NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS  
Numer. Linear Algebra Appl. 2015; 22:1161–1179  
Publié en ligne le 18 juin 2015 dans Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.1999

Un algorithme cyclique pour l'estimation du maximum de vraisemblance utilisant le complément de Schur

Assi N'Guessan1,\*,† et Issa Cherif Geraldo 1,2  
1Laboratoire Paul Painlevé (UMR CNRS 8524), Université de Lille 1, 59655 Villeneuve d'Ascq CEDEX, France  
2Département 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

RÉSUMÉ  
En utilisant le complément de Schur d'une matrice, nous proposons un cadre computationnel pour effectuer une estimation du maximum de vraisemblance sous contraintes dans lequel les paramètres inconnus peuvent être partitionnés en deux ensembles. Sous des conditions de régularité appropriées, les équations d'estimation correspondantes forment un système non linéaire d'équations avec contraintes. La résolution de ce système est typiquement accomplie via des méthodes qui nécessitent le calcul ou l'estimation d'une matrice Hessienne. Nous présentons un algorithme alternatif qui résout le système non linéaire contraint de manière descendante par coordonnées par blocs. Une forme explicite pour la solution est donnée. L'algorithme global se révèle, dans des études numériques, plus rapide que les méthodes standard qui calculent ou approximent la Hessienne, ainsi que l'algorithme classique de Nelder-Mead. Nous appliquons notre approche à un problème motivant d'évaluation de l'efficacité des Politiques de Sécurité Routière. Cela inclut plusieurs études numériques sur des données simulées. Copyright © 2015 John Wiley & Sons, Ltd.

Reçu le 30 novembre 2013 ; Révisé le 20 mai 2015 ; Accepté le 28 mai 2015

MOTS CLÉS : complément de Schur ; optimisation sous contraintes ; maximum de vraisemblance ; algorithme cyclique ; mesure de sécurité routière ; simulation

1. INTRODUCTION  
La méthode du maximum de vraisemblance [1-3], très souvent citée et utilisée en statistique, est une méthode d'optimisation numérique permettant, selon les données du problème, d'estimer les paramètres inconnus liés à une fonction de probabilité. L'approche consistant à maximiser cette fonction de probabilité est donc appelée méthode du maximum de vraisemblance avec ou sans contraintes. Cette fonction est souvent obtenue sous la forme d'un produit de probabilités sous contraintes. Il est donc équivalent de maximiser son logarithme en tenant compte des contraintes. On parle alors de la méthode du logarithme de la vraisemblance contrainte. L'une des fonctions de probabilité les plus populaires et utilisées est celle généralement appelée loi ou distribution multinomiale. Le principe de base de cette fonction multinomiale consiste à répartir un nombre fini d'éléments dans un nombre fini de catégories ou de classes. La probabilité qu'un objet appartienne à une classe est appelée probabilité de classe, la somme de toutes les probabilités de classe étant égale à 1. Étant donné une distribution, l'estimation de la probabilité de classe est obtenue par maximisation du produit des probabilités de classe sous une contrainte linéaire (somme des probabilités de classe égale à 1) et sous une contrainte de limite (chaque probabilité de classe est entre 0 et 1). Les probabilités de classe maximisant ce produit sont facilement obtenues grâce au rapport entre le nombre d'objets tombés dans une classe et la somme totale des éléments à distribuer. En pratique, malheureusement, chaque probabilité de classe dépend non seulement des données à distribuer mais aussi de paramètres auxiliaires inconnus, qui sont très souvent sous contraintes.

\*Correspondance à : Assi N'Guessan, Bât. Polytech'Lille, Université Lille 1, 59655 Villeneuve d'Ascq, France.  
†E-mail : [assi.nguessan@polytech-lille.fr](https://www.google.com/url?sa=E&q=mailto:assi.nguessan@polytech-lille.fr" \t "https://aistudio.google.com/prompts/_blank)

## Copyright © 2015 John Wiley & Sons, Ltd.

Page 2 :

1162  
A. N'GUESSAN ET I. C. GERALDO  
Ce problème d'optimisation sous contraintes peut être résolu grâce à différentes approches classiques. La plupart de ces méthodes itératives nécessitent d'abord les dérivées premières et même secondes de la fonction objectif. Une revue complète est donnée, par exemple, dans [4] et [1]. La méthode itérative de base la plus souvent utilisée et citée est celle de Newton-Raphson. Elle implique néanmoins le calcul et l'inversion de la matrice Hessienne à chaque itération. Cette matrice Hessienne peut être coûteuse et difficile à obtenir si la fonction de probabilité (fonction objectif) prend une forme complexe ou si la dimension des paramètres augmente avec la dimension des données. Dans ce cas, les méthodes quasi-Newton représentent une alternative attrayante, du fait qu'elles calculent une approximation de l'inverse de la matrice Hessienne en utilisant l'expression du gradient. L'algorithme de Broyden–Fletcher–Goldfarb–Shanno (BFGS) décrit dans [5–8] est l'une des méthodes quasi-Newton les plus populaires. Il existe également d'autres méthodes qui n'utilisent pas de dérivées, telles que l'algorithme de Nelder-Mead [9].  
Toutes ces méthodes classiques ont été adaptées au problème du maximum de vraisemblance contraint (voir par exemple, [10-12]). D'autres auteurs (par exemple [13, 14]) suggèrent également des algorithmes de Minorisation-Maximisation (MM) pour résoudre des situations particulières. Malgré ces contributions significatives, les études de cas pratiques restent très sensibles, et plusieurs complications peuvent compromettre les performances de ces algorithmes traditionnels, en particulier dans le cas de données discrètes multivariées, qui sont les suivantes :

1. la matrice Hessienne ou une approximation peut être coûteuse en termes de calcul,
2. elle peut ne pas être définie positive, c'est-à-dire que l'inversion n'est pas possible,
3. pour des données de dimensions importantes, la résolution du système linéaire de recherche de solution peut être coûteuse,
4. si des contraintes de paramètres ou des contraintes limites apparaissent, alors la mise à jour elle-même nécessite des modifications adaptées, et
5. le choix d'un vecteur solution initial permettant une convergence rapide reste une clé importante pour toutes les méthodes itératives.

Malgré les nombreuses solutions et garanties apportées par les résultats scientifiques, nous sommes toujours confrontés à la complexité de plus en plus grande des algorithmes numériques et au fait qu'ils ne sont pas accessibles aux non-spécialistes.

Dans ce contexte particulier, N'Guessan et Truffier [15] et N’Guessan [16] ont remplacé le problème du maximum de vraisemblance contraint par le problème de la résolution d'un système d'équations non linéaires contraint. Ces auteurs ont prouvé l'existence de solutions mais ils n'ont considéré que quelques simulations et comparé leurs résultats uniquement à la méthode de Newton-Raphson. Dans cet article, nous généralisons les résultats de ces derniers auteurs et proposons des algorithmes génériques, dont l'un est celui de N’Guessan [16].

Nos résultats sont donc organisés comme suit :

- la section 2 définit le problème, les conditions associées et les résultats obtenus. Les structures générales de notre algorithme y sont également présentées.

- Dans la section 3, nous décrivons rapidement les algorithmes classiques d'optimisation sous contraintes en concurrence avec les nôtres.

- Dans la section 4, nous présentons une étude de cas sur la modélisation des données d'accidents de la route.

Les principaux résultats de la section 2 sont utilisés pour obtenir des formes explicites de l'estimation de l'effet moyen d'une mesure de sécurité routière et des risques de gravité associés. Nous présentons également l'algorithme générique sous une forme plus accessible en pratique.

Dans la section 5, nous nous concentrons sur des simulations avec différents vecteurs initiaux et différentes valeurs du nombre d'accidents de la route. L'erreur quadratique moyenne est ensuite utilisée comme critère de comparaison entre notre algorithme et d'autres méthodes classiques utilisant ou non la matrice Hessienne. Nous terminons par une conclusion en section 6.

## 2. CADRE DU PROBLÈME ET RÉSULTATS PRINCIPAUX 2.1. Cadre du problème Considérons x = (x₁, ..., xR)ᵀ un vecteur de dimension R (R > 2) constitué de données observées et xᵢ (i = 1, ..., R) étant une valeur entière ou nulle telle que n = Σᵢ=₁ᴿ xᵢ, (n > 0). Nous nous concentrons alors sur la maximisation d'une fonction de probabilité notée l(Θ, x) où Θ ∈ ℝᵈ est un vecteur de dimension d (1 < d < R) constitué de paramètres inconnus et sous la contrainte h(Θ) = 0. Ce problème est équivalent à max\_{Θ∈ℝᵈ} l(Θ,x) sous la contrainte h(Θ) = 0. (1) Au fur et à mesure de l'article, nous supposons les conditions suivantes : (H₁) Θ = (θ, φᵀ)ᵀ, θ > 0 et φ ∈ ℝᵈ⁻¹ tel que φⱼ > 0. Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 3 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1163  
(H₂) Les fonctions l : ℝᵈ → ℝ, Θ ↦ l(Θ, x) et h : ℝᵈ → ℝ, Θ ↦ h(Θ) sont continûment différentiables par rapport à Θ.  
(H₃) ∂l/∂θ = 0, ∂h/∂φⱼ ≠ 0 (j = 1, ..., d − 1).  
(H₄) Il existe une constante η non nulle telle que (∇φh, φ) = η où ∇φ est l'opérateur gradient, (·, ·) est le produit scalaire classique (par rapport à la matrice identité de ℝᵈ⁻¹).  
(H₅) Étant donné Θ = (θ, φᵀ)ᵀ le maximum de l(Θ, x) s'il existe. Supposons que pour un θ donné, il existe une matrice Dφ,x non singulière de dimension (d − 1) × (d − 1), un vecteur Bx de dimension (d − 1) × 1 tels que le système non linéaire (∇φl) − (1/η)((∇φl), φ)(∇φh) = 0d−1 ; h(Θ) = 0est équivalent au système linéaire par rapport à φ[ Dφ,x (∇φh)φ ] [ φ ] = [ Bx ] , h(Θ) = 0, [ (∇φh)ᵀφ 0 ] [ λ ] [ η ]où0d−1 = (0, ..., 0) ∈ ℝᵈ⁻¹. (H₆) Il existe deux fonctions g₁ : ℝ → ℝ inversible et g₂ : ℝᵈ⁻¹ → ℝ telles que l'équation ∂l/∂θ = 0 est équivalente à g₁(θ) − g₂(φ; x) = 0.`

Remarque 1  
La condition (H₁) permet de préciser un peu plus où se situe le vecteur Θ. En particulier, supposer que les composantes du vecteur Θ sont strictement positives est relatif au principe du maximum de vraisemblance où le logarithme de la fonction de probabilité utilisée est maximisé. Nous sommes donc amenés à prendre le logarithme de certaines composantes du vecteur paramètre comme le montre l'étude de cas présentée plus loin. La séparation des composantes de Θ en deux sous-ensembles est liée au principe d'estimation alternée ou cyclique des composantes que nous présentons dans notre approche. Les conditions (H₂) à (H₄) permettent de caractériser la structure de la contrainte h. En particulier, nous notons que h dépend essentiellement du sous-vecteur φ. Les conditions (H₅) et (H₆) spécifient la structuration de l'estimation par blocs de Θ, qui consiste à obtenir linéairement φ, θ étant fixé et vice versa. Ainsi, nous pouvons commencer la procédure d'estimation en initialisant la première composante. Elles montrent également que notre méthode n'utilise plus les dérivées secondes de l(Θ, x), ce qui signifie ni la matrice Hessienne ni une approximation.

Lemme 1  
Sous les hypothèses (H₁)-(H₄), la solution Θ du problème (1), si elle existe, est également une solution du système d'équations non linéaires suivant :  
∂l/∂θ = 0 et (∇φl) − (1/η)((∇φl)ᵀφ)(∇φh) = 0d−1 (2)

## Preuve Le problème (1) est équivalent à la maximisation de la fonction L(Θ, x) = l(Θ, x) – λh(Θ) (3) où λ est le multiplicateur de Lagrange. Soit Θ̂ la solution de L(Θ, x), si elle existe, telle que h(Θ̂) = 0 alors (∇L)│\_{Θ̂} = (∇l)│\_{Θ̂} – λ̂(∇h)│\_{Θ̂} = 0d (4) où λ̂ = λ(Θ̂). En considérant (H₁) à (H₃), le système (4) est équivalent à ∂l/∂θ = 0 et ∂l/∂φⱼ − λ̂ ∂h/∂φⱼ = 0, (j = 1, ..., d − 1). Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 4 :

1164  
A. N'GUESSAN ET I. C. GERALDO  
En pré-multipliant ce dernier système par φⱼ et en sommant par rapport à l'indice j, nous obtenons  
Σ\_{j=1}^{d-1} φⱼ(∂l/∂φⱼ) − λ̂ Σ\_{j=1}^{d-1} φⱼ(∂h/∂φⱼ) = 0.  
Cette équation est équivalente à  
((∇φl), φ) – λ̂((∇φh), φ) = 0.  
D'où l'égalité λ̂ = λ(Θ̂) = (1/η)((∇φl), φ) en utilisant (H₄). Nous obtenons alors (2) par substitution de λ̂ dans (4). □

Théorème 1  
Sous les hypothèses (H₁)-(H₆), l'estimateur du maximum de vraisemblance contraint Θ̂ = (θ̂, φ̂ᵀ)ᵀ de Θ est donné par  
φ̂ = D\_{θ,x}⁻¹ Bx  
θ̂ = g₁⁻¹(g₂ (φ̂; x)). (5)

Preuve  
En utilisant les hypothèses (H₁)-(H₅) et le Lemme 1, le sous-vecteur φ̂ est la solution du système  
[ D\_{θ,x} (∇φh)φ̂ ] [ φ̂ ] = [ Bx ] avec h(Θ̂) = 0.  
[ (∇φh)ᵀφ̂ 0 ] [ λ̂ ] [ η ]  
Nous posons  
Ω\_{θ,x} = [ D\_{θ,x} (∇φh)φ̂ ]  
[ (∇φh)ᵀφ̂ 0 ]  
L'obtention de φ̂ consiste à inverser la matrice Ω\_{θ,x}, en utilisant les résultats généraux relatifs au complément de Schur [17–19]. En effet, sous les hypothèses (H₅) et (H₃), D\_{θ,x}⁻¹ existe et  
(∇φh)ᵀφ̂ D\_{θ,x}⁻¹ (∇φh)φ̂ = ||(∇φh)φ̂||²\_{D\_{θ,x}⁻¹} > 0.  
où ||(∇φh)φ̂||²\_{D\_{θ,x}⁻¹} = ((∇φh)φ̂, (∇φh)φ̂)\_{D\_{θ,x}⁻¹} est la norme du vecteur (∇φh)φ̂ par rapport à la matrice D\_{θ,x}⁻¹. Par conséquent, l'inversion de Ω\_{θ,x} est possible et nous avons  
Ω\_{θ,x}⁻¹ = [ M\_{θ,x} −||(∇φh)φ̂||^{-2}\_{D\_{θ,x}⁻¹} D\_{θ,x}⁻¹ (∇φh)φ̂ ] (6)  
[ −||(∇φh)φ̂||^{-2}\_{D\_{θ,x}⁻¹} (∇φh)ᵀφ̂ D\_{θ,x}⁻¹ −||(∇φh)φ̂||^{-2}\_{D\_{θ,x}⁻¹} ]  
où  
M\_{θ,x} = D\_{θ,x}⁻¹ − ||(∇φh)φ̂||^{-2}\_{D\_{θ,x}⁻¹} D\_{θ,x}⁻¹ (∇φh)φ̂ (∇φh)ᵀφ̂ D\_{θ,x}⁻¹  
Les détails techniques de ce résultat sont donnés en annexe. Nous déduisons φ̂ en multipliant Ω\_{θ,x}⁻¹ par le vecteur (Bᵀx, η)ᵀ. La condition (H₆) permet d'obtenir θ̂ en fonction de φ̂. □

## Corollaire 1 En plus des conditions du Théorème 1, nous supposons également que (H₇) R = 2r, d = r + 1 (r étant un entier non nul), θ > 0, φⱼ > 0 tel que Σ\_{j=1}^{r} φⱼ = 1. Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 5 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1165  
(H₈) Il existe un vecteur Z = (w₁, ..., wᵣ)ᵀ, wⱼ > 0 tel que  
D\_{θ,x} = Λ\_{θ,Z} − θBxZᵀ  
où Λ\_{θ,Z} = Diag(1 + θw₁, ..., 1 + θwᵣ) est une matrice diagonale r × r,  
Bx = [ (x₁ + x₁₊ᵣ)/n ... (xᵣ + x₂ᵣ)/n ]ᵀ ∈ ℝʳ  
est un vecteur de dimension r ainsi défini avec n = Σ\_{j=1}^{r}(xⱼ + xⱼ₊ᵣ).  
Alors  
φ̂ⱼ = (1 / (1 - Σ\_{m=1}^{r} (θw\_m x\_{.m}) / (1 + θw\_m))) × (1 / (1 + θwⱼ)) × ((xⱼ + xⱼ₊ᵣ)/n)  
θ̂ = g₁⁻¹(g₂(φ̂, x)).  
où x\_{.m} = xₘ + xₘ₊ᵣ.  
Preuve  
En utilisant le fait que Λ\_{θ,Z}⁻¹ existe et 0 < θZᵀΛ\_{θ,Z}⁻¹Bx < 1, nous montrons que D\_{θ,x}⁻¹ existe et que  
D\_{θ,x}⁻¹ = Λ\_{θ,Z}⁻¹ + θ(1 - θ(Z, Bx)\_{Λ\_{θ,Z}⁻¹})⁻¹ Λ\_{θ,Z}⁻¹BxZᵀΛ\_{θ,Z}⁻¹  
Après quelques manipulations matricielles, nous montrons que  
φ̂ = D\_{θ,x}⁻¹Bx = (1 / (1 - θ(Z, Bx)\_{Λ\_{θ,Z}⁻¹})) Λ\_{θ,Z}⁻¹Bx  
où  
1 - θ(Z, Bx)\_{Λ\_{θ,Z}⁻¹} = 1 - (θ/n) Σ\_{m=1}^{r} (w\_m x\_{.m}) / (1 + θw\_m).  
Nous déduisons alors l'expression de φ̂ⱼ (j = 1, ..., r) du Corollaire 1. Quelques lignes de développement se trouveront en annexe. □

Remarque 2  
Le corollaire permet d'avoir des formules plus explicites pour la composante φ̂ et généralise ainsi les résultats de [16]. Nous pouvons alors démarrer les procédures d'optimisation de l(Θ, x), en travaillant composante après composante. Par exemple, lorsque θ⁽⁰⁾ = 0 alors φ̂ⱼ⁽¹⁾ = (xⱼ + xⱼ₊ᵣ)/n, (j = 1, ..., r) et θ̂⁽¹⁾ = g₁⁻¹(g₂(φ̂⁽¹⁾)) et ainsi de suite. Nous automatisons ainsi notre processus d'estimation de θ̂ en nous concentrant par exemple sur l'initialisation de sa première composante. Ainsi, la recherche d'un vecteur initial contraint est ramenée à un point initial. Nous décrivons la structure générale de notre méthode d'estimation par rapport au Théorème 1 dans les sections suivantes. Ensuite, nous donnons une version plus pratique dans le cadre du Corollaire 1.

## 2.2. Cadre général de l'algorithme cyclique Cette approche générale permet d'alterner l'estimation de Θ entre ses deux composantes θ et φ. Pour commencer la procédure, nous initialisons d'abord la composante θ⁽⁰⁾. Ensuite, nous calculons φ⁽⁰⁾ = D\_{θ⁽⁰⁾,x}⁻¹Bx et définissons Θ⁽⁰⁾ = (θ⁽⁰⁾, (φ⁽⁰⁾)ᵀ)ᵀ. Inversement, nous pouvons, si les données le permettent, initialiser φ⁽⁰⁾ et obtenir θ⁽⁰⁾ = g₁⁻¹(g₂(φ⁽⁰⁾, x)). À l'étape k (k > 0), nous calculons θ^{(k)} = g₁⁻¹(g₂(φ^{(k-1)}, x)), puis φ^{(k)} grâce à θ^{(k)}, Bx et D\_{θ^{(k)},x} qui est mis à jour. Les conditions de maximisation de l(Θ, x) et les hypothèses sont Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 6 :

1166  
A. N'GUESSAN ET I. C. GERALDO  
Algorithme 1 Schéma général de l'algorithme cyclique  
Requis : x = (x₁, ..., xR)ᵀ, ε₁ > 0, ε₂ > 0 deux précisions, η.  
Assurer : Θ̂ l'EMV de Θ, k₀ le nombre d'itérations.  
1: Calculer Bx, θ̂⁽⁰⁾, D\_{θ̂⁽⁰⁾,x}⁻¹, φ̂⁽⁰⁾ = D\_{θ̂⁽⁰⁾,x}⁻¹Bx, Θ̂⁽⁰⁾ = (θ̂⁽⁰⁾, (φ̂⁽⁰⁾)ᵀ)ᵀ, l(Θ̂⁽⁰⁾, x) et (∇φh)Θ̂⁽⁰⁾.  
2: k = 0.  
3: STOP = 0.  
4: tant que STOP ≠ 1 faire  
5: Calculer θ̂⁽ᵏ⁺¹⁾ = g₁⁻¹(g₂(φ̂⁽ᵏ⁾, x)) et D\_{θ̂⁽ᵏ⁺¹⁾,x}⁻¹.  
6: Calculer φ̂⁽ᵏ⁺¹⁾ = D\_{θ̂⁽ᵏ⁺¹⁾,x}⁻¹Bx.  
7: Définir Θ̂⁽ᵏ⁺¹⁾ = (θ̂⁽ᵏ⁺¹⁾, (φ̂⁽ᵏ⁺¹⁾)ᵀ)ᵀ et calculer l(Θ̂⁽ᵏ⁺¹⁾, x), (∇φh)Θ̂⁽ᵏ⁺¹⁾.  
8: si |( (∇φh)Θ̂⁽ᵏ⁺¹⁾ )ᵀ φ̂⁽ᵏ⁺¹⁾ − η| > ε₁ ou |l(Θ̂⁽ᵏ⁺¹⁾, x) − l(Θ̂⁽ᵏ⁾, x)| > ε₂ alors  
9: STOP = 0.  
10: sinon  
11: STOP = 1.  
12: Définir Θ̂ = Θ̂⁽ᵏ⁺¹⁾.  
13: fin si  
14: k = k + 1.  
15: fin tant que  
16: k₀ = k.  
17: Calculer h(Θ̂), l(Θ̂, x), (∇φh)Θ̂.

testées. Le processus est ainsi répété jusqu'à ce que toutes les conditions soient satisfaites. En utilisant le Corollaire 1, nous obtenons l'Algorithme 1.  
Dans cette version, nous notons que nous pouvons démarrer la procédure en utilisant les données du problème via le vecteur Bx. Ainsi, en pratique, notre algorithme est automatisé dès que les données du problème sont entrées, en utilisant φ̂ⱼ⁽⁰⁾ = Bx,j puis θ̂⁽¹⁾ = g₁⁻¹(g₂(φ̂⁽⁰⁾, x)) et φ̂ⱼ⁽¹⁾ = (1/Δ) × (1/(1 + θ̂⁽¹⁾wⱼ))Bx,j et ainsi de suite. Dans l'ensemble, le choix de Θ̂⁽⁰⁾ peut se faire sur l'une ou l'autre composante de Θ̂ selon les données du problème. Nous pouvons également noter que les dérivées partielles secondes de l(Θ, x) ne sont plus utilisées dans notre algorithme.  
Le but de cet article n'est pas de réaliser une étude théorique des propriétés de l'algorithme cyclique. Nous nous concentrons plutôt sur les propriétés numériques de cet algorithme à travers une application. Néanmoins, on peut remarquer que l'estimation de Θ̂ avec l'algorithme cyclique n'utilise plus le calcul et l'inversion de la matrice des dérivées secondes de la fonction objectif. Ceci suggère que l'estimation est améliorée, du moins en termes de temps de calcul.

## 3. UN BREF APERÇU DE QUELQUES ALGORITHMES D'OPTIMISATION CLASSIQUES 3.1. Méthode de Newton-Raphson La méthode de Newton-Raphson pour maximiser une fonction l(Θ) avec Θ ∈ ℝᵈ, consiste en un plan itératif prenant la forme Θ̂⁽ᵏ⁺¹⁾ = Θ̂⁽ᵏ⁾ – (∇²l)⁻¹\_{Θ̂⁽ᵏ⁾} (∇l)\_{Θ̂⁽ᵏ⁾} (7) où (∇²l) représente la matrice Hessienne de l et Θ̂⁽⁰⁾ est la solution initiale. L'avantage de cette méthode est qu'elle converge lorsque la solution initiale Θ̂⁽⁰⁾ est proche de la vraie solution, ce qui est inconnu en pratique. Cependant, plusieurs problèmes peuvent survenir lors de l'utilisation de la méthode de Newton-Raphson. Le premier étant que l'évaluation et l'inversion de la matrice Hessienne peuvent apparaître très Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 7 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1167  
Algorithme 2  
Requis : x = (x₁, ..., x₂ᵣ)ᵀ, ε₁ > 0, ε₂ > 0, η = 1, Z = (w₁, ..., wᵣ)ᵀ, 1ᵣ = (1, ..., 1)ᵀ ∈ ℝʳ.  
Assurer : Θ̂ l'EMV de Θ, k₀ le nombre d'itérations.  
1: Calculer n = Σ\_{j=1}^{r}(xⱼ + xⱼ₊ᵣ), Bx = n⁻¹(x₁ + x₁₊ᵣ, ..., xᵣ + x₂ᵣ)ᵀ  
2: θ̂⁽⁰⁾ = 0  
3: Pour j = 1, ..., r, Calculer Bx,j = n⁻¹(xⱼ + xⱼ₊ᵣ)  
4: Θ̂⁽⁰⁾ = (0, Bᵀx)  
5: k = 0.  
6: STOP = 0.  
7: tant que STOP ≠ 1 faire  
8: Calculer θ̂⁽ᵏ⁺¹⁾ = g₁⁻¹(g₂(φ̂⁽ᵏ⁾, x))  
9: Pour m = 1, ..., r, Δ⁽ᵏ⁺¹⁾\_{n,m} = (θ̂⁽ᵏ⁺¹⁾wₘ(xₘ + xₘ₊ᵣ)) / (n(1 + θ̂⁽ᵏ⁺¹⁾wₘ)) et Δ⁽ᵏ⁺¹⁾\_{n,•} = 1 − Σ\_{m=1}^{r} Δ⁽ᵏ⁺¹⁾\_{n,m}.  
10: Pour j = 1, ..., r, calculer φ̂ⱼ⁽ᵏ⁺¹⁾ = Δ⁽ᵏ⁺¹⁾\_{n,•} × (Bx,j / (1 + θ̂⁽ᵏ⁺¹⁾wⱼ))  
11: Calculer φ̂⁽ᵏ⁺¹⁾ = (φ̂₁⁽ᵏ⁺¹⁾, ..., φ̂ᵣ⁽ᵏ⁺¹⁾)ᵀ.  
12: Définir Θ̂⁽ᵏ⁺¹⁾ = (θ̂⁽ᵏ⁺¹⁾, (φ̂⁽ᵏ⁺¹⁾)ᵀ)ᵀ.  
13: si |1ᵀᵣ φ̂⁽ᵏ⁺¹⁾ − η| > ε₁ ou |l(Θ̂⁽ᵏ⁺¹⁾, x) − l(Θ̂⁽ᵏ⁾, x)| > ε₂ alors  
14: STOP = 0.  
15: sinon  
16: STOP = 1.  
17: Définir Θ̂ = Θ̂⁽ᵏ⁺¹⁾.  
18: fin si  
19: k = k + 1.  
20: fin tant que  
21: k₀ = k.  
22: Calculer h(Θ̂), l(Θ̂, x).

difficile et coûteuse, numériquement parlant, si d, la dimension de l'espace des paramètres, est élevée ou si l'expression de l(Θ) est complexe. Une autre possibilité consiste à procéder en deux étapes à chaque itération. Le système linéaire  
(∇²l)\_{Θ̂⁽ᵏ⁾} s = (∇l)\_{Θ̂⁽ᵏ⁾}  
est résolu puis on procède à la mise à jour Θ̂⁽ᵏ⁺¹⁾ = Θ̂⁽ᵏ⁾ + s. Mais cela ne change pas fondamentalement la situation car rien ne prouve que la matrice Hessienne est positive et donc inversible. Le deuxième problème possible vient du fait que lorsque Θ̂⁽ᵏ⁾ est loin de la vraie solution, la méthode de Newton n'est pas une méthode ascendante, c'est-à-dire que nous n'avons pas nécessairement l(Θ̂⁽ᵏ⁺¹⁾) > l(Θ̂⁽ᵏ⁾).

## 3.2. Les algorithmes de minorisation-maximisation (MM) Dans les problèmes de maximisation, le premier M de l'acronyme MM signifie minorer et le second M maximiser. La philosophie MM consiste à substituer un problème d'optimisation simple à un problème d'optimisation difficile. Elle peut être résumée comme suit (par exemple [20]). La première étape M d'un algorithme MM de minorisation-maximisation consiste à définir une fonction g\_{Θ̂⁽ᵏ⁾}(Θ) qui minore l(Θ) au point Θ̂⁽ᵏ⁾, c'est-à-dire g\_{Θ̂⁽ᵏ⁾}(Θ) ≤ l(Θ), pour tout Θ, g\_{Θ̂⁽ᵏ⁾}(Θ̂⁽ᵏ⁾) = l(Θ̂⁽ᵏ⁾). Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 8 :

1168  
A. N'GUESSAN ET I. C. GERALDO  
où Θ̂⁽ᵏ⁾ représente l'itéré courant. Dans la deuxième étape M, l'itéré suivant Θ̂⁽ᵏ⁺¹⁾ est produit en maximisant la fonction minorante g\_{Θ̂⁽ᵏ⁾}(Θ) plutôt que la fonction réelle l(Θ).

3.3. Les méthodes quasi-Newton  
La principale caractéristique des méthodes quasi-Newton est qu'elles utilisent une approximation inverse de la matrice Hessienne (∇²l)\_{Θ̂⁽ᵏ⁾} à chaque itération. Elles permettent ainsi d'éviter le calcul (et l'inversion) de la matrice Hessienne. Partant de l'approximation de premier ordre suivante :  
(∇l)\_{Θ̂⁽ᵏ⁾} – (∇l)\_{Θ̂⁽ᵏ⁺¹⁾} ≈ (∇²l)\_{Θ̂⁽ᵏ⁾} (Θ̂⁽ᵏ⁾ – Θ̂⁽ᵏ⁺¹⁾). (8)  
et en notant J\_{k+1} l'approximation de l'inverse de la matrice Hessienne (∇²l)\_{Θ̂⁽ᵏ⁾},  
sₖ = Θ̂⁽ᵏ⁺¹⁾ – Θ̂⁽ᵏ⁾  
yₖ = (∇l)\_{Θ̂⁽ᵏ⁺¹⁾} – (∇l)\_{Θ̂⁽ᵏ⁾},  
la condition (8) est équivalente à J\_{k+1}yₖ = sₖ et appelée équation de la sécante. Pour déterminer J\_{k+1}, nous utilisons la formule BFGS, qui est considérée comme la plus efficace des formules quasi-Newton [3, 4]. Son nom venant des noms de ses auteurs (Broyden, Fletcher, Goldfarb, Shanno), elle consiste en une mise à jour de la forme  
J\_{k+1} = (I – γₖsₖyₖᵀ) Jₖ (I – γₖyₖsₖᵀ) + γₖsₖsₖᵀ (9)  
avec  
γₖ = 1 / (yₖᵀsₖ) (10)  
Les méthodes quasi-Newton présentent plusieurs avantages : elles ne nécessitent pas de dérivées premières, elles économisent l'inversion de la matrice Hessienne, elles restent "proches" de la méthode de Newton et la matrice Jₖ est définie positive, ce qui permet de garantir leur "succès" (convergence).

3.4. Les algorithmes d'optimisation sans dérivées  
Les algorithmes d'optimisation sans dérivées (DFO), comme leur nom l'indique, n'utilisent pas la dérivée de la fonction à optimiser. Ces algorithmes n'approximent donc pas le gradient mais déterminent les itérations successives à partir des valeurs de la fonction sur un ensemble fini de points.  
Parmi les algorithmes DFO, la méthode de Nelder-Mead est particulièrement utilisée. C'est une méthode heuristique (donnant rapidement une solution exploitable, pas nécessairement optimale ou exacte) proposée par Nelder et Mead [9] en 1965. Elle consiste à maximiser une fonction l(Θ), Θ ∈ ℝᵈ, à partir d'un simplexe (un ensemble de d + 1 points) de ℝᵈ. À chaque itération, le sommet avec la plus petite valeur de l est remplacé par un autre point. Les itérations sont répétées jusqu'à ce que les images des sommets par l soient suffisamment proches.

4. UNE ÉTUDE DE CAS  
Nos résultats et algorithmes sont appliqués pour modéliser des données d'accidents de la route lorsqu'une mesure de sécurité routière (aménagement de carrefour, revêtement d'une section d'autoroute, etc.) est appliquée à un site expérimental présentant plusieurs types d'accidents mutuellement exclusifs (accidents mortels, blessés graves, blessés légers, dommages matériels, etc.) sur une période de temps déterminée.

## 4.1. Modèle statistique Soit r > 1 le nombre de différents types d'accidents survenant sur le site expérimental, X₁ = (X₁₁, ..., X₁ᵣ)ᵀ (respectivement X₂ = (X₂₁, ..., X₂ᵣ)ᵀ) le vecteur aléatoire donnant le nombre d'accidents de chaque type sur le site expérimental dans la période avant (respectivement après) l'application de la mesure. Les vecteurs X₁ et X₂ sont tels que X₁ⱼ (respectivement X₂ⱼ), j = 1, ..., r Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 9 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1169  
représente le nombre d'accidents de type j survenus dans la période "avant" (respectivement "après"). Afin de prendre en compte certains facteurs externes (flux de trafic, variation de la limitation de vitesse, conditions météorologiques, etc.), le site expérimental est associé à une zone de contrôle où la mesure de sécurité n'a pas été directement appliquée. Soit C = (c₁, ..., cᵣ)ᵀ un vecteur r × 1 tel que cⱼ désigne le rapport du nombre d'accidents de type j pour la période après à la période avant dans la zone de contrôle sur la même période. Le vecteur C est supposé fixe et connu et ses composantes sont appelées coefficients de contrôle. Nous adoptons la modélisation multinomiale avant-après proposée par N'Guessan et al. [21] lorsque le nombre de sites est égal à 1. Nous supposons donc que (X₁, X₂) suit la distribution multinomiale  
(X₁, X₂) ∼ M(n; p₁(Θ), p₂(Θ))  
où n est le nombre total d'accidents enregistrés sur le site expérimental dans les deux périodes, et les composantes p\_{tj}(Θ) de p\_t(Θ); (t = 1, 2) sont données par  
p₁ⱼ(Θ) = φⱼ / (1 + θCᵀφ) et p₂ⱼ(Θ) = (θcⱼφⱼ) / (1 + θCᵀφ) ∀j = 1, ..., r (11)  
où θ > 0, 0 < φⱼ < 1 avec Σ\_{j=1}^{r} φⱼ = 1. Le scalaire θ représente l'effet moyen inconnu de la mesure de sécurité routière, tandis que chaque φⱼ (j = 1, 2, ..., r) désigne le risque global d'accident de type j avant et après l'application de la mesure de sécurité routière. Une valeur de θ significativement inférieure à 1 signifie que l'introduction des changements a diminué le nombre d'accidents. Le vecteur paramètre Θ = (θ, φᵀ)ᵀ est soumis aux contraintes suivantes θ > 0, 0 < φⱼ < 1 et la contrainte linéaire  
h(Θ) = 0, avec h(Θ) = φᵀ1ᵣ − 1. (12)  
Étant donné des données observées x = (x₁; x₂) où x₁ = (x₁₁, ..., x₁ᵣ) et x₂ = (x₂₁, ..., x₂ᵣ) et en utilisant l'expression des probabilités cellulaires de (11), la fonction de probabilité liée au vecteur aléatoire (X₁, X₂) est donnée par  
L(Θ, x) = (n! / (Π\_{j=1}^{r} x₁ⱼ!x₂ⱼ!)) Π\_{j=1}^{r} (φⱼ / (1 + θCᵀφ))^{x₁ⱼ} ((θcⱼφⱼ) / (1 + θCᵀφ))^{x₂ⱼ} (13)  
Le vecteur inconnu Θ sous les contraintes (12) est estimé en maximisant L(Θ, x) lorsque nous connaissons le vecteur x. Il est équivalent de maximiser l(Θ, x) = log(L(Θ, x)) à une constante additive près comme  
l(Θ, x) = Σ\_{j=1}^{r} [x\_{.j} log(φⱼ) + x₂ⱼ log(θ) − x\_{.j} log(1 + θ Σ\_{m=1}^{r} cₘφₘ)] (14)  
avec x\_{.j} = x₁ⱼ + x₂ⱼ. Différentes méthodes itératives [13, 22] peuvent être utilisées pour résoudre le problème d'estimation du maximum de vraisemblance contraint de Θ. La méthode de Newton-Raphson ou la méthode du score de Fisher pour le calcul d'une solution sont probablement les méthodes itératives les plus largement utilisées. Chacune d'elles calcule les dérivées du second ordre de l(Θ, x) et nécessite un vecteur de départ Θ̂⁽⁰⁾. Cependant, de telles méthodes peuvent échouer si Θ̂⁽⁰⁾ n'est pas assez proche de la vraie valeur, ce qui en pratique est inconnu.

Remarque 3  
L'expression l(Θ, x) donnée dans la formule (14) et les contraintes relatives aux paramètres θ et φ du modèle (11) justifient l'hypothèse (H₁). L'hypothèse (H₂) est naturellement vérifiée ainsi que (H₃) avec ∇φh = (1, ..., 1)ᵀ ∈ ℝʳ. De même, nous montrons que (∇φh)ᵀφ = 1 = η.

## 4.2. Algorithme cyclique pour l'estimation de l'effet moyen et des différents risques En dérivant l(Θ, x) – λh(Θ) par rapport à θ et φⱼ (j = 1, ..., r), nous montrons que Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 10 :

1170  
A. N'GUESSAN ET I. C. GERALDO  
∂l/∂θ = (Σ\_{j=1}^{r} x₂ⱼ)/θ − (nCᵀφ)/(1 + θCᵀφ)  
∂l/∂φⱼ = x\_{.j}/φⱼ − (θcⱼn)/(1 + θCᵀφ) − λ, j = 1, ..., r. (15)  
où λ est le multiplicateur de Lagrange. En fixant les dérivées partielles de (15) à zéro, nous montrons que Θ̂, s'il existe, est la solution du système d'équations non linéaires suivant :  
(Σ\_{j=1}^{r} x₂ⱼ)/θ̂ − (nCᵀφ̂)/(1 + θ̂Cᵀφ̂) = 0  
x\_{.j}/φ̂ⱼ − (θ̂cⱼn̂)/(1 + θ̂Cᵀφ̂) − λ̂ = 0, j = 1, ..., r. (16)  
1ᵀᵣφ̂ = 1  
où n̂ = n(1 + θ̂Cᵀφ̂)⁻¹.

Corollaire 2  
Les composantes de la solution Θ̂ de (16) sont obtenues avec les expressions suivantes :  
θ̂ = x₂.(x₁.)⁻¹(Cᵀφ̂)⁻¹  
φ̂ⱼ = (1 − Σ\_{m=1}^{r} (θ̂cₘx\_{.m})/(n̂(1 + θ̂cₘ)))⁻¹ × (x\_{.j}/(n̂(1 + θ̂cⱼ))) (j = 1, ..., r), (17)  
où x\_{.m} = x₁ₘ + x₂ₘ et x\_{t.} = Σ\_{m=1}^{r} x\_{tm}, (t = 1, 2).

Remarque 4  
La preuve du Corollaire 2 est similaire à celle du Corollaire 1, en prenant Z = C, la fonction g₁ égale à l'identité et la fonction g₂(φ, x) = x₂.(x₁.)⁻¹(Cᵀφ)⁻¹. De plus, en remplaçant λ̂ = λ(Θ̂) = n(1 + Cᵀφ̂)⁻¹ dans la deuxième équation de (16) et en multipliant cette dernière équation par le rapport φ̂ⱼ/n, nous obtenons le système d'équations non linéaires par rapport à φⱼ  
(1 + Cᵀφ̂)⁻¹(1 + θ̂cⱼ)φ̂ⱼ = x\_{.j}/n.  
Nous montrons ensuite (voir [16] pour les détails techniques) que ce dernier système est équivalent à  
D\_{θ̂,x}φ̂ = Bx,  
où D\_{θ̂,x} = Λ\_{θ̂,x} − θ̂BxCᵀ.  
Grâce aux formules explicites des composantes de Θ̂ du Corollaire 2, nous proposons l'Algorithme 3, qui est une forme pratique de l'Algorithme 2.

Remarque 5  
Comme dans l'Algorithme 2, nous aurions pu démarrer l'Algorithme 3 à φ̂⁽⁰⁾ = Bx car Bx est disponible et de plus 1ᵀᵣBx = 1. D'autres solutions initiales peuvent également être utilisées comme le montrent les études numériques dans la section suivante.

## 5. ÉTUDES NUMÉRIQUES Cette section se concentre sur l'étude numérique de l'algorithme cyclique (AC) appliqué au modèle multinomial présenté à la Section 4. À la connaissance des auteurs, trois critères sont habituellement étudiés sur les algorithmes itératifs, qui sont les suivants : la robustesse (l'algorithme doit bien fonctionner pour tous Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 11 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1171  
Algorithme 3  
Requis : x = (x₁₁, ..., x₁ᵣ, x₂₁, ..., x₂ᵣ)ᵀ, ε₁ > 0, ε₂ > 0.  
Assurer : Θ̂ l'EMV de Θ, k₀ le nombre d'itérations.  
1: Calculer n, Bx = n⁻¹(x\_{.1}, ..., x\_{.r})ᵀ, x₁., x₂..  
2: φ̂⁽⁰⁾ = Bx, Θ̂⁽⁰⁾ = (0, Bᵀx).  
3: k = 0.  
4: STOP = 0.  
5: tant que STOP ≠ 1 faire  
6: Calculer θ̂⁽ᵏ⁺¹⁾ = x₂.(x₁.)⁻¹(Cᵀφ̂⁽ᵏ⁾)⁻¹  
7: Calculer Δ⁽ᵏ⁺¹⁾\_{n,m} = (x\_{.m}/n) × cₘθ̂⁽ᵏ⁺¹⁾ / (1 + cₘθ̂⁽ᵏ⁺¹⁾) pour m = 1, ..., r.  
8: Calculer Δ⁽ᵏ⁺¹⁾\_{n,•} = 1 − Σ\_{m=1}^{r} Δ⁽ᵏ⁺¹⁾\_{n,m}.  
9: Calculer φ̂ⱼ⁽ᵏ⁺¹⁾ = (Δ⁽ᵏ⁺¹⁾\_{n,•} / (n(1 + θ̂⁽ᵏ⁺¹⁾cⱼ))) × x\_{.j} pour j = 1, ..., r.  
10: Définir φ̂⁽ᵏ⁺¹⁾ = (φ̂₁⁽ᵏ⁺¹⁾, ..., φ̂ᵣ⁽ᵏ⁺¹⁾)ᵀ.  
11: Définir Θ̂⁽ᵏ⁺¹⁾ = (θ̂⁽ᵏ⁺¹⁾, (φ̂⁽ᵏ⁺¹⁾)ᵀ)ᵀ.  
12: si |1ᵀᵣφ̂⁽ᵏ⁺¹⁾| > ε₁ ou |l(Θ̂⁽ᵏ⁺¹⁾, x) − l(Θ̂⁽ᵏ⁾, x)| > ε₂ alors  
13: STOP = 0  
14: sinon  
15: STOP = 1  
16: Définir Θ̂ = Θ̂⁽ᵏ⁺¹⁾.  
17: fin si  
18: k = k + 1.  
19: fin tant que  
20: k₀ = k.

## les choix raisonnables de la supposition initiale), la précision (l'algorithme doit être capable d'identifier une solution proche des vraies valeurs avec précision) et l'efficacité (l'algorithme ne doit pas nécessiter trop de temps de calcul ou de stockage). Dans notre expérience, la robustesse sera vérifiée par le nombre d'itérations. Si le nombre d'itérations est approximativement le même pour différentes valeurs de départ Θ̂⁽⁰⁾, alors on peut dire que l'algorithme est robuste. Afin d'évaluer la précision de l'algorithme, nous calculons l'erreur quadratique moyenne (EQM) EQM(Θ̂) = (1 / (1 + r)) [ (θ̂ − θ⁰)² + Σ\_{j=1}^{r} (φ̂ⱼ − φ⁰ⱼ)² ] (18) avec Θ̂⁰ = (θ⁰, (φ⁰)ᵀ)ᵀ le vrai vecteur paramètre. L'efficacité sera surveillée par le temps d'unité centrale de traitement (CPU) calculé en secondes. Nous comparons également notre algorithme cyclique à certains des meilleurs algorithmes d'optimisation disponibles. Les méthodes sélectionnées pour cette comparaison sont décrites à la Section 3. C'est-à-dire, l'algorithme de Newton-Raphson, l'algorithme quasi-Newton BFGS [5–8], l'algorithme de Nelder–Mead [9] et l'algorithme de Minorisation-Maximisation (MM) proposé par [14]. Les performances des différents algorithmes en termes de temps de calcul sont comparées avec leurs ratios de temps CPU respectifs calculés comme le ratio entre leur durée moyenne et la durée moyenne de l'algorithme cyclique. Ainsi, le ratio de temps CPU de l'algorithme cyclique est toujours égal à 1. Les calculs présentés dans cette section ont été effectués avec le logiciel R [23] (version 3.0.2) et exécutés sur un PC avec un processeur AMD E-350 à 1.6 GHz. Tout temps d'exécution donné ci-dessous est le temps CPU utilisateur rapporté par la fonction R system.time(). Les algorithmes BFGS et Nelder-Mead sont implémentés en utilisant la fonction constrOptim.nl du package R alabama [24]. L'algorithme de Newton-Raphson est implémenté en utilisant la description donnée à la Section 3 tandis que Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 12 :

1172  
A. N'GUESSAN ET I. C. GERALDO  
l'algorithme MM est implémenté en utilisant l'algorithme proposé dans [14]. Comme le montre l'Algorithme 3, par exemple, la manière habituelle de calculer le nombre d'itérations consiste à compter le nombre de fois qu'un algorithme est répété jusqu'à ce que tous les critères d'arrêt soient satisfaits.  
Les codes R générant nos résultats numériques peuvent être téléchargés depuis le site web [http://www.assi-nguessan.fr](https://www.google.com/url?sa=E&q=http://www.assi-nguessan.fr" \t "https://aistudio.google.com/prompts/_blank).

5.1. Principe de génération des données  
Étant donné r (le nombre de types d'accidents) et n (le nombre total d'accidents), nous générons les composantes du vecteur C = (c₁, ..., cᵣ)ᵀ à partir d'une variable aléatoire uniforme U[0.5; 2.5]. Nous générons la vraie valeur du paramètre θ, notée θ⁰, comme la moyenne d'une variable aléatoire uniforme U[0; 1] et la vraie valeur du vecteur φ, notée φ⁰ = (φ⁰₁, ..., φ⁰ᵣ)ᵀ, à partir d'une distribution de Dirichlet. Ensuite, nous calculons les vraies valeurs  
p⁰₁ⱼ(Θ) = φ⁰ⱼ / (1 + θ⁰ Σ\_{m=1}^{r} cₘφ⁰ₘ) et p⁰₂ⱼ(Θ) = (θ⁰cⱼφ⁰ⱼ) / (1 + θ⁰ Σ\_{m=1}^{r} cₘφ⁰ₘ) ∀j = 1, ..., r  
liées à la distribution multinomiale de (X₁, X₂). Enfin, les données x = (x₁, x₂) sont générées aléatoirement à partir de la distribution multinomiale M(n; p₁ (Θ⁰), p₂(Θ⁰)).

5.2. Résultats  
Les résultats présentés dans cet article correspondent à r ∈ {3; 5} et deux valeurs de n : une petite valeur (n = 50) et une grande valeur (n = 5000). Afin d'explorer toutes les positions de départ possibles, nous avons considéré quatre manières différentes de définir le vecteur paramètre de départ Θ̂⁽⁰⁾ = (θ̂⁽⁰⁾, (φ̂⁽⁰⁾)ᵀ)ᵀ. Le paramètre θ̂⁽⁰⁾ est généré aléatoirement et le vecteur paramètre φ̂⁽⁰⁾ est généré de quatre manières différentes :  
(I₁) Uniforme : φ̂⁽⁰⁾ = (1/r, ..., 1/r)ᵀ.  
(I₂) Proportionnel I : φ̂⁽⁰⁾ = (1/n)(x₁ + x₂).  
(I₃) Aléatoire : φ̂⁽⁰⁾ = (1/s)U où U = (u₁, ..., uᵣ)ᵀ est un vecteur r-dimensionnel dont les composantes sont générées aléatoirement à partir d'une distribution uniforme U[0.05; 0.95] et s = Σ\_{j=1}^{r} uⱼ.  
(I₄) Proportionnel II : φ̂⁽⁰⁾ = (1/n₁)x₁ où n₁ = Σ\_{j=1}^{r} x₁ⱼ. Ce paramétrage de φ̂⁽⁰⁾ est suggéré par certaines hypothèses de base faites dans [21].  
En combinant ces différentes valeurs de r, n et φ̂⁽⁰⁾, nous obtenons 16 scénarios différents. Les résultats présentés dans les Tableaux II-IX correspondent aux valeurs moyennes obtenues pour 1000 simulations pour tous les scénarios.

## *5.2.1. Étude numérique de l'algorithme cyclique.* La performance globale de l'algorithme cyclique est présentée dans le Tableau I. L'efficacité sera étudiée plus tard lorsque nous comparerons l'AC avec les autres algorithmes. Comme mentionné précédemment, nous utilisons l'EQM comme indicateur de la précision de l'algorithme. On remarque que l'EQM est de l'ordre de 10⁻² lorsque n est petit et de 10⁻⁴ lorsque n est grand. Les résultats présentés dans les Tableaux II-IX montrent que les estimations produites par l'AC sont assez proches des vraies valeurs, et lorsque n est grand, les estimations correspondent aux vraies valeurs. Les résultats suggèrent également que l'algorithme cyclique est robuste par rapport à la valeur de départ (ou estimation initiale). Pour chaque valeur de r, 8000 valeurs de départ (correspondant à quatre schémas d'initialisation pour φ̂⁽⁰⁾, deux valeurs de n et 1000 simulations) sont sélectionnées aléatoirement dans l'espace des paramètres. Pour toutes ces valeurs, le nombre d'itérations se situe entre 3 et 4 en moyenne. On peut remarquer que pour une valeur donnée de n, il n'y a pas d'augmentation significative du nombre d'itérations lorsque la dimension de l'espace des paramètres (r + 1) augmente. Lorsque n augmente, une légère augmentation du nombre moyen Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 13 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1173  
Tableau I. Résultats de l'algorithme cyclique pour les 16 000 simulations.  
(Le contenu du tableau est constitué de chiffres et d'acronymes déjà définis, donc pas de traduction textuelle nécessaire ici, juste la légende.)  
EQM, erreur quadratique moyenne ; CPU, unité centrale de traitement.

Tableau II. Résultats pour le scénario r = 3 et φ̂⁽⁰⁾ = (1/r, ..., 1/r)ᵀ.  
(Le contenu du tableau est constitué de chiffres et d'acronymes déjà définis.)  
AC, algorithme cyclique ; NR, méthode de Newton-Raphson ; BFGS, algorithme de Broyden-Fletcher-Goldfarb-Shanno ; NM, algorithme de Nelder-Mead ; EQM, erreur quadratique moyenne ; CPU, unité centrale de traitement.

d'itérations peut être observée. Pour une valeur fixe de n, 8000 valeurs de départ (correspondant à quatre schémas d'initialisation pour φ̂⁽⁰⁾, 2 valeurs de r et 1000 simulations) sont sélectionnées aléatoirement dans l'espace des paramètres, et l'EQM est à peu près la même même lorsque r varie de 3 à 5. On peut également remarquer que le temps de calcul ne varie pas trop (tous les temps CPU sont proches de 10⁻³) et cela renforce la suggestion que l'AC est robuste.

## *5.2.2. Comparaison avec d'autres algorithmes.* On peut voir que pour tous les scénarios, l'AC, Newton-Raphson et MM sont précis. Pour une petite valeur de n, l'algorithme de Nelder-Mead est aussi précis que l'AC sauf dans le scénario où r = 5, n = 5000 et φ̂⁽⁰⁾ est choisi aléatoirement. En général, l'algorithme BFGS a une EQM plus élevée que les autres méthodes. Par exemple, pour un schéma d'initialisation (I₂), n = 5000 et r = 3, l'ordre de l'EQM correspondant à BFGS est de 10⁻¹ tandis que les autres sont de 10⁻⁵. Ceci est également observé pour le schéma d'initialisation (I₃), n = 5000 et r = 5. En ce qui concerne la robustesse, on peut voir que l'AC et le NR ont presque le même nombre d'itérations (approximativement entre 3 et 5), ce qui est le plus bas. Les algorithmes MM, BFGS et NM utilisent approximativement trois à quatre fois plus d'itérations que l'AC. Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 14 :

1174  
A. N'GUESSAN ET I. C. GERALDO  
Tableau III. Résultats pour le scénario r = 3 et φ̂⁽⁰⁾ = (x₁ + x₂)/n.  
(Légende identique au Tableau II)

Tableau IV. Résultats pour le scénario r = 3 et φ̂⁽⁰⁾ choisi aléatoirement.  
(Légende identique au Tableau II)

## Plus important encore, l'algorithme AC est efficace. Il nécessite beaucoup moins de temps que les autres algorithmes. En effet, aucun des ratios de temps CPU n'est inférieur à 1, ce qui signifie qu'aucun des algorithmes ne nécessite moins de temps de calcul que l'AC. En moyenne et dans tous les cas, l'AC est cinq à six fois plus rapide que l'algorithme MM et, bien qu'ayant le même nombre d'itérations, l'AC est cinq à huit fois plus rapide que le NR. Ceci est compréhensible car à chaque itération de l'algorithme NR, une inversion de matrice est effectuée. L'AC est approximativement 174 (respectivement 554) à 512 (respectivement 2598) fois plus rapide que BFGS (respectivement NM). Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 15 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1175  
Tableau V. Résultats pour r = 3 et φ̂⁽⁰⁾ = x₁/Σ\_{m=1}^{r}x₁ₘ.  
(Légende identique au Tableau II)

## Tableau VI. Résultats pour le scénario r = 5 et φ̂⁽⁰⁾ = (1/r, ..., 1/r)ᵀ. (Légende identique au Tableau II) Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 16 :

1176  
A. N'GUESSAN ET I. C. GERALDO  
Tableau VII. Résultats pour le scénario r = 5 et φ̂⁽⁰⁾ = (x₁ + x₂)/n.  
(Légende identique au Tableau II)

## Tableau VIII. Résultats pour le scénario r = 5 et φ̂⁽⁰⁾ choisi aléatoirement. (Légende identique au Tableau II) Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 17 :

UN ALGORITHME CYCLIQUE UTILISANT LE COMPLÉMENT DE SCHUR  
1177  
Tableau IX. Résultats pour r = 5 et φ̂⁽⁰⁾ = x₁/Σ\_{m=1}^{r}x₁ₘ.  
(Légende identique au Tableau II)

6. CONCLUSION  
En statistique, lorsque l'estimation du maximum de vraisemblance avec ou sans contraintes est considérée, la méthode de Newton et/ou celle du score de Fisher viennent immédiatement à l'esprit. Si, de plus, l'expression complète de la fonction de vraisemblance et les contraintes sont disponibles, alors tout utilisateur utilisera immédiatement un logiciel commercial ou gratuit pour essayer de résoudre le problème auquel il est confronté.  
En théorie, tout progiciel dédié à l'optimisation sous contraintes permet de trouver des solutions. Le modèle statistique utilisé dans cet article n'échappe pas à cette règle à condition que le progiciel soit implémenté, c'est-à-dire, y avoir accès, donner des solutions initiales appropriées et être capable d'avoir la matrice Hessienne ou une approximation. Pour toutes ces raisons, nous proposons, sous certaines conditions de régularité, une méthode d'estimation qui généralise et complète les résultats de [16].  
Nous présentons quelques propriétés numériques de notre algorithme itératif cyclique pour l'estimation des paramètres d'un modèle multinomial. Ensuite, nous l'avons appliqué à la modélisation de l'effet moyen d'une mesure de sécurité routière sur le risque d'accident et les données d'accidents via une distribution multinomiale. Cet algorithme est très simple à programmer sans aucune inversion de matrice et il intègre facilement les contraintes d'inégalité ou d'égalité. Les résultats présentés dans cet article suggèrent que cet algorithme est robuste par rapport aux valeurs de départ, efficace et précis. De plus, la comparaison des performances de l'algorithme cyclique avec certains des meilleurs algorithmes d'optimisation disponibles suggère qu'il est aussi précis que les autres et, plus important encore, qu'il est beaucoup plus rapide en ce qui concerne la convergence.

## ANNEXE A. PREUVES A.1. Détails de la preuve du Théorème 1 D\_{θ,x} existe par hypothèse et ∂h/∂φᵢ ≠ 0. Par conséquent, l'inversion de la matrice Ω\_{θ,x} du Théorème 1 est possible car le complément de Schur de D\_{θ,x} dans Ω\_{θ,x} noté (Ω\_{θ,x}|D\_{θ,x}) existe et son inversion Copyright © 2015 John Wiley & Sons, Ltd. Numer. Linear Algebra Appl. 2015; 22:1161-1179 DOI: 10.1002/nla

Page 18 :

1178  
A. N'GUESSAN ET I. C. GERALDO  
est possible avec (Ω\_{θ,x}|D\_{θ,x}) = (∇φh)ᵀD\_{θ,x}⁻¹(∇φh) ≠ 0. Nous savons alors que (par exemple, [17, 18, 25])  
Ω\_{θ,x}⁻¹ = [ M\_{θ,x} −D\_{θ,x}⁻¹(∇φh)((Ω\_{θ,x}|D\_{θ,x}))⁻¹ ]  
[ −((Ω\_{θ,x}|D\_{θ,x}))⁻¹(∇φh)ᵀD\_{θ,x}⁻¹ ((Ω\_{θ,x}|D\_{θ,x}))⁻¹ ]  
En multipliant Ω\_{θ,x}⁻¹ par (Bᵀx, η)ᵀ, et en utilisant η = (∇φh)ᵀφ̂, nous montrons que  
0 = ||(∇φh)ᵀ||^{-2}\_{D\_{θ,x}⁻¹}(∇φh)ᵀD\_{θ,x}⁻¹Bx − ||(∇φh)ᵀ||^{-2}\_{D\_{θ,x}⁻¹}η  
φ̂ = D\_{θ,x}⁻¹Bx.

A.2. Détails de la preuve du Corollaire 1  
Posons  
Σ\_θ = [ Λ\_{θ,Z} √θ Bx ]  
[ √θ Zᵀ 1 ]  
Comme Λ\_{θ,Z}⁻¹ existe, alors le complément de Schur de Λ\_{θ,Z} dans Σ\_θ existe ainsi que le complément de Schur de 1 dans Σ\_θ. Nous avons alors D\_{θ,x} = (Σ\_θ|1) = Λ\_{θ,Z} − θBxZᵀ et  
(Σ\_θ|1)⁻¹ = Λ\_{θ,Z}⁻¹ + Λ\_{θ,Z}⁻¹θ^{1/2}Bx(Σ\_θ|Λ\_{θ,Z})⁻¹θ^{1/2}ZᵀΛ\_{θ,Z}⁻¹  
où (Σ\_θ|Λ\_{θ,Z})⁻¹ = (1 − θZᵀΛ\_{θ,Z}⁻¹Bx)⁻¹ > 0. Nous en déduisons alors que  
φ̂ = D\_{θ,x}⁻¹Bx = (1 − θ(Z, Bx)\_{Λ\_{θ,Z}⁻¹})⁻¹Λ\_{θ,Z}⁻¹Bx.

REMERCIEMENTS  
Nous remercions l'éditeur et les relecteurs anonymes pour leurs remarques et suggestions, qui ont permis une amélioration substantielle de cet article. L'article a été partiellement écrit alors que le premier auteur visitait le CIRRELT de l'Université de Montréal, Canada. Le second auteur a été partiellement soutenu par le "Groupe d'Intérêt Economique" (GIE) de PSA-RENAULT de France dans le cadre de l'accord N° USTL-GIE PSA-RENAULT 248 02 515.

## RÉFÉRENCES (La liste des références reste en anglais car ce sont des titres d'ouvrages ou d'articles et des noms propres.)

Page 19 :  
(La liste des références continue et reste en anglais.)

J'ai fait de mon mieux pour que la traduction soit fidèle et "efficace" pour un contexte académique, tout en respectant scrupuleusement votre consigne de ne pas toucher aux formules.