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 $x = (x_1, \ldots, x_R)^T$ a vector of dimension R ($R > 2$) made of observed data and $x_i (i = 1, \ldots, R)$ 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 $\phi_j > 0$.

DOI: 10.1002/nla

- (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 Θ.
- (H_3) $\frac{\partial h}{\partial \theta} = 0$, $\frac{\partial h}{\partial \phi_j} \neq 0$ $(j = 1, \ldots, d-1)$.
- (H_4) There is a non-zero constant η such that $\langle \nabla_\phi h, \phi \rangle = \eta$ where ∇_ϕ 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 $0_{d-1} = (0, \ldots, 0) \in \mathbb{R}^{d-1}$.
- (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 Θ belongs. In particular, supposing that vector Θ'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 Θ 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 ϕ. Conditions (H_5) and (H_6) specify the estimation structuration by blocks of Θ, which consists in linearly obtaining ϕ, θ 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 λ 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

$$\left(\frac{\partial \ell}{\partial \theta} \right)_{\hat{\Theta}} = 0 \quad \text{et} \quad \left(\frac{\partial \ell}{\partial \phi_j} \right)_{\hat{\theta}} - \hat{\lambda} \left(\frac{\partial h}{\partial \phi_j} \right)_{\hat{\Theta}} = 0, \; (j = 1, \ldots, d-1).$$

DOI: 10.1002/nla

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 Θ 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 $\sum_{j=1}^{r} \phi_j = 1$.

DOI: 10.1002/nla

(H_8) There is a vector $Z = (w_1, \ldots, w_r)^T$, $w_j > 0$ such that

$$D_{\theta,x} = \Lambda_{\theta,Z} - \theta B_x Z^T$$

where $\Lambda_{\theta,Z} = \mathrm{Diag}(1 + \theta w_1, \ldots, 1 + \theta w_r)$ is a diagonal matrix $r \times r$,

$$B_x = \left(\frac{x_1 + x_{1+r}}{n}, \ldots, \frac{x_r + x_{2r}}{n} \right)^T \in \mathbb{R}^r$$

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 $\Lambda_{\theta,Z}^{-1}$ exists and $0 < \hat{\theta} Z^T \Lambda_{\hat{\theta},Z}^{-1} B_x < 1$, we show that $D_{\hat{\theta},x}^{-1}$ exists and that

$$D_{\hat{\theta},x}^{-1} = \Lambda_{\hat{\theta},Z}^{-1} + \hat{\theta} \left(1 - \hat{\theta} \langle Z, B_x \rangle_{\Lambda_{\hat{\theta},Z}^{-1}} \right)^{-1} \Lambda_{\hat{\theta},Z}^{-1} B_x Z^T \Lambda_{\hat{\theta},Z}^{-1}.$$

After some matrix manipulations, we show that

$$\hat{\phi} = D_{\hat{\theta},x} B_x = \frac{1}{1 - \hat{\theta} \langle Z, B_x \rangle_{\Lambda_{\hat{\theta},Z}^{-1}}} \Lambda_{\hat{\theta},Z}^{-1} B_x$$

where

$$1 - \hat{\theta} \langle Z, B_x \rangle_{\Lambda_{\theta,Z}^{-1}} = 1 - \frac{1}{n} \sum_{m=1}^{r} \frac{\hat{\theta} w_m x_{\cdot m}}{1 + \hat{\theta} w_m}.$$

We then deduce the expression of $\hat{\phi}_j$ ($j = 1, \ldots, r$) 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 $\hat{\phi}_j^{(1)} = (x_j + x_{j+r})/n$, ($j = 1, \ldots, r$) 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 $\hat{\phi}^{(0)} = D_{\hat{\theta}^{(0)},x}^{-1} B_x$ 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 $D_{\hat{\theta}^{(k)},x}^{-1}$. $\hat{\Theta}^{(k)}$ is updated. Maximization conditions of $\ell(\Theta, x)$ and assumptions are

DOI: 10.1002/nla

Algorithm 1 General scheme of the cyclic algorithm

Require: $x = (x_1, \ldots, x_R)^T$, $\epsilon_1 > 0$, $\epsilon_2 > 0$ two precisions, η.
Ensure: $\hat{\Theta}$ the MLE of Θ, k_0 the number of iterations.

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 $(\nabla_\phi h)_{\hat{\Theta}^{(0)}}$.
Set $k = 0$.
Set STOP $= 0$.
while STOP $\neq 1$ **do**
    Compute $\hat{\theta}^{(k+1)} = g_1^{-1}(g_2(\hat{\phi}^{(k)}), x)$ and $D^{-1}_{\hat{\theta}^{(k+1)},x}$.
    Compute $\hat{\phi}^{(k+1)} = D^{-1}_{\hat{\theta}^{(k+1)},x} B_x$.
    Set $\hat{\Theta}^{(k+1)} = \left(\hat{\theta}^{(k+1)}, (\hat{\phi}^{(k+1)})^T\right)^T$ and compute $\ell(\hat{\Theta}^{(k+1)}, x)$, $(\nabla_\phi h)_{\hat{\Theta}^{(k+1)}}$.
    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**
        STOP $= 0$.
    else
        STOP $= 1$.
        Set $\hat{\Theta} = \Theta^{(k+1)}$.
    end if
    $k = k + 1$.
end while
Set $k_0 = k$.
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

$$\hat{\Theta}^{(k+1)} = \hat{\Theta}^{(k)} - \left(\nabla^2_\Theta \ell\right)^{-1}_{\hat{\Theta}^{(k)}} \left(\nabla_\Theta \ell\right)_{\hat{\Theta}^{(k)}} \tag{7}$$

where $\left(\nabla^2_\Theta \ell\right)$ represents the Hessian matrix of ℓ 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

DOI: 10.1002/nla

Algorithm 2

Require: $x = (x_1, \ldots, x_{2r})^T, \epsilon_1 > 0, \epsilon_2 > 0, \eta = 1, Z = (w_1, \ldots, w_r)^T, 1_r = (1, \ldots, 1)^T \in \mathbb{R}^r$.

Ensure: $\hat{\Theta}$ the MLE of Θ, k_0 the number of iterations.

Compute $n = \sum_{j=1}^{r}(x_j + x_{j+r})$, $B_x = n^{-1}(x_1 + x_{1+r}, \ldots, x_r + x_{2r})^T$
Set $\hat{\theta}^{(0)} = 0$
For $j = 1, \ldots, r$, Compute $B_{x,j} = n^{-1}(x_j + x_{j+r})$
Set $\hat{\Theta}^{(0)} = \left(0, B_x^T\right)^T$
Set $k = 0$.
Set STOP $= 0$.
while STOP $\neq 1$ **do**
    Compute $\hat{\theta}^{(k+1)} = g_1^{-1}(g_2(\hat{\phi}^{(k)}), x)$
    For $m = 1, \ldots, r$, $\hat{\Delta}_{n,m}^{(k+1)} = \dfrac{\hat{\theta}^{(k+1)} w_m (x_m + x_{m+r})}{n(1 + \hat{\theta}^{(k+1)} w_m)}$ and $\hat{\Delta}_n^{(k+1)} = 1 - \sum_{m=1}^{r} \hat{\Delta}_{n,m}^{(k+1)}$.
    For $j = 1, \ldots, r$, compute $\hat{\phi}_j^{(k+1)} = \frac{1}{\hat{\Delta}_n^{(k+1)}} \times \frac{B_{x,j}}{1+\hat{\theta}^{(k+1)} w_j}$.
    Compute $\hat{\phi}^{(k+1)} = (\hat{\phi}_1^{(k+1)}, \ldots, \hat{\phi}_r^{(k+1)})^T$.
    Set $\hat{\Theta}^{(k+1)} = \left(\hat{\theta}^{(k+1)}, (\hat{\phi}^{(k+1)})^T\right)^T$.
    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**
        STOP $= 0$.
    else
        STOP $= 1$.
        Set $\hat{\Theta} = \Theta^{(k+1)}$.
    end if
    $k = k + 1$.
end while
Set $k_0 = k$.
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

$$g_{\hat{\Theta}^{(k)}}(\Theta) \leqslant \ell(\Theta), \quad \text{for all } \Theta,$$
$$g_{\hat{\Theta}^{(k)}}(\hat{\Theta}^{(k)}) = \ell(\hat{\Theta}^{(k)}).$$

DOI: 10.1002/nla

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 ℓ is replaced by another point. Iterations are repeated until the images of the vertices by ℓ 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}), $j = 1, \ldots, r$

DOI: 10.1002/nla

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 $C = (c_1, \ldots, c_r)^T$ 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

$$p_{1j}(\Theta) = \frac{\phi_j}{1 + \theta C^T \phi} \quad \text{and} \quad p_{2j}(\Theta) = \frac{\theta c_j \phi_j}{1 + \theta C^T \phi} \qquad \forall j = 1, \ldots, r \tag{11}$$

where $\theta > 0$, $0 < \phi_j < 1$ with $\sum_{j=1}^{r} \phi_j = 1$. The scalar θ represents the unknown average effect of the road safety measure, while each $\phi_j (j = 1, 2, \ldots, r)$ denotes the global accident risk of type j before and after the application of the road safety measure. A value of θ 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 $x_1 = (x_{11}, \ldots, x_{1r})$ and $x_2 = (x_{21}, \ldots, x_{2r})$ 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 Θ 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

$$\ell(\Theta, x) = \sum_{j=1}^{r} \left[x_{.j} \log(\phi_j) + x_{2j} \log(\theta) - x_{.j} \log \left(1 + \theta \sum_{m=1}^{r} c_m \phi_m \right) \right] \tag{14}$$

with $x_{.j} = x_{1j} + x_{2j}$. Different iterative methods [13, 22] may be used to solve the constrained maximum likelihood estimation problem of Θ. 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 θ and ϕ of model (11) justify assumption (H_1). Assumption (H_2) is naturally verified along with (H_3) with $\nabla_\phi h = (1, \ldots, 1)^T \in \mathbb{R}^r$. 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 θ and ϕ_j $(j = 1, \ldots, r)$, we show that

DOI: 10.1002/nla

$$\begin{aligned}\frac{\partial \ell}{\partial \theta} &= \frac{\sum_{j=1}^{r} x_{2j}}{\theta} - \frac{n c^T \phi}{1+\theta c^T \phi} \\ \frac{\partial \ell}{\partial \phi_j} &= \frac{x_{.j}}{\phi_j} - \frac{\theta c_j n}{1+\theta c^T \phi} - \lambda, \quad j = 1, \ldots, r.\end{aligned} \tag{15}$$

where λ 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:

$$\begin{aligned}\frac{\sum_{j=1}^{r} x_{2j}}{\hat{\theta}} - \frac{n c^T \hat{\phi}}{1+\hat{\theta} c^T \hat{\phi}} &= 0 \\ \frac{x_{.j}}{\hat{\phi}_j} - \frac{\hat{\theta} c_j n}{1+\hat{\theta} c^T \hat{\phi}} - \hat{\lambda} &= 0, \quad j = 1, \ldots, r. \\ 1_r^T \hat{\phi} &= 1\end{aligned} \tag{16}$$

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:

$$\begin{cases} \hat{\theta} = x_{2.}(x_{1.})^{-1}(c^T \hat{\phi})^{-1} \\ \hat{\phi}_j = \left(1 - \frac{1}{n} \sum_{m=1}^{r} \frac{\hat{\theta} c_m x_{.m}}{1+\hat{\theta} c_m}\right)^{-1} \times \frac{x_{.j}}{n(1+\hat{\theta} c_j)} \qquad (j = 1, \ldots, r), \end{cases} \tag{17}$$

where $x_{.m} = x_{1m} + x_{2m}$ et $x_{t.} = \sum_{m=1}^{r} x_{tm}, (t = 1, 2)$.

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

DOI: 10.1002/nla

Algorithm 3

Require: $x = (x_{11}, \ldots, x_{1r}, x_{21}, \ldots, x_{2r})^T$, $\epsilon_1 > 0$, $\epsilon_2 > 0$.
Ensure: $\hat{\Theta}$ the MLE of Θ, k_0 the number of iterations.

Compute n, $B_x = n^{-1}(x_{.1}, \ldots, x_{.r})^T$, $x_{1.}$, $x_{2.}$.
Set $\hat{\phi}^{(0)} = B_x$, $\hat{\Theta}^{(0)} = \left(0, B_x^T\right)^T$.
Set $k = 0$.
Set STOP $= 0$.
while STOP $\neq 1$ **do**
    Compute $\hat{\theta}^{(k+1)} = x_{2.}(x_{1.})^{-1}(c^T\hat{\phi}^{(k)})^{-1}$
    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 $m = 1, \ldots, r$.
    Compute $\hat{\Delta}_n^{(k+1)} = 1 - \sum_{m=1}^{r} \hat{\Delta}_{n,m}^{(k+1)}$.
    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 $j = 1, \ldots, r$.
    Set $\hat{\phi}^{(k+1)} = (\hat{\phi}_1^{(k+1)}, \ldots, \hat{\phi}_r^{(k+1)})^T$.
    Set $\hat{\Theta}^{(k+1)} = \left(\hat{\theta}^{(k+1)}, (\hat{\phi}^{(k+1)})^T\right)^T$.
    if $|1_r^T\hat{\phi}| > \epsilon_1$ or $|\ell(\hat{\Theta}^{(k+1)}, x) - \ell(\hat{\Theta}^{(k)}, x)| > \epsilon_2$ **then**
        STOP $= 0$
    else
        STOP $= 1$
        Set $\hat{\Theta} = \Theta^{(k+1)}$.
    end if
    $k = k + 1$.
end while
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)

$$\mathrm{MSE}(\hat{\Theta}) = \frac{1}{1+r}\left((\hat{\theta} - \theta^0)^2 + \sum_{j=1}^{r}\left(\hat{\phi}_j - \phi_j^0\right)^2\right) \tag{18}$$

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

DOI: 10.1002/nla

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 $C = (c_1, \ldots, c_r)^T$ from a uniform random variable $U[0.5; 2.5]$. We generate the true value of parameter θ, denoted θ^0, as the mean of a uniform random variable $U]0; 1[$ and the true value of vector ϕ, denoted $\phi^0 = \left(\phi_1^0, \ldots, \phi_r^0\right)^T$ from a Dirichlet distribution. Afterwards, we compute the true values

$$p_{1j}^0(\Theta) = \frac{\phi_j^0}{1 + \theta^0 \sum_{m=1}^r c_m \phi_m^0} \quad \text{and} \quad p_{2j}^0(\Theta) = \frac{\theta^0 c_j \phi_j^0}{1 + \theta^0 \sum_{m=1}^r c_m \phi_m^0} \qquad \forall j = 1, \ldots, r$$

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 $U = (u_1, \ldots, u_r)^T$ 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

DOI: 10.1002/nla

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
$n = 50$	Θ^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
$n = 5000$	Θ^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.

DOI: 10.1002/nla

Table III. Results for scenario $r = 3$ and $\hat{\phi}^{(0)} = (x_1 + x_2)/n$.

		CA	NR	MM	BFGS	NM
$n = 50$	Θ^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
$n = 5000$	Θ^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
$n = 50$	Θ^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
$n = 5000$	Θ^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).

DOI: 10.1002/nla

Table V. Results for $r = 3$ and $\hat{\phi}^{(0)} = x_1 / \sum_{m=1}^{r} x_{1m}$.

		CA	NR	MM	BFGS	NM
$n = 50$	Θ^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
$n = 5000$	Θ^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
$n = 50$	Θ^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
$n = 5000$	Θ^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.

DOI: 10.1002/nla

Table VII. Results for scenario $r = 5$ and $\hat{\phi}^{(0)} = (x_1 + x_2)/n$.

		CA	NR	MM	BFGS	NM
$n = 50$	Θ^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
$n = 5000$	Θ^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
$n = 50$	Θ^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
$n = 5000$	Θ^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.

DOI: 10.1002/nla

Table IX. Results for $r = 5$ and $\hat{\phi}^{(0)} = x_1 / \sum_{m=1}^{r} x_{1m}$.

		CA	NR	MM	BFGS	NM
$n = 50$	Θ^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
$n = 5000$	Θ^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

DOI: 10.1002/nla

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 Σ_θ exists as well as the Schur complement of 1 in Σ_θ. 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

$$\hat{\phi} = D^{-1}_{\hat{\theta},x} B_x = \left(1 - \hat{\theta} \langle Z, B_x \rangle_{\Lambda^{-1}_{\theta,Z}}\right)^{-1} \Lambda^{-1}_{\theta,Z} B_x.$$

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.

DOI: 10.1002/nla

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.

DOI: 10.1002/nla