,S,X,c.matrix)
t2=Sys.time()
param_CA[i,]=res1$Theta.estim
nb_iter[i,1]=res1$nbiter
converg[i,1]=1*(res1$convergence==TRUE)
temps[i,1]=difftime(t2,t1,units="sec")
err_quadr[i,1]=mean((param_CA[i,]-Theta.vrai)^2)

#----------------------- MM Algorithm -----------------------#

data=list(y=X,z=c.matrix)
Theta.init2=list(effet=theta.init,proba=P.init)
t1=Sys.time()
res2=MM_algorithm(Theta.init2,data,w=1,maxiter=100,eps=1e-5)
t2=Sys.time()
param_MM[i,]=c(res2$alpha_0,t(res2$alpha))
nb_iter[i,2]=res2$nbiter
converg[i,2]=1*(res2$convergence==TRUE)
temps[i,2]=difftime(t2,t1,units="sec")
```

```
err_quadr[i,2]=mean((param_MM[i,]-Theta.vrai)^2)

#----------------------- Newton-Raphson ----------------------#

t1=Sys.time()
res3=try(nleqslv(x=reparam2(Theta.init), fn=systfun2, jacobian=systfun2.jac,
 method = "Newton", global = "none", xscalm = "fixed",
 control = list(maxit=100)), silent=TRUE)

t2=Sys.time()
if(class(res3)!="try-error")
{
param_NR[i,]=reparam2.inv(res3$x)
nb_iter[i,3]=res3$iter
#converg[i,3]=1*(res3$termcd==1)*(sum(param_NR[i,]>0)==(1+S*R))
converg[i,3]=1*(res3$termcd==1)
temps[i,3]=difftime(t2,t1,units="sec")
err_quadr[i,3]=mean((param_NR[i,]-Theta.vrai)^2)
}


#------------------------- BFGS et NM ------------------------#

t1=Sys.time()
res4=constrOptim.nl(par=Theta.init, fn=negloglik, #gr=negloglik.grad,
 heq=heq, heq.jac=heq.jac, hin.jac=hin.jac, hin=hin,
 control.outer=list(method="BFGS",trace=FALSE,itmax=500,eps=EPS))
t2=Sys.time()
param_BFGS[i,]=res4$par
nb_iter[i,4]=res4$outer.iterations
converg[i,4]=1*(res4$convergence==0)
temps[i,4]=difftime(t2,t1,units="sec")
err_quadr[i,4]=mean((param_BFGS[i,]-Theta.vrai)^2)


# Nelder-Mead
t1=Sys.time()
res5=constrOptim.nl(par=Theta.init, fn=negloglik,
 heq=heq, heq.jac=heq.jac, hin.jac=hin.jac, hin=hin,
 control.outer=list(method="Nelder-Mead",trace=FALSE,itmax=500,eps=EPS))
t2=Sys.time()
param_NM[i,]=res5$par
nb_iter[i,5]=res5$outer.iterations
converg[i,5]=1  # pour NM, res5$convergence ne semble rien donner
temps[i,5]=difftime(t2,t1,units="sec")
err_quadr[i,5]=mean((param_NM[i,]-Theta.vrai)^2)

}


#------------------------------------------------------------#
#                            Stats                           #
#------------------------------------------------------------#
```

```
# Number of convergence
nb_conver=apply(converg,2,sum)

# index to be considered
idx_CA=which(converg[,1]==1)
idx_MM=which(converg[,2]==1)
idx_NR=which(converg[,3]==1)
idx_BFGS=which(converg[,4]==1)
idx_NM=which(converg[,5]==1)

# Number of iterations: min, max et mean
nb_min_iter=c(min(nb_iter[idx_CA,1]),min(nb_iter[idx_MM,2]),
  min(nb_iter[idx_NR,3]),min(nb_iter[idx_BFGS,4]),
        min(nb_iter[idx_NM,5]))
nb_max_iter=c(max(nb_iter[idx_CA,1]),max(nb_iter[idx_MM,2]),
  max(nb_iter[idx_NR,3]),max(nb_iter[idx_BFGS,4]),
        max(nb_iter[idx_NM,5]))
nb_moy_iter=c(mean(nb_iter[idx_CA,1]),mean(nb_iter[idx_MM,2]),
  mean(nb_iter[idx_NR,3]),mean(nb_iter[idx_BFGS,4]),
        mean(nb_iter[idx_NM,5]))

# CPU
temps_moy=c(mean(temps[idx_CA,1]),mean(temps[idx_MM,2]),
  mean(temps[idx_NR,3]),mean(temps[idx_BFGS,4]),
        mean(temps[idx_NM,5]))

ratio_temps=temps_moy/temps_moy[1]

# Parameters
par_moy=t(rbind(colMeans(param_CA[idx_CA,]),
      colMeans(param_MM[idx_MM,]),
    colMeans(param_NR[idx_NR,]),
                colMeans(param_BFGS[idx_BFGS,]),
    colMeans(param_NM[idx_NM,])))
rownames(par_moy)=c("theta",paste("phi",10*rep(1:S,each=R)+(1:R),sep="_"))

# MSE
err_quadr_moy=c(mean(err_quadr[idx_CA,1]),mean(err_quadr[idx_MM,2]),
  mean(err_quadr[idx_NR,3]),mean(err_quadr[idx_BFGS,4]),
        mean(err_quadr[idx_NM,5]))


# Stats

stats_res_2=rbind(nb_conver,nb_min_iter,nb_max_iter,nb_moy_iter,
temps_moy,ratio_temps,err_quadr_moy)
stats_res=rbind(par_moy,stats_res_2)

print(stats_res_2)


}
```

```
}


} # end for scen
```

# Appendix D

# Other publications of the author

**Journal articles (1)**

[**A1**] I. C. Geraldo, T. Bose, K. M. Pekpe, J.-P. Cassar, A.R. Mohanty, K. Paumel (2014), Acoustic monitoring of sodium boiling in a liquid metal fast breeder reactor from autoregressive models, *Nuclear Engineering and Design*, 278, pp. 573-585.

**Conferences (2)**

[**C1**] I. C. Geraldo, T. Bose, K. M. Pekpe, J.-P. Cassar, A.R. Mohanty, K. Paumel (2015). Autoregressive Model-Based Boiling Detection in a Liquid Metal Fast Breeder Reactor, *IFAC-PapersOnLine*, 48 (21), pp. 566-570, *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS*.

[**C2**] T. Bose, I. C. Geraldo, K. M. Pekpe, J.-P. Cassar, A.R. Mohanty, K. Paumel (2014). Boiling Detection in a LMFBR Using Autoregressive Models and SVM, *Proceedings of the 19th IFAC World Congress*, pp. 8885-8890.