La Distribution Enroulée de la Sécante Hyperbolique et ses Mélanges Binaires *

Simon L. Marshall
Air Quality Services
Department of Water and Environmental Regulation
Perth, WA 6000
Australia†

Résumé

Les données circulaires bimodales sont souvent analysées par la formation de mélanges binaires de distributions von Mises, ou moyennant la distribution von Mises généralisée. Bien que la dernière fonction ait l'avantage d'être de la classe exponentielle, la fonction de répartition n'est pas facile à calculer pour l'une ou l'autre de ces distributions, puisqu'elle contient des séries infinies de fonctions de Bessel modifiées. Dans cet article, il est montré que la distribution enroulée de la sécante hyperbolique montre un comportement qualitatif similaire à la distribution von Mises, mais avec l'avantage supplémentaire d'être analytiquement intégrable. Nous dérivons des représentations en séries à convergence rapide pour la fonction de la densité de probabilité, d'où viennent des formules simples pour les moments circulaires d'ordre arbitraire, et nous décrivons des processus robustes et efficaces pour l'application de la méthode des moments à la détermination des paramètres dans les mélanges binaires de telles distributions. L'application de ces distributions composées à la représentation de données bimodales de la direction du vent est démontrée.

*À soumettre à la *Revue Canadienne de la Statistique*.
†Present address: 24 Flinders Crescent, Bull Creek, Western Australia

1 Introduction

La distribution de la sécante hyperbolique (sech) n'est pas très bien connue, mais elle a en réalité une longue histoire. Outre son rôle significatif dans la théorie de l'échantillonnage des coefficients de corrélation, comme le montre le travail de Fisher [1921] (voir également Smyth [1994]), elle a connu relativement peu d'applications en soi, mais dans un éventail remarquablement divers de contextes pratiques. Perks [1932] l'a utilisée dans l'analyse des données de mortalité pour des applications actuarielles, et Talacko [1956, 1958] a examiné son rôle dans la théorie des variables stochastiques de Wiener. Plus récemment, Ding [2014] a fourni une revue intéressante de ses applications aux corrélations et à l'analyse des tableaux de contingence, et Losev [1989] a décrit son utilisation dans la représentation empirique de profils de pics en spectroscopie photoélectronique de rayons X. La courbe de densité de probabilité de la distribution de la sécante hyperbolique présente un pic symétrique de forme qualitativement similaire à celle des distributions normale et logistique, mais avec des "queues" proportionnellement plus lourdes. Du point de vue computationnel, ses caractéristiques attrayantes sont que tous ses moments existent et que la fonction de probabilité cumulative et son inverse sont tous deux exprimables sous une forme fermée.

Les distributions définies sur la droite réelle peuvent être adaptées à la description de données circulaires de diverses manières, dont la plus connue est la procédure de "enroulement". Pour satisfaire la condition de bord périodique (égalité des valeurs à 0 et 2π), la fonction unique représentant la distribution linéaire est remplacée par une série infinie de copies de la fonction dans laquelle l'argument est déplacé d'un multiple entier de 2π [Mardia et Jupp, 2000, pages 49-52]. Dans les cas favorables (notamment celui de la distribution de Cauchy enroulée), la série infinie résultante peut être sommée sous forme fermée. Il convient de noter en passant que la rapidité de convergence de telles expressions n'est nullement garantie; diverses méthodes d'accélération numérique et analytique de la convergence pourraient être requises. Nadarajah et Zhang [2017] ont récemment développé un package R pour le calcul de plusieurs dizaines de distributions enroulées, mais aucune de celles-ci ne correspond à la distribution linéaire sech. Le présent article a pour triple objectif

1. dériver des expressions utiles sur le plan computationnel pour la distribution sech enroulée;

2. décrire des méthodes efficaces pour la simulation et la paramétrisation de distributions sech enroulées simples et de leurs mélanges binaires; et

3. démontrer l'utilité des modèles de mélange dans la représentation de données bimodales.

La performance de ces modèles de mélange sont illustrées non seulement en référence à des données synthétiques, mais également à des mesures de direction du vent.

2 La Distribution sech

Afin de fournir une base pour l'interprétation des propriétés de la distribution sech enroulée, il peut être utile de nous rappeler quelques propriétés pertinentes de sa contrepartie linéaire. La forme canonique de la distribution sech avec support $x \in (-\infty, \infty)$ est

$$g(x) = \frac{1}{2}\text{sech}\left(\frac{\pi}{2}x\right), \qquad G(x) = \frac{2}{\pi}\tan^{-1}\left[e^{\pi x/2}\right]. \tag{2-1}$$

La courbe de la densité de probabilité a un kurtosis excès (coefficient d'aplatissement) de l'unité, ce qui indique un pic plus étroite et queues plus lourdes que la distribution normale, pour laquelle cette paramètre est nulle. Pour nos applications cette équation peut être trivialement généralisée à la forme à deux paramètres

$$g(x) = \frac{a}{\pi}\text{sech}\left[a(x-b)\right] \tag{2-2}$$

dans laquelle a et b peuvent être appelés respectivement paramètre d'échelle et paramètre d'emplacement. De toute évidence, le maximum de la densité de probabilité se produit à $x = b$, et la largeur et l'étroitesse du pic résultent, respectivement, de petites et grandes valeurs de a, comme illustré par la figure 1. L'usage du résultat

$$2\int_{-\infty}^{\infty}\frac{dx}{e^{a(x-b)}+e^{-a(x-b)}} = \frac{2}{a}\int_0^{\infty}\frac{du}{u^2+1} = \frac{\pi}{a} \tag{2-3}$$

dans l'intégration de l'équation 2-2 confirme que la distribution est correctement normalisée. La fonction de distribution cumulative est

$$p = \frac{2a}{\pi}\int_{-\infty}^{X} f(x)dx = \frac{2}{\pi}\int_0^{e^{a(X-b)}}\frac{du}{u^2+1} = \frac{2}{\pi}\tan^{-1}\left[e^{a(X-b)}\right], \tag{2-4}$$

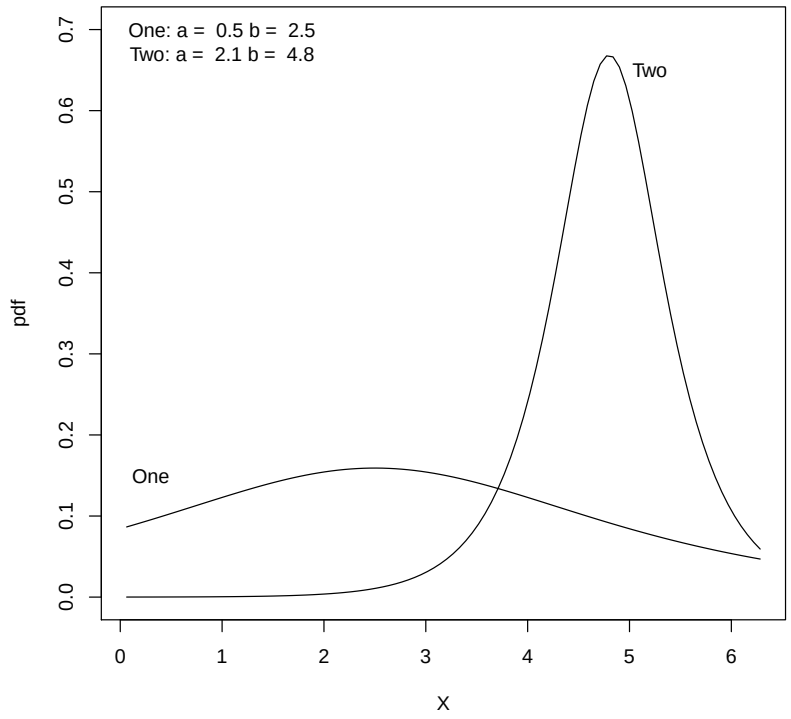

Figure 1: Fonctions de la densité de probabilité de l'équation 2-2.

à partir de laquelle l'abscisse correspondant à une valeur spécifiée de p s'obtient facilement :

$$X = b + \frac{1}{a} \ln \left[\tan \frac{\pi p}{2} \right]. \tag{2-5}$$

3 La Distribution sech Enroulée

Cette section a pour but la dérivation des formules efficaces sur le plan computationel pour la distribution sech enroulée et ses moments circulaires, moyennant la Formule Sommatoire de Poisson (FSP).

3.1 Formulation

La distribution enroulée $f(x)$, formée à partir d'une densité de probabilité normalisée $g(x)$ definie sur la droite réelle, est [Mardia et Jupp, 2000, page 48]

$$f(x) = \sum_{k=-\infty}^{\infty} g(x + 2\pi k) \approx \sum_{k=-K}^{K} g(x + 2\pi k), \tag{3-1}$$

dans laquelle l'indice limite de la sommation K est choisie selon une précision spécifiée. Dans le package R `wrapped` developpé par Nadarajah et Zhang [2017], K est spécifié comme paramètre d'entrée, mais aucun critère ne semble être donné pour attribuer sa valeur. L'utilisation de la PSF pour établir un tel critère pour la distribution sech enroulée sera démontrée dans les sections suivantes.

3.2 Accélération de la Convergence

Au delà de son importance dans l'analyse harmonique, la FSP s'avère d'une grande valeur pour accélérer des séries à convergence lente - en particulier des solutions en série de Fourier des équations aux dérivées partielles [Marshall, 1998a,b], et des fonctions de Green [Marshall, 1999, 2000, 2002]. Dans le contexte du problème d'enroulment, l'application de la FSP convertit une série infinie de la forme 3-1 en une série infinie de transformées de Fourier. Ainsi, pour une fonction convenablement intégrable $h(x)$, la formule de Poisson est

$$\sum_{n=-\infty}^{\infty} h(n) = \sum_{N=-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} e^{2N\pi it} h(t) dt \right\} \tag{3-2}$$

Avec $h(n) = g(x+2n\pi)$, les transformées de Fourier requises dans le problème d'enroulement peuvent être transformées par le changement de variable $u = x + 2t\pi$, $dt = du/2\pi$,

$$\int_{-\infty}^{\infty} e^{2N\pi it} g(x + 2t\pi) dt = \frac{e^{-iNx}}{2\pi} \int_{-\infty}^{\infty} e^{iNu} g(u) du, \tag{3-3}$$

de sorte que la forme pertinente du FSP soit

$$\sum_{n=-\infty}^{\infty} g(x + 2\pi n) = \frac{1}{2\pi} \sum_{N=-\infty}^{\infty} e^{-iNx} \left\{ \int_{-\infty}^{\infty} e^{iNu} g(u) du \right\}. \tag{3-4}$$

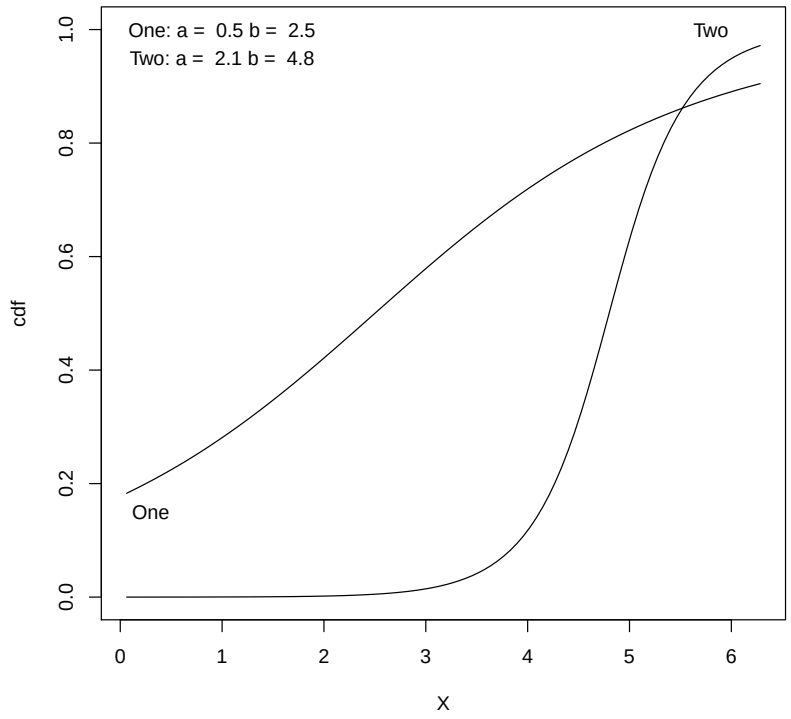

Figure 2: Fonctions de répartition calculées de l'équation 2-2.

Une observation cruciale qui se peut faire concernant l'équation 3-4 est que si g diminue lentement avec l'augmentation de n, sa transformée de Fourier diminuera rapidement avec l'augmentation de N, et *vice versa*. Dépendant des valeurs des autres paramètres, l'usage de la FSP d'une telle manière peut produire une accélération significative de la convergence, et au cas favorables, la somme des transformées de Fourier peut être évaluer sous forme fermée. Le résultat le plus connu que l'on puisse ainsi obtenir est donc la distribution enroulée de Cauchy,

$$\frac{\sigma}{\pi}\sum_{n=-\infty}^{\infty}\frac{1}{\sigma^2+(x-\mu+2n\pi)^2}=\frac{1}{2\pi}\frac{\sinh\sigma}{\cosh\sigma-\cos(x-\mu)}. \tag{3-5}$$

dans laquelle les intégrales de Fourier sont des fonctions exponentielles qui peuvent être sommées comme séries géometriques. Bien que cette méthode ne se puisse pas appliquer à l'enroulement de la distribution sech, Marshall [1998b] a appliqué la FSP à la série très similaire

$$\sum_{n=-\infty}^{\infty} \frac{\cos cn}{\cosh an}, \tag{3-6}$$

qui converge très lentement pour des petites valeurs de a, pour fournir une forme équivalente à convergence rapide qui est presque aussi bonne qu'un résultat de forme fermée. Ainsi,

$$\sum_{n=-\infty}^{\infty} \frac{\cos cn}{\cosh an} = \sum_{N=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{2N\pi it} \frac{\cos ct}{\cosh at} dt. \tag{3-7}$$

L'intégrale qui paraît ici se trouve dans la collection de transformées Fourier à cosinus de Erdélyi et al. [1954, page 30] sous la forme

$$\int_0^{\infty} \frac{\cos xy}{\cosh ax} dx = \frac{\pi}{2a} \frac{1}{\cosh(\pi y/2a)} \tag{3-8}$$

qui est valide pour $y > 0$; ici, $y = 2N\pi \pm c$. La somme désirée est

$$\sum_{n=-\infty}^{\infty} \frac{\cos cn}{\cosh an} = \frac{\pi}{a} \frac{1}{\cosh(\pi c/2a)} \tag{3-9}$$

$$+\frac{\pi}{2a} \sum_{N=1}^{\infty} \left[\frac{1}{\cosh[(2N\pi + c)\pi/2a]} + \frac{1}{\cosh[(2N\pi - c)\pi/2a]} \right].$$

Si $a \ll \pi/2$, tous les termes avec $N \geq 1$ seront complètement négligeables. Dans le cas $c = 0$, similaire à notre série, cela se réduit à

$$\sum_{n=-\infty}^{\infty} \frac{1}{\cosh an} = \frac{\pi}{a} \sum_{N=-\infty}^{\infty} \frac{1}{\cosh(N\pi^2/a)}. \tag{3-10}$$

De cette forme, il est plus clair que pour $a \ll \pi/2$, la somme est trivialement différente de l'intégrale sur $(-\infty, \infty)$, qui selon l'équation 2-3 est π/a.

3.3 Moments Circulaires

Or, pour enrouler cette distribution, la somme à évaluer est

$$\sum_{n=-\infty}^{\infty} g(x+2n\pi) = \frac{a}{\pi} \sum_{n=-\infty}^{\infty} \frac{1}{\cosh[a(x-b+2n\pi)]} \tag{3-11}$$

ce qui à son tour a besoin de

$$\int_{-\infty}^{\infty} e^{2N\pi it} \frac{1}{\cosh[a(x-b+2t\pi)]} dt = \frac{e^{-iN(x-b)}}{2\pi} \int_{-\infty}^{\infty} \frac{\cos Nu}{\cosh au} du. \tag{3-12}$$

Application de la FSP donc produit

$$\sum_{n=-\infty}^{\infty} \frac{1}{\cosh[a(x-b+2n\pi)]} = \sum_{N=-\infty}^{\infty} \frac{e^{-iN(x-b)}}{2\pi} \int_{-\infty}^{\infty} \frac{\cos Nu}{\cosh au} du$$

$$= \frac{1}{2\pi} \left[\int_{-\infty}^{\infty} \frac{du}{\cosh au} + 2 \sum_{N=1}^{\infty} \cos N(x-b) \int_{-\infty}^{\infty} \frac{\cos Nu}{\cosh au} du \right]. \tag{3-13}$$

Utilisant l'équation 3-8 avec $y = N$, la distribution sech enroulée est

$$f(x|a,b) = \frac{a}{\pi} \sum_{n=-\infty}^{\infty} \frac{1}{\cosh[a(x+2n\pi)]} = \frac{a}{\pi} \left[\frac{1}{2a} + \frac{1}{a} \sum_{N=1}^{\infty} \frac{\cos N(x-b)}{\cosh(N\pi/2a)} \right] \tag{3-14}$$

Il ressort peut-être plus clairment de la deuxième forme de l'équation 3-14 que $f(x|a,b)$ est normalisée, car après intégration sur $[0, 2\pi]$ tous les termes avec $N > 0$ disparaissent par orthogonalité,

$$\int_0^{2\pi} \cos Nx \cos Mx dx = \pi \delta_{NM}, \tag{3-15}$$

et la constante donne π/a. Une conséquence supplémentaire de l'équation 3-15 et

$$\int_0^{2\pi} \sin Nx \sin Mx dx = \pi \delta_{NM}, \qquad \int_0^{2\pi} \sin Nx \cos Mx dx = 0 \tag{3-16}$$

est que les moments trigonométriques de l'ordre M sont

$$\int_0^{2\pi} f(x|a,b) \cos Mx = \frac{\cos Mb}{\cosh(M\pi/2a)}, \tag{3-17}$$

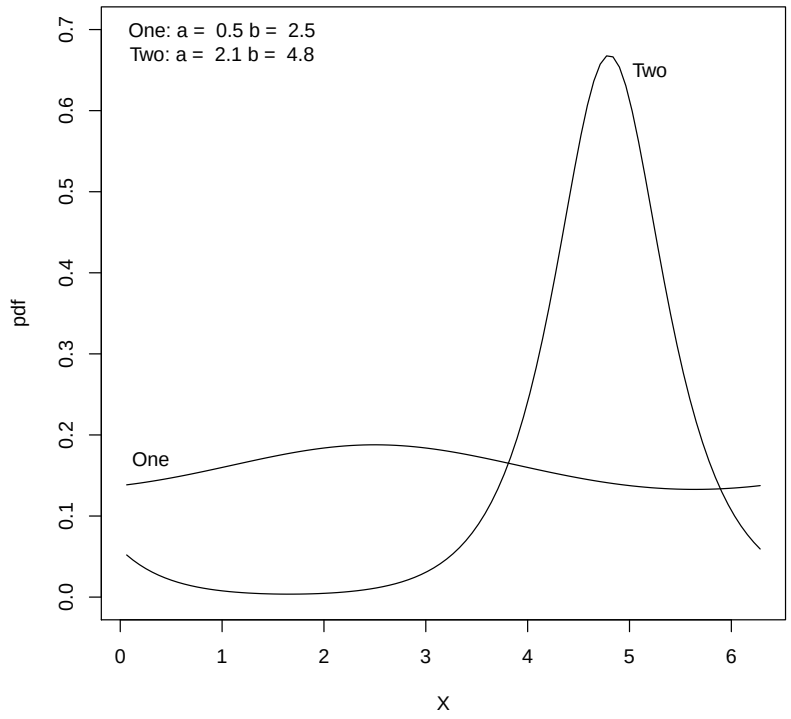

Figure 3: Densités de probabilité calculées de l'équation 3-14, avec les paramètres adoptés dans les Figures 1 et 2.

$$\int_0^{2\pi} f(x|a,b) \sin Mx = \frac{\sin Mb}{\cosh(M\pi/2a)}. \tag{3-18}$$

Les contreparties enroulées des distributions sech de la Figure 1 sont illustrées à la Figure 3. La fonction de répartition corréspondante à l'équation 3-14 est

$$F(x|a,b) = \frac{2}{\pi} \sum_{n=-\infty}^{\infty} \left[\tan^{-1}\left(e^{a(2n\pi+x-b)}\right) - \tan^{-1}\left(e^{a(2n\pi-b)}\right) \right] \tag{3-19}$$

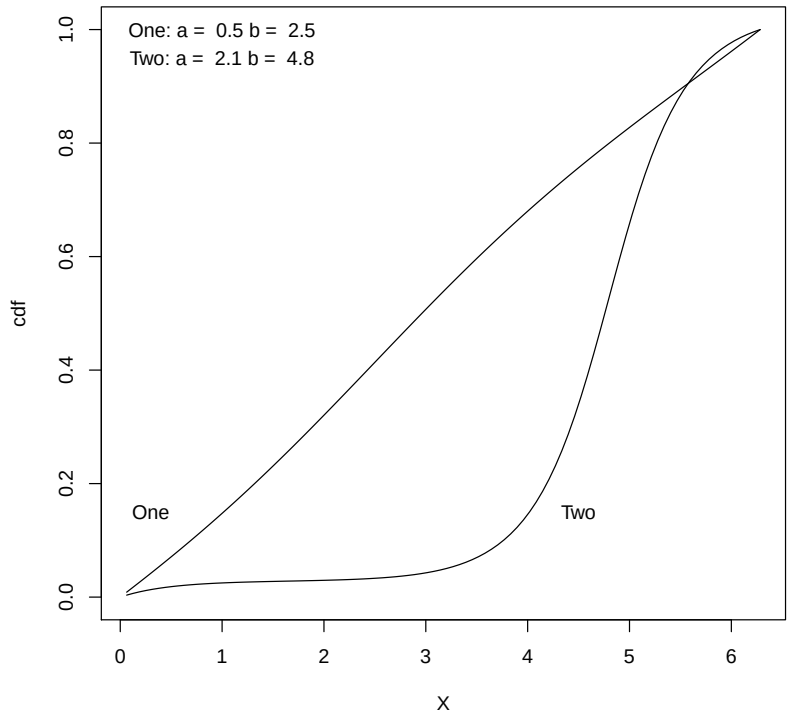

Figure 4: Fonctions de répartition calculées de l'équation 3-19, avec les paramètres adoptés dans les Figures 1 et 2.

qui résulte de la série originale, tandis que de la série transformée on obient

$$F(x|a,b) = \frac{x}{2\pi} + \frac{1}{\pi}\sum_{N=1}^{\infty}\frac{\sin N(x-b)}{N\cosh(N\pi/2a)}; \tag{3-20}$$

le premier terme à droite représente la contribution de la distribution uniforme, à laquelle la fonction se réduit comme $a \to 0$ (voir Figure 4.)

3.4 Calculs Numériques

L'équivalence des deux séries pour la densité de probabilité peut être vérifiée par calcul numérique. Ainsi, pour a=0.25, b=2.9 et x=1.49782, la sortie d'un

script R calculant l'index, le terme principal et le total cumulé des sommes est la suivante :

```
0 2 2
1 0.002507122 2.002507
2 -2.632731e-05 2.002481
3 -2.524473e-08 2.002481
4 7.598939e-11 2.002481
5 1.356658e-13 2.002481
6 -1.79968e-16 2.002481
0.1593524
Using original series:
0 0.9415552 0.9415552
1 0.8296825 1.771238
2 0.1830687 1.954306
3 0.03815897 1.992465
4 0.007933398 2.000399
.......
20 9.648297e-14 2.002481
21 2.005684e-14 2.002481
22 4.169407e-15 2.002481
23 8.667346e-16 2.002481
0.1593524
```

tandis qu'avec a=1/0.25=4, et les mêmes valeurs des autres quantités, la sortie abbreviée est

```
0 0.125 0.125
1 0.03891525 0.1639152
2 -0.1781042 -0.01418898
3 -0.06813037 -0.08231936
4 0.07781839 -0.004500963
.......
87 -6.261304e-16 0.007331425
88 -3.165292e-16 0.007331425
89 2.137408e-16 0.007331425
90 1.927584e-16 0.007331425
0.00933466
Using original series:
0 0.007331419 0.007331419
1 6.635309e-09 0.007331425
2 8.069569e-20 0.007331425
0.00933466
```

Comparaison de ces deux sorties démontre l'observation faite après l'équation 3-4 concernant l'influence des paramètres sur la vitesse de convergence - dans ce cas, rélative au paramètre d'échelle a.

La série originale et la série transformée de l'équation 3-14 se peuvent utiliser pour le développement d'une méthode efficace d'évaluation. Car, supposons que la tolérance de sommation de la série originale est ϵ. Le dernier terme sera inférieur à celui-ci pour les valeurs de n telles que

$$\cosh[a(x - 2n\pi)] = \cosh[a(2n\pi - x)] > \frac{1}{\epsilon} \tag{3-21}$$

Ceci équivaut à exiger que

$$n > \frac{1}{2\pi}\left[x + \frac{1}{a}\cosh^{-1}\left(\frac{1}{\epsilon}\right)\right]. \tag{3-22}$$

Pour la série transformée, la condition correspondante est que

$$\cosh\frac{\pi N}{2a} > \frac{1}{\epsilon} \tag{3-23}$$

qui équivaut à

$$N > \frac{2a}{\pi}\cosh^{-1}\left(\frac{1}{\epsilon}\right). \tag{3-24}$$

Ainsi, pour valeurs spécifiées de a, b, x, la fonction peut être évalué par la série avec moins de termes, $\min(n, N)$. Par exemple, avec ϵ=1×10^{-15}, $x = 2.3479$, $a = 0.5$, et $b = 0$, ce processus conduit à la choix de la série transformée. L'indice, dernier terme et total cumulatif sont

```
1 -0.1209835 0.8790165
2 -0.0001239098 0.8788926
3 0.0002338612 0.8791265
4 -1.394169e-05 0.8791125
5 4.082098e-07 0.8791129
6 1.295876e-09 0.8791129
7 -8.408447e-10 0.8791129
8 4.853917e-11 0.8791129
9 -1.371473e-12 0.8791129
10 -7.526413e-15 0.8791129
11 3.017283e-15 0.8791129
12 -1.688064e-16 0.8791129
0.1399152
```

(le dernier nombre est la somme multipliée par le facteur de normalisation a/π.) Pour a=1.5 et les mêmes valeurs des autres paramètres, on choisit la série originale, et les valeurs des termes et la somme sont

```
1 0.005467599 0.06450552
2 4.412359e-07 0.06450596
3 3.560752e-11 0.06450596
4 2.87351e-15 0.06450596
5 2.318909e-19 0.06450596
0.03079933
```

Bien que l'équivalence des deux séries soit gratifiante, il faut également admettre qu'en pratique, il est peu probable que l'on rencontre des distributions avec de petites valeurs de a. L'avantage le plus important de la série de Fourier est de fournir des expressions simples pour les moments trigonométriques, qui, comme on le verra, sont d'une grande valeur dans la paramétrisation des modèles de mélange.

3.5 Simulation

Alors que des méthodes numériques spécialisées ont été développées pour la génération de variables aléatoires suivant les distributions circulaires les plus connues - telles que la distribution de von Mises - aucune technique de ce type n'existe pour la distribution sech et sa contrepartie enveloppée. Des collections de telles variables aléatoires peuvent être générées de manière pratique par la méthode de transformation [Press et al., 1992, pages 277–280], dans laquelle les nombres sont obtenus comme les abscisses des points auxquels la distribution cumulative est égale aux nombres aléatoires échantillonnés de la distribution uniforme $u(0, 1)$. Étant donné que ce processus nécessite la solution itérative d'une équation non linéaire - pour laquelle la méthode de Newton-Raphson est particulièrement bien adaptée, parce que la dérivée requise est simplement la densité de probabilité - l'efficacité de calcul est potentiellement un problème. Dans cette optique, l'algorithme suivant a été développé:

1. Pour les paramètres spécifiés a, b, générer un tableau des valeurs de la fonction de densité de probabilité et de son intégrale, pour des abscisses rapprochées (100 points à des intervalles de $2\pi/100$ suffisent.)

2. Générer un nombre aléatoire y à partir de $u(0,1)$ (en utilisant la commande R intégrée ou l'une des nombreuses méthodes disponibles.)

3. Trouver un triplet entre parenthèses d'abscisses consécutives (x_1, x_2, x_3), de sorte que $y \in [F(x_1|a,b)), F(x_3|a,b)]$.

4. Former la parabole lagrangienne passant par les points $(x_1, F(x_1|a,b))$, $(x_2, F(x_2|a,b))$ et $(x_3, F(x_3|a),b))$, *viz.*,

$$p(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)((x_1-x_3)}F(x_1|a,b)$$

$$+\frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3))}F(x_2|a,b) + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)((x_3-x_1)}F(x_3|a,b)$$

   et trouvez la valeur de $x \in [x_1, x_3]$ à laquelle elle prend la valeur désirée en résolvant l'équation quadratique $p(x) = y$.

5. Initier l'itération Newton - Raphson à partir du point (x, y).

Expérience avec cet algorithme montre que convergence à precision double est réalisée en 3 ou 4 itérations, et que des milliers de points peuvent être générés en moins d'une seconde.

3.6 Paramétrisation de Maximum-Vraisemblance

La génération de variables aléatoires de cette manière fournit une méthode pour vérifier les performances de diverses méthodes d'optimisation pour la représentation des données. Cela implique généralement la minimisation de la fonction de log-vraisemblance négative

$$\mathcal{L} \equiv -\frac{1}{n}\sum_{k=1}^{n} \ln f(x|a,b) \tag{3-25}$$

par rapport à a, b. Étant donné que la densité de probabilité n'est pas représentée sous forme fermée ni par la série originale ni par la série transformée, la formation analytique des équations du maximum de vraisemblance $\nabla\mathcal{L} = \mathbf{0}$ devient plutôt lourde. Le minimum de $\mathcal{L}$ peut cependant être obtenue par la méthode du simplexe de Nelder-Mead [Press et al., 1992, pages 402–406], qui s'appuie uniquement sur les valeurs de la fonction objective et n'utilise pas de dérivées. Comme l'espace des paramètres n'a que

deux dimensions, celui-ci atteint assez rapidement le minimum. Le point de départ requis à cet effet peut être obtenu en trouvant a et b de sorte que les moments sinus et cosinus donnés par les équations 3-18 et 3-17 d'ordre N=1 concordent avec ceux déterminés à partir des données , qui sont

$$S_1 = \frac{1}{n}\sum_{k=1}^{n} \sin\theta_k, \qquad C_1 = \frac{1}{n}\sum_{k=1}^{n} \cos\theta_k. \tag{3-26}$$

Ainsi on obtient pour b,

$$\tan b = \frac{S_1}{C_1} \longrightarrow b = \tan^{-1}\frac{S_1}{C_1} \tag{3-27}$$

et pour a

$$S_1^2 + C_1^2 = \frac{1}{\cosh^2(\pi/2a)} \longrightarrow \frac{\pi}{2a} = \cosh^{-1}\left(\frac{1}{\sqrt{S_1^2 + C_1^2}}\right) \tag{3-28}$$

ou

$$a = \frac{\pi/2}{\operatorname{arccosh}\left(1/\sqrt{S_1^2 + C_1^2}\right)}.$$

La figure 5 montre un histogramme d'un ensemble de données contenant 10000 points aléatoires calculés comme décrit ci-dessus de la distribution 3-14 avec a=2.1 et b=4.8. Pour cet ensemble de données les deux moments requis sont

$$S_1 = -0.7698032, \qquad C_1 = 0.07436659$$

d'où

$$a = 2.100089, \qquad b = -1.47449;$$

quand cette valeur de b est ajoutée à 2π, le résultat est 4.808695. À partir de ce point, l'optimiseur Nelder-Mead (avec tolérance fonctionelle égale à 10^{-7}) converge après 20 itérations vers les estimations du maximum de vraisemblance (a, b) = (2.103027, 4.80757), avec lesquelles la courbe de la Figure 5 à été calculée. L'accord excellent avec les données est à noter.

4 Distribution sech Enroulée Mixte

4.1 Formulation

La distribution sech enroulée peut donner une bonne représentation des données unimodales, mais ne convient pas aux données bimodales telles que

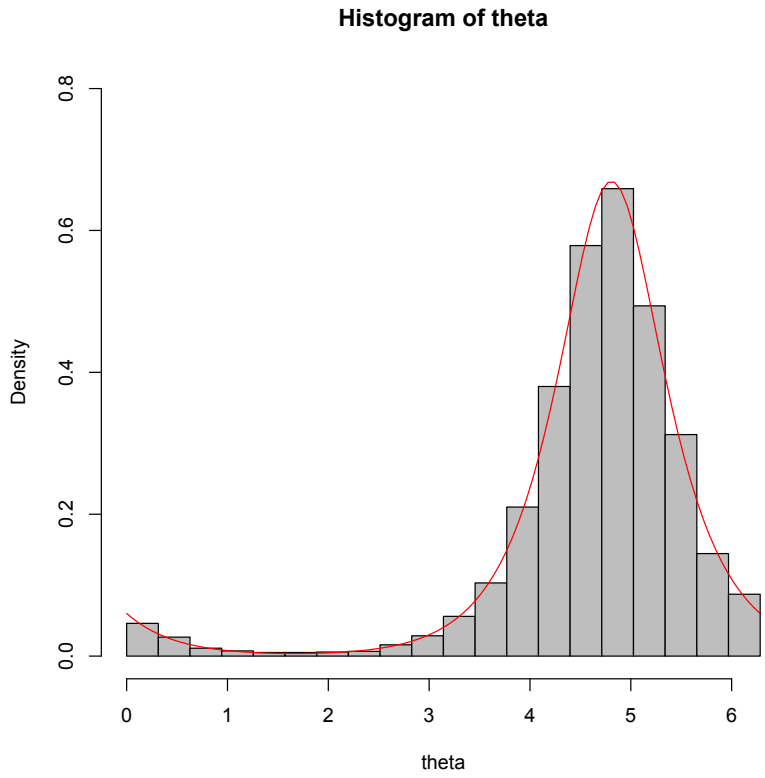

Figure 5: Histogramme de 10000 nombres aléatoires générés de l'équation 3-14 avec a=2.1 et b=4.8. Courbe: calculée de l'équation 3-14 avec estimations du maximum de vraisemblance (a=2.103027, b=4.80757).

les distributions des directions du vent. Une méthode pour surmonter cette limitation est la construction de modèles de mélange, dans lesquels la densité de probabilité est postulée comme étant une combinaison linéaire de distributions unimodales avec des coefficients contraints à s'ajouter à 1. Combinaisons de ce type formées à partir de la distribution de von Mises

$$f(\theta|\kappa,\mu) = \frac{1}{2\pi I_0(\kappa)} \exp[\kappa \cos(\theta - \mu)], \tag{4-1}$$

où I_0 est la fonction de Bessel modifiée de la première espèce d'ordre zéro, sont peut-être les plus connues. Mélanges de distributions von Mises ont

été appliquées à l'analyse de données de la direction du vent [Razali et al., 2012; Belu et Koracin, 2013; Masseran et al., 2015]; plusieurs contributions publiées ont montré que deux distributions von Mises suffisent. Un package R pour la paramétrisation de données multimodales a été publié [Hornik et Grün, 2014].

La paramétrisation de distributions mixtes, formées soit des distributions von Mises seules, soit comme combinaison d'une distribution von Mises avec une distribution uniforme[Bentley, 2006], est plus difficile que la paramétrisation d'une seule distribution von Mises, parce que la combinaison additive n'est plus de la classe exponentielle. La conséquence pratique de cela est que la formation de la fonction log-vraisemblance ne produit pas des sommes des données (comme les sommes trigonométriques pour une seule distribution von Mises, ou sommes de carrés pour les distributions normales) qui se doivent évaluer une seule fois. Plutôt, sommation sur tous les points est nécessaire pour chaque point dans l'espace des paramètres visité par l'optimiseur.

La densité de probabilité pour un mélange de distributions sech aux paramètres a_1, b_1 and a_2, b_2 peut être représentée sous la forme

$$\phi(x|a_1, b_1, a_2, b_2, Y) =$$

$$f(x|a_1, b_1)(1 + \tanh Y)/2 + f(x|a_2, b_2)(1 - \tanh Y)/2, \qquad (4\text{-}2)$$

où Y peut être positif ou négatif. L'utilisation de la fonction tangente hyperbolique assure que la contribution fractionnaire de chaque distribution est comprise entre 0 et 1. Les observations faites précédemment sur les difficultés de paramétrisation des distributions sech enroulées s'appliquent *a fortiori* aux mélanges, où 5 paramètres sont à déterminer; l'évaluation de la fonction de log-vraisemblance sur des points dans un espace à 5 dimensions rend l'application de la méthode de Nelder-Mead moins attrayante. Le fait que les équations du premier moment fournissent une excellente approximation des estimations du maximum de vraisemblance pour la distribution sech enroulée simple suggère qu'une alternative plus efficace pour la distribution mixte consiste à trouver les paramètres pour lesquels les moments trigonométriques sont en accord avec ceux déterminés à partir des données, comme indiqué dans les paragraphes suivants.

4.2 Solution Moindres-Carrés des Équations des Moments

Les premiers moments du sinus et cosinus pour un ensemble de données circulaires $\theta_1, \theta_2, ..., \theta_n$ sont donnés par l'équation 3-26, et les deuxième et troisième moments sont definis de la même manière comme

$$S_2 = \frac{1}{n}\sum_{k=1}^{n} \sin 2\theta_k, \qquad C_2 = \frac{1}{n}\sum_{k=1}^{n} \cos 2\theta_k, \tag{4-3}$$

et

$$S_3 = \frac{1}{n}\sum_{k=1}^{n} \sin 3\theta_k, \qquad C_3 = \frac{1}{n}\sum_{k=1}^{n} \cos 3\theta_k. \tag{4-4}$$

L'idée est de trouver les valeurs des paramètres pour lesquelles les moments de la distribution du mélange concordent avec ceux calculés à partir des données selon ces définitions. Puisqu'il y a cinq paramètres et six équations, il y a évidemment deux façons d'y parvenir. Les quatre premières équations à résoudre sont

$$g_1 \equiv X\frac{\cos b_1}{\cosh(\pi/2a_1)} + (1-X)\frac{\cos b_2}{\cosh(\pi/2a_2)} - C_1 = 0 \tag{4-5}$$

$$g_2 \equiv X\frac{\sin b_1}{\cosh(\pi/2a_1)} + (1-X)\frac{\sin b_2}{\cosh(\pi/2a_2)} - S_1 = 0$$

$$g_3 \equiv X\frac{\cos 2b_1}{\cosh(\pi/a_1)} + (1-X)\frac{\cos 2b_2}{\cosh(\pi/a_2)} - C_2 = 0$$

$$g_4 \equiv X\frac{\sin 2b_1}{\cosh(\pi/a_1)} + (1-X)\frac{\sin 2b_2}{\cosh(\pi/a_2)} - S_2 = 0.$$

Pour déterminer le cinquième paramètre $Y = \tanh^{-1}(2X-1)$, on peut utiliser soit le troisième moment cosinus, soit le troisième moment sinus. Avec le premier choix, la cinquième équation à satisfaire est

$$g_5 \equiv X\frac{\cos 3b_1}{\cosh(3\pi/2a_1)} + (1-X)\frac{\cos 3b_2}{\cosh(3\pi/2a_2)} - C_3 = 0, \tag{4-6}$$

tandis qu'avec le deuxième

$$g_5 \equiv X\frac{\sin 3b_1}{\cosh(3\pi/2a_1)} + (1-X)\frac{\sin 3b_2}{\cosh(3\pi/2a_2)} - S_3 = 0. \tag{4-7}$$

Pour chaque fermeture, le vecteur des paramètres $\mathbf{z} \equiv (a_1, b_1, a_2, b_2, Y)^T$ pour lequel $\mathbf{g} \equiv (g_1, g_2, g_3, g_4, g_5)^T = \mathbf{0}$ peut être trouvé par une itération Newton - Raphson, la matrice Jacobienne $\partial\mathbf{g}/\partial\mathbf{z}$ requise étant soit évalué analytiquement soit approximé par des quotients de différence centraux. Les vecteurs de paramètres ainsi obtenus sont tous deux de bonnes approximations des valeurs correctes, mais ne sont pas identiques.

La manière théoriquement la plus satisfaisante d'obtenir un compromis entre les deux fermetures des équations des moments est de calculer la solution des moindres carrés du système non linéaire surdéterminé en utilisant les deux équations 4-7 et 4-8. C'est l'approche utilisée dans les travaux de Koutbeiy [1990] sur les modèles de mélange basés sur la distribution de von Mises. Cette approche peut être généralisée et étendue en considérant les cinq premiers moments trigonométriques et en trouvant la solution des moindres carrés du modèle à deux équations défini implicitement par les équations

$$g_{1,n} = \frac{1+\tanh Y}{2}\frac{\sin nb_1}{\cosh(n\pi/2a_1)} + \frac{1-\tanh Y}{2}\frac{\sin nb_2}{\cosh(n\pi/2a_2)} - S_n \qquad (4\text{-}8)$$

$$g_{2,n} = \frac{1+\tanh Y}{2}\frac{\cos nb_1}{\cosh(n\pi/2a_1)} + \frac{1-\tanh Y}{2}\frac{\cos nb_2}{\cosh(n\pi/2a_2)} - C_n$$

aux cinq points (n, S_n, C_n), pour $n = 1$ à 5; le système surdéterminé comprend dix équations en cinq inconnus. Avec la supposition de poids uniformes, la matrice de covariance pour chaque point est

$$\boldsymbol{\sigma} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix}, \qquad (4\text{-}9)$$

et la solution moindres-carrés de cette modèle non-linéaire à variables indépendants fixes peut être obtenue moyennant la généralisation à plusieurs équations de la méthode classique de Gauss-Newton. (La valeur de σ^2 n'est pas pertinente, car elle s'annule des équations normales pour les incréments de paramètre.) Ce calcul itératif a été exécuté avec un programme R basé sur la formulation implicite des problèmes moindres-carrés généralisés donnée par Lybanon [1984, 1985]), comme généralisée par Marshall [2003] (voir aussi Marshall et Blencoe [2005]) pour les problèmes à plusieurs équations de condition.

4.3 Simulation et Validation

La création d'un ensemble de données angulaires synthétiques pour valider l'ajustement par la méthode des moments se déroule comme pour l'ajustement des distributions sech simples enroulées, mais avec une densité de probabilité donnée par l'équation 4-2 et la probabilité cumulée

$$\Phi(x|a_1, b_1, a_2, b_2, Y) =$$

$$F(x|a_1, b_1)(1 + \tanh Y)/2 + F(x|a_2, b_2)(1 - \tanh Y)/2. \tag{4-10}$$

Considérons par exemple 10000 angles générés d'une distribution sech enroulée mixte, avec $a_1 = 1.5$, $b_1 = 2.5$, $a_2 = 2.1$, $b_2 = 4.8$, et $Y = 0.15$, ce qui est bien bimodale.

La génération d'un point du départ pour l'itération moindres-carrés est plus complexe pour le modèle mixte, et se déroule en deux étapes. Tout d'abord, les paramètres b sont estimés des positions des pics dans l'histogramme des données. Ensuite, supposant que X=0.5 (Y=0), les équations pour les moments sinus et cosinus du premier ordre peuvent être utilisées pour générer des valeurs pour a_1 et a_2, selon

$$\begin{bmatrix} X\cos b_1 & (1-X)\cos b_2 \\ X\sin b_1 & (1-X)\sin b_2 \end{bmatrix} \begin{bmatrix} 1/\cosh(\pi/2a_1) \\ 1/\cosh(\pi/2a_2) \end{bmatrix} = \begin{bmatrix} C_1 \\ S_1 \end{bmatrix}. \tag{4-11}$$

La règle de Cramer donne

$$\cosh(\pi/2a_1) = \frac{X\sin(b_2 - b_1)}{C_1\sin b_2 - S_1\cos b_2} \tag{4-12}$$

et

$$\cosh(\pi/2a_2) = \frac{(1-X)\sin(b_2 - b_1)}{S_1\cos b_1 - C_1\sin b_1}. \tag{4-13}$$

La Figure 6 montre l'histogramme des données et la densité de probabilité avec les paramètres utilisés pour la génération des données (la courbe noire), et celle de $(a_1, b_1, a_2, b_2, Y) = (1.4887, 2.537668, 2.137992, 4.82255, 0.161089)$ détérminés par la solution des équations des moments (la courbe rouge.) Les courbes concordent bien entre elles et avec les données.

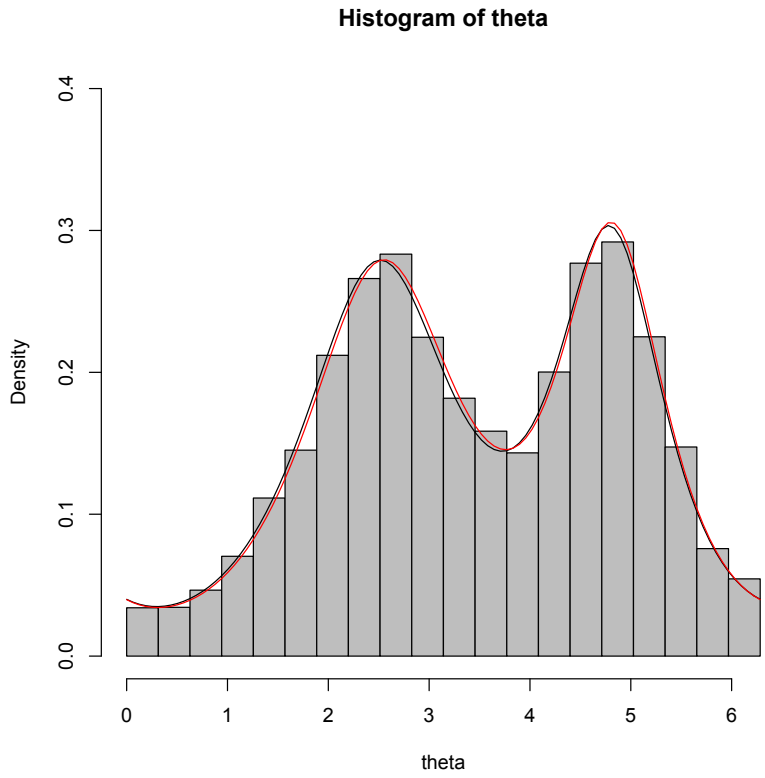

Figure 6: Histogramme de 10000 angles aléatoires générés de l'équation 4-2 avec $a_1 = 1.5$, $b_1 = 2.5$, $a_2 = 2.1$, $b_2 = 4.8$, Y=0.15, avec les densités de probabilité originale (courbe noire) et ajustée (courbe rouge).

5 Application aux Données de Direction du Vent

Les calculs des paragraphes précédents ont démontré que l'ajustement des moments par la méthode des moindres carrés est bien capable de récupérer les paramètres à partir d'ensembles de données synthétiques. Il reste à déterminer si les procédures sont tout aussi efficaces lorsqu'elles sont appliquées aux données de direction du vent, pour lesquelles les réponses correctes ne sont pas connues! Plus précisément, on peut se demander si la

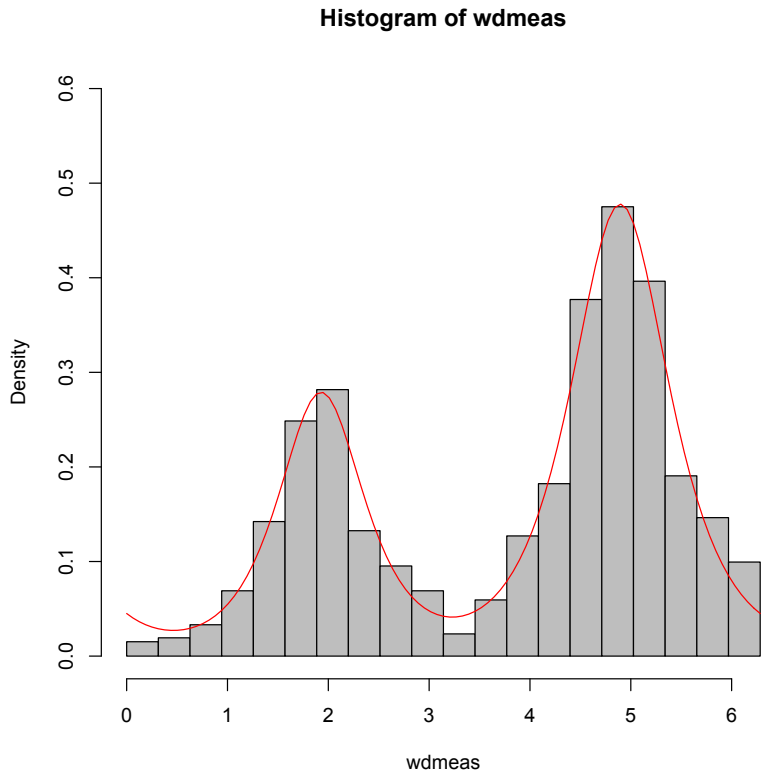

Figure 7: Histogramme de 2305 mesures de direction du vent près de Collie, Australie occidentale. Courbe : ajustement de l'équation 4-2 (a_1, b_1, a_2, b_2, Y) = (2.581866, 1.927496, 2.26429, 4.895631, -0.3372373).

tendance leptokurtique de la distribution sech est-elle compatible avec cette application? Pour étudier cela, on peut utiliser 2305 données de direction du vent recueillies à des intervalles de 10 minutes à une station gouvernementale de surveillance de la qualité de l'air à proximité de Collie, Australie occidentale, entre le 24 septembre et le 10 octobre 1998. Le bon accord entre l'histogramme de ces données et la courbe ajustée (Figure 7) suggèrent une réponse affirmative. Le processus de la paramétrisation ajuste avec succès les paramètres a_1 et a_2 pour s'adapter aux différentes largeurs et hauteurs des pics - les valeurs des paramètres sont (a_1, b_1, a_2, b_2, Y) = (2.581866, 1.927496, 2.26429, 4,895631, -0,3372373). Les déviations apparentes aux extrémités

du domaine résultent en partie de la condition aux limites périodique (la courbure vers le haut à l'approche de zéro) ; cet effet est beaucoup moins prononcé dans l'ajustement de l'échantillon plus large utilisé dans la figure 6.

6 Conclusions

Une nouvelle distribution angulaire, obtenue de l'enroulement d'une distribution de la sécante hyperbolique à deux paramètres sur $[0, 2\pi]$, a été décrite. Application de la FSP produit une série de Fourier à convergence rapide, qui fournit des expressions simples et efficaces pour tous les moments circulaires. L'ajustement de vraisemblance maximale des distributions sech enroulées simples, et la paramétrisation de modèles de mélange binaires par l'ajustement moindres-carrés des cinq premiers moments circulaires ont été démontrés. Cette procédure a été appliquée avec succès aux mesures de la direction du vent.

Bibliographie

Radian Belu and Darko Koracin. Statistical and spectral analysis of wind characteristics relevant to wind energy assessment using tower measurements in complex terrain. *Journal of Wind Energy*, 739162:1–12, 2013.

John Bentley. Modelling circular data using a mixture of von Mises and uniform distribution. Master's thesis, Simon Fraser University, Burnaby, BC, Canada, Fall 2006.

Peng Ding. Three occurrences of the hyperbolic secant distribution. *The American Statistician*, 68(1):32–35, 2014.

Arthur Erdélyi, Wilhelm Magnus, Fritz Oberhettinger, and Francesco G. Tricomi. *Tables of Integral Transforms*, volume 1. McGraw-Hill, New York, 1954. Based, in part, on notes left by Harry Bateman; prepared at California Institute of Technology as part of the Bateman Manuscript Project.

R.A. Fisher. On the "probable errror" of a coefficient of correlation deduced from a small sample. *Metron*, 1(4):3–32, 1921.

Kurt Hornik and Bettina Grün. movMF: an R package for fitting mixtures of von Mises-Fisher distributions. *Journal of Statistical Software*, 58(1): 1–31, 2014.

Majdi Amine Koutbeiy. *Estimating the Parameters in Mixtures of Circular and Spherical Distributions*. PhD thesis, University of St Andrews, St Andrews, Scotland, 1990.

A. Losev. A new lineshape for fitting X-ray photoelectron peaks. *Surface and Interface Analysis*, 14(12):845–849, 1989.

M. Lybanon. A better least-squares method when both variables have uncertainties. *American Journal of Physics*, 52:22–26, 1984.

M. Lybanon. A simple generalized least-squares algorithm. *Computers and Geosciences*, 11:501–508, 1985.

Kanti V. Mardia and Peter E. Jupp. *Directional Statistics*. Wiley, Chichester, 2000.

Simon L. Marshall. Convergence acceleration of Fourier series by analytical and numerical application of Poisson's formula. *Journal of Physics A: Mathematical and General*, 31:2691–2704, 1998a.

Simon L. Marshall. On the analytical summation of Fourier series and its relation to the asymptotic behavior of transforms. *Journal of Physics A: Mathematical and General*, 31:9957–9973, 1998b.

Simon L. Marshall. A rapidly-converging modified Green's function for Laplace's equation in a rectangle. *Proceedings of the Royal Society of London, A*, 455:1739–1766, 1999.

Simon L. Marshall. A periodic Green function for calculation of Coulombic lattice potentials. *Journal of Physics: Condensed Matter*, 12:4575–4601, 2000.

Simon L. Marshall. Calculation of coulombic lattice potentials. Part II. Spherical harmonic expansion of the Green function. *Journal of Physics: Condensed Matter*, 14:3175–3198, 2002.

Simon L. Marshall. Generalized least-squares parameter estimation from multi-equation implicit models. *AIChE Journal*, 49(10):2577–2594, 2003.

Simon L. Marshall and James G. Blencoe. Generalized least-squares fit of multi-equation models. *American Journal of Physics*, 73(1):69–82, 2005.

Nurulkamal Masseran, Ahmad Mahir Razali, and Kamarulzaman Ibrahim. Application of mixtures of von Mises-Fisher model to investigate the statistical characteristics of the wind direction data. *WSEAS Transactions on Mathematics*, 14:313–323, 2015.

Saralees Nadarajah and Yuanyuan Zhang. Wrapped: An R package for circular data. *PLoS ONE*, 12(12):e0188512, 2017.

Wilfred Perks. On some experiments in the graduation of mortality statistics. *Journal of the Institute of Actuaries*, 58:12–57, 1932.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2 edition, 1992.

A.M. Razali, A. Ahmad, M.S. Sapuan, and A. Zaharim. Circular statistics: An analysis of wind direction data. In Sorinel Oprisan, Azami Zaharim, Saeid Eslamian, Ming-Shen Jian, Claudia A.F. Aiub, and Ahadollah Azami, editors, *Advances in Environment, Computational Chemistry and Bioscience*, Proceedings of the 10th WSEAS International Conference on Environment, Ecosystems and Development (EED12), pages 153–156, Montreux, Switzerland, December 29-31 2012.

Gordon K. Smyth. A note on modelling cross-correlations: Hyperbolic secant regression. *Biometrika*, 81(2):396–402, 1994.

Joseph Talacko. Perks' distributions and their role in the theory of Wiener's stochastic variables. *Trabajos de Estadistica*, 7(2):159–174, 1956.

Joseph Talacko. A note about a family of Perks' distributions. *Sankhya*, 20 (3):323–328, 1958.

A Codes Sources (non destinés à la publication)

Les codes R utilisés pour générer les illustrations du présent article sont présentés ici.

A.1 Figures 1 et 2

La densité de probabilité et la fonction de répartition de la distribution de la sécante hyperbolique, montrées dans les Figures 1 et 2, respectivement, ont été calculées par le programme suivant `WrappedSechPaper_Figure1+2.R`:

```
sechdist.pdf <- function(a,b,x){
    pdf <- (a/pi)/cosh(a*(x-b))
    return(pdf)
}
sechdist.cdf <- function(a,b,x){
    cdf <- (2/pi)*atan(exp(a*(x-b)))
    return(cdf)
}

a1 <- 0.5
b1 <- 2.5
a2 <- 2.1
b2 <- 4.8
npoint <- 100
X <- vector(length=npoint)
Y1 <- vector(length=npoint)
Y2 <- vector(length=npoint)
for(k in 1:npoint){
  X[k] <- 2*pi*k/npoint
  Y1[k] <- sechdist.pdf(a1,b1,X[k])
  Y2[k] <- sechdist.pdf(a2,b2,X[k])
}
picname="WrappedSechPaper_Figure1.pdf"
pdf(picname)
plot(X,Y1,type="l",ylim=c(0,0.7),ylab="pdf")
text(0.3,0.15,"One")
text(5.2,0.65,"Two")
text(1,0.70,paste("One: a = ",as.character(a1),"b = ",
as.character(b1)))
text(1,0.67,paste("Two: a = ",as.character(a2),"b = ",
as.character(b2)))
```

```
lines(X,Y2)
dev.off()
for(k in 1:npoint){
  X[k] <- 2*pi*k/npoint
  Y1[k] <- sechdist.cdf(a1,b1,X[k])
  Y2[k] <- sechdist.cdf(a2,b2,X[k])
}
picname="WrappedSechPaper_Figure2.pdf"
pdf(picname)
plot(X,Y1,type="l",ylim=c(0,1),ylab="cdf")
text(0.3,0.15,"One")
text(5.8,1,"Two")
text(1,1.00,paste("One: a = ",as.character(a1),"b = ",
as.character(b1)))
text(1,0.95,paste("Two: a = ",as.character(a2),"b = ",
as.character(b2)))
lines(X,Y2)
dev.off()
```

A.2 Figures 3 et 4

Les contreparties enroulées de la distribution sech des Figures 1 et 2 ontété calculées par le suivant:

```
wsech.pdf <- function(a,b,x){
  # This subprogram evaluates the probability density function of the
  # wrapped sech distribution for scale parameter a, location parameter
  # b, and argument x
  #
  epsilon <- 1e-15
  xb <- x-b
  n1 <- (acosh(1/epsilon)/a-xb)/(2*pi)
  n2 <- (2*a/pi)*acosh(1/epsilon)
  n2n1 <- n2-n1
  ssign <- ifelse(n2n1 >= 0,1,0)
  #ssign <- sign(n2-n1)
  #if(n2==n1){ssign <- 1}
  tsign <- ssign*ssign
  term1 <- 1/cosh(a*xb)
  term2 <- 1/(2*a)
  term <- tsign*(term1*(1+ssign)/2+term2*(1-ssign)/2)+(1-tsign)*term1
  sum <- term
  for(n in 1:min(n1,n2)){
    term1 <- 1/cosh(a*(xb+2*pi*n))+1/cosh(a*(xb-2*pi*n))
    term2 <- cos(n*xb)/(a*cosh(n*pi/(2*a)))
```

```
    term <- tsign*(term1*(1+ssign)/2+term2*(1-ssign)/2)+term1*(1-tsign)
    sum <- sum+term
  }
  pdf <- sum*a/pi
  return(pdf)
}
wsech.cdf <- function(a,b,x){
  # This subprogram returns the cdf of the wrapped hyperbolic secant
  # distribution
  epsilon <- 1e-15
  xb <- x-b
  n1 <- 1+floor((xb + acosh(1/epsilon)/a)/(2*pi))
  n2 <- 1+floor(acosh(1/epsilon)*2*a/pi)
  if(n1 >= n2){
  # Use transformed series
    sum2 <- x/2
    for(n in 1:n2){
      term2 <- (sin(n*xb)+sin(n*b))/(n*cosh(n*pi/(2*a)))
      sum2 <- sum2+term2
    }
    cdf <- sum2/pi
    return(cdf)
  } else {
  # Use original series
    sum1 <- atan(exp(a*xb))-atan(exp(-a*b))
    for(n in 1:n1){
      term1 <- atan(exp(a*(2*n*pi+xb)))-atan(exp(a*(2*n*pi-b)))+
      atan(exp(a*(-2*n*pi+xb)))-atan(exp(a*(-2*n*pi-b)))
      # cat(n,term1,"\n")
      sum1 <- sum1+term1
    }
    cdf <- sum1*2/pi
    return(cdf)
  }
}


a1 <- 0.5
b1 <- 2.5
a2 <- 2.1
b2 <- 4.8
npoint <- 100
X <- vector(length=npoint)
Y1 <- vector(length=npoint)
Y2 <- vector(length=npoint)
```

```
for(k in 1:npoint){
  X[k] <- 2*pi*k/npoint
  Y1[k] <- wsech.pdf(a1,b1,X[k])
  Y2[k] <- wsech.pdf(a2,b2,X[k])
}
picname="WrappedSechPaper_Figure3.pdf"
pdf(picname)
plot(X,Y1,type="l",ylim=c(0,0.7),ylab="pdf")
text(0.3,0.17,"One")
text(5.2,0.65,"Two")
text(1,0.70,paste("One: a = ",as.character(a1),"b = ",
as.character(b1)))
text(1,0.67,paste("Two: a = ",as.character(a2),"b = ",
as.character(b2)))
lines(X,Y2)
dev.off()
for(k in 1:npoint){
  X[k] <- 2*pi*k/npoint
  Y1[k] <- wsech.cdf(a1,b1,X[k])
  Y2[k] <- wsech.cdf(a2,b2,X[k])
}
picname="WrappedSechPaper_Figure4.pdf"
pdf(picname)
plot(X,Y1,type="l",ylim=c(0,1),ylab="cdf")
text(0.3,0.15,"One")
text(4.5,0.15,"Two")
text(1,1.00,paste("One: a = ",as.character(a1),"b = ",
as.character(b1)))
text(1,0.95,paste("Two: a = ",as.character(a2),"b = ",
as.character(b2)))
lines(X,Y2)
dev.off()
```

A.3 Figure 5

Les 10000 nombres aléatoires générés selon la distribution sech enroulée, montrés dans l'histogramme de la Figure 5 et sa courbe ajustée, sont contenus dans le fichier `junk.data2`. Ceux-ci ont été générés par la méthode de transformation, utilisant le générateur de nombres aléatoires uniformes intégré dans R, qui est implémenté dans le suivant

```
pl2slv <- function(X1,Y1,X2,Y2,X3,Y3,Y){
  # This subprogram calculates the abscissa of the Lagrange
  # interpolation parabola that passes through the three points
```

```
  # (X1,Y1), (X2,Y2), (X3,Y3) corresponding to the ordinate Y
  #
  A <- Y1/(X1-X2)/(X1-X3)+Y2/(X2-X1)/(X2-X3)+Y3/(X3-X1)/(X3-X2)
  B <- -Y1*(X2+X3)/(X1-X2)/(X1-X3)-Y2*(X1+X3)/(X2-X1)/(X2-X3)-
  Y3*(X1+X2)/(X3-X1)/(X3-X2)
  C <- Y1*X2*X3/(X1-X2)/(X1-X3)+Y2*X1*X3/(X2-X1)/(X2-X3)+
  Y3*X1*X2/(X3-X1)/(X3-X2)
  CY <- C-Y
  D <- B*B-4*A*CY
  # cat(Y,D,"\n")
  if(D > 0){
    XY1 <- (-B+sqrt(D))/(2*A)
    XY2 <- (-B-sqrt(D))/(2*A)
    # Select the root that lies between X1 and X3
    if((XY1 > X1) && (XY1 < X3)){
      return(XY1)
    } else if ((XY2 > X1) && (XY2 < X3)){
      return(XY2)
    }
  } else if(D < 0){
    XY <- X2
    return(XY)
  }
}
wsech.rng <- function(a,b,n){
  # This subprogram generates a vector of random numbers of length n
  # following the wrapped sech distribution with parameters a and b
  #
  itmax <- 10                       # Maximum number of N-R iterations
  tol <- 1e-15                      # Relative error tolerance
  npoint <- 100                     # Number of look-up points
  theta <- vector(length=n)         # Resulting random number array
  prob <- runif(n)                  # Random numbers from U(0,1)
  X <- vector(length=npoint)        # Abscissae of lookup table
  Y <- vector(length=npoint)        # Ordinates of lookup table
  #
  # Calculation of CDF values to initiate N-R iteration
  # cat("Values for lookup","\n")
  for(k in 1:npoint){
    x <- 2*pi*k/npoint              # Points separated by 2*pi/npoint
    pdf <- wsech.pdf(a,b,x)
    cdf <- wsech.cdf(a,b,x)
    X[k] <- x
    Y[k] <- cdf
    # cat(X[k],Y[k],"\n")
```

```
  }
  # Find abscissa for each uniform probability
  for(j in 1:n){
    p <- prob[j]
    # cat("Cumulative probability = ",p,"\n")
    # Bracketing abscissae
    x1 <- 0
    x3 <- X[1]
    for(k in 1:npoint){
      if((p > Y[k]) && (p < Y[k+1])){
        x1 <- X[k]
        y1 <- Y[k]
        x3 <- X[k+1]
        y3 <- Y[k+1]
        break
      }
    }
    # cat("Root between",x1," and ",x3,"\n")
    # Approximate solution by inverse quadratic interpolation
    x2 <- (x1+x3)/2
    y2 <- wsech.cdf(a,b,x2)
    x <- pl2slv(x1,y1,x2,y2,x3,y3,p)
    # cat("Quadratic approximation = ",x,"\n")
    #
    # Newton-Raphson iteration
    for(iter in 1:itmax){
      f <- wsech.cdf(a,b,x)-p
      fprime <- wsech.pdf(a,b,x)
      deltax <- -f/fprime
      relerr <- abs(deltax/x)
      # cat("N-R iteration",iter,"Relative error",relerr,"\n")
      if(relerr < tol){break}
      x <- x+deltax
    }
  theta[j]=x
  }
  return(theta)
}
```

Appels a ce sous-programme ont été déactivés dans le programme qui a produit la figure, afin de permettre l'usage du même fichier plusieurs fois:

```
z <- vector(length=2)
# a <- 2.1
# b <- 4.8
```

```
# n <- 10000
# theta <- wsech.rng(a,b,n)
X <- read.table("junk.data2")
theta <- X[[1]]
# pdfname <- "WrappedSechPaper_Figure5.pdf"
# pdf(pdfname)
hist(theta,freq=FALSE,col="grey",breaks=((0:20)*pi/10),
xlim=c(0,2*pi),ylim=c(0,0.8))
# curve(wsech.pdf(a,b,x),from=0,to=2*pi,add=TRUE)
z <- wsech.mle(theta)
a <- z[1]
b <- z[2]
curve(wsech.pdf(a,b,x),from=0,to=2*pi,col="red",add=TRUE)
# dev.off()
cat(z,"\n")
```

L'optimisation vraisemblance-maximale des paramètres est executée par

```
wsech.mle <- function(theta){
  # This subprogram performs a maximum likelihood fit of a vector of
  # angular variables theta to the Wrapped Sech Distribution
  #
  n <- 1
  moments <- SCmoments(n,theta)
  S1 <- moments[1,1]
  C1 <- moments[1,2]
  cat(S1,C1,"\n")
  ratio <- S1/C1
  b <- atan(S1/C1)
  if(ratio < 0){b <- b+2*pi}
  a <- (pi/2)/acosh(1/sqrt(S1^2+C1^2))
  z[1] <- a
  z[2] <- b
  zrange <- c(0.001,0.001)
  ftol <- 1.e-07
  cat("Start Nelder-Mead optimization at:","\n")
  cat(z,"\n")
  z <- dsm.min(wsech.llf,theta,z,zrange,ftol)
  return(z)
}
```

Le premier étape de ce processus est le calcul des moments sinus et cosinus des données, de l'ordre $n = 1$, déterminés par le sous-programme

```
SCmoments <- function(n,theta){
```

```
  # This subprogram returns the sine and cosine moments of a data set
  # theta[1:ndata], up to a given order n, as the elements of a matrix
  # SC[1:n,1:2]
  #
  ndata <- length(theta)
  SC <- matrix(nrow=n,ncol=2)
  SC[1:n,]=0
  #
  # Accumulation of sums
  for(k in 1:ndata){
    thetak <- theta[k]
    for(j in 1:n){
      SC[j,1] <- SC[j,1] + sin(j*thetak)
      SC[j,2] <- SC[j,2] + cos(j*thetak)
    }
  }
  # 1st to nth trigonometric moments
  for(i in 1:n){
    SC[i,] <- SC[i,]/ndata
  }
  return(SC)
}
```

qui s'appliquent à la génération des valeurs approchées des paramètres selon les équations 3-27 et 3-28. L'optimisation soi-même se déroule par l'application de la méthode Nelder-Mead, implémentée dans les sous-programmes suivants `dsmmin` and `amoeba`,

```
amoeba <- function(funk,y,p,q,ftol){
  # Definition of parameters...
  alpha <- 1
  beta <- .5
  gamma <- 2
  itmax <- 100
  # ...and array declarations
  nz <- ncol(p)
  nz1 <- nz+1
  pq <- matrix(nrow=nz1,ncol=nz1) #Simplex and function values
  pbar <- vector(length=nz)
  pr <- vector(length=nz)
  prr <- vector(length=nz)
  iter <- 0
  # Main iteration loop starts here:
  while(iter <= itmax){
    ilo <- 1
```

```
if(q[1] > q[2]){
  ihi <- 1
  inhi <- 2
} else {
  ihi <- 2
  inhi <- 1
}
for(i in 1:nz1){
  if(q[i] < q[ilo]){ilo <- i}
  if(q[i] > q[ihi]){
    inhi <- ihi
    ihi <- i
  } else if(q[i] > q[inhi]){
    if(i != ihi){inhi <- i}
  }
}
# Fractional range from highest to lowest
rnum <- q[ihi]-q[ilo]
rden <- q[ihi]+q[ilo]
rtol <- 2*abs(rnum)/abs(rden)
pq <- cbind(p,q)
if(rtol < ftol || abs(rnum) < ftol){break}
iter <- iter+1
# cat("At iteration number ",iter,q[ihi],q[ilo],rtol,"\n")
for(j in 1:nz){pbar[j] <- 0}
for(i in 1:nz1){
  if(i != ihi) {
    for(j in 1:nz){
      pbar[j] <- pbar[j]+p[i,j]
    }
  }
}
pbar <- pbar/nz
pr <- (1+alpha)*pbar-alpha*p[ihi,]
# cat("pbar = ",pbar,"\n")
# cat("pr   = ",pr,"\n")
qpr <- funk(y,pr)
if(qpr <= q[ilo]){
  prr <- gamma*pr+(1-gamma)*pbar
  qprr <- funk(y,prr)
  if(qprr < q[ilo]){
    p[ihi,] <- prr
    q[ihi] <- qprr
  } else {
    p[ihi,] <- pr
```

```
        q[ihi] <- qpr
      }
    } else if(qpr >= q[inhi]){
      if(qpr < q[ihi]){
        p[ihi,] <- pr
        q[ihi] <- qpr
      }
      prr <- beta*p[ihi,]+(1-beta)*pbar
      qprr <- funk(y,prr)
      if(qprr < q[ihi]){
        p[ihi,] <- prr
        q[ihi] <- qprr
      } else {
        for(i in 1:nz1){
          if(i != ilo){
            pr <- (p[i,]+p[ilo,])/2
            p[i,] <- pr
            q[i] <- funk(y,pr)
          }
        }
      }
    } else {
      p[ihi,] <- pr
      q[ihi] <- qpr
    }
    #cat("\n")
  }
  return(pq)
}
dsm.min <- function(funk,y,z,zrange,ftol){
  # This subprogram finds the minimum of funk(y,z), with respect
  # to the components of the vector z[1:nz], the other parameters in
  # vector y[1:ny] being held constant, by means of the Nelder--Mead
  # Downhill Simplex Method. The nz+1 vertices of the initial simplex
  # are obtained from the starting point z by componentwise addition of
  # the array zrange[1:nz]
  #
  # Array declarations
  ny <- length(y)
  nz <- length(z)
  nz1 <- nz+1
  w <- vector(length=nz)
  z1 <- vector(length=nz)
  zero <- matrix(nrow=1,ncol=nz)
  p <- matrix(nrow=nz1,ncol=nz)
```

```
  pw <- matrix(nrow=nz1,ncol=nz1)
  # Initial polytope and function values
  for(i in 1:nz1){p[i,] <- z}
  for(j in 1:nz){zero[,j] <- 0}
  p <- p+rbind(zero,diag(zrange))
  for(i in 1:nz1){
    z1 <- p[i,]
    w[i] <- funk(y,z1)
  }
  wmin <- min(w)
  # Optimize funk by Nelder-Mead method
  pw <- amoeba(funk,y,p,w,ftol)
  # Select best point and return
  p <- pw[,-nz1]
  w <- pw[,nz1]
  for(i in 1:nz1){
    if(w[i] < wmin){
      wmin <- w[i]
      imin <- i
    }
  }
  z <- p[imin,]
  return(z)
}
```

à la fonction log vraisemblance negative calculée par le sous-programme `wsech.llf`.

```
wsech.llf <- function(y,z){
  # This subprogram evaluates the log-likelihood function for the
  # wrapped hyperbolic secant distribution, for the angular data in
  # array y[1:ndata] and parameters in array z{1:2]
  #
  ssign <- -1
  ndata <- length(y)
  a <- z[1]
  b <- z[2]
  sum <- 0
  for(k in 1:ndata){
    theta <- y[k]
    pdf <- wsech.pdf(a,b,theta)
    sum <- sum+log(pdf)
    }
  wsechllf <- ssign*sum/ndata
  cat("wsech.llf(",z,")=",wsechllf,"\n")
```

```
  return(wsechllf)
}
```

A.4 Figure 6

L'évaluation de la distribution sech enroulée mixte est réalisée par les sous-programmes

```
wsech2.pdf <- function(a1,b1,a2,b2,y,theta){
  # This subprogram evaluates the probability density function for a
  # mixture of two wrapped sech distributions of respective parameters
  # a1,b1 and a2,b2, where the fraction of the first and second
  # distributions are [1+tanh(y)]/2 and [1-tanh(y)]/2, respectively
  #
  x <- (1+tanh(y))/2
  pdf <- x*wsech.pdf(a1,b1,theta)+(1-x)*wsech.pdf(a2,a2,theta)
  return(pdf)
}
wsech2.cdf <- function(a1,b1,a2,b2,y,theta){
  # This subprogram evaluates the cumulative probability function for a
  # mixture of two wrapped sech distributions of respective parameters
  # a1,b1 and a2,b2, where the fraction of the first and second
  # distributions are [1+tanh(y)]/2 and [1-tanh(y)]/2, respectively
  #
  x <- (1+tanh(y))/2
  cdf <- x*wsech.cdf(a1,b1,theta)+(1-x)*wsech.cdf(a2,a2,theta)
  return(cdf)
}
```

et la génération des nombres aléatoires d'une telle distribution selon la méthode de transformation par

```
wsech2.rng <- function(a1,b1,a2,b2,y,n){
  # This subprogram generates a vector of random numbers of length n
  # following the mixture of wrapped sech distributions for which the
  # parameters are a1,b1 and a2,b2 and the respective mixture fractions
  # are xmix = [1+tanh(y)]/2 and 1-xmix = [1-tanh(y)]/2
  #
  itmax <- 10                       # Maximum number of N-R iterations
  tol <- 1e-15                      # Relative error tolerance
  npoint <- 100                     # Number of look-up points
  theta <- vector(length=n)         # Resulting random number array
  prob <- runif(n)                  # Random numbers from U(0,1)
  X <- vector(length=npoint)        # Abscissae of lookup table
```

```
Y <- vector(length=npoint)       # Ordinates of lookup table
#
# Calculation of CDF values to initiate N-R iteration
# cat("Values for lookup","\n")
xmix <- (1+tanh(y))/2
for(k in 1:npoint){
  x <- 2*pi*k/npoint         # Points separated by 2*pi/npoint
  pdf <- xmix*wsech.pdf(a1,b1,x)+(1-xmix)*wsech.pdf(a2,b2,x)
  cdf <- xmix*wsech.cdf(a1,b1,x)+(1-xmix)*wsech.cdf(a2,b2,x)
  X[k] <- x
  Y[k] <- cdf
  # cat(X[k],Y[k],"\n")
}
# Find abscissa for each uniform probability
for(j in 1:n){
  p <- prob[j]
  # cat("Cumulative probability = ",p,"\n")
  # Bracketing abscissae
  x1 <- 0
  x3 <- X[1]
  for(k in 1:npoint){
    if((p > Y[k]) && (p < Y[k+1])){
      x1 <- X[k]
      y1 <- Y[k]
      x3 <- X[k+1]
      y3 <- Y[k+1]
      break
    }
  }
  # cat("Root between",x1," and ",x3,"\n")
  # Approximate solution by inverse quadratic interpolation
  x2 <- (x1+x3)/2
  y2 <- xmix*wsech.cdf(a1,b1,x2)+(1-xmix)*wsech.cdf(a2,b2,x2)
  x <- pl2slv(x1,y1,x2,y2,x3,y3,p)
  # cat("Quadratic approximation = ",x,"\n")
  #
  # Newton-Raphson iteration
  for(iter in 1:itmax){
    f <- xmix*wsech.cdf(a1,b1,x)+(1-xmix)*wsech.cdf(a2,b2,x)-p
    fprime <- xmix*wsech.pdf(a1,b1,x)+(1-xmix)*wsech.pdf(a2,b2,x)
    deltax <- -f/fprime
    relerr <- abs(deltax/x)
    # cat("N-R iteration",iter,"Relative error",relerr,"\n")
    if(relerr < tol){break}
    x <- x+deltax
```

```
    }
  theta[j]=x
  }
  return(theta)
}

a1 <- 0.5
b1 <- 2.5
a2 <- 2.1
b2 <- 4.8
y <- 0.15
n <- 10
theta <- wsech2.rng(a1,b1,a2,b2,y,n)
for(i in 1:10){
  cat(theta[i],"\n")
}
```

Les données synthétiques qui sont produites à cette manière pour vérifier les méthodes de paramétrisation sont contenues dans le fichier `junk.data3`. Pour l'ajustement moindres-carrés des premiers cinq moments sinus et cosinus, ces quantités sont évaluées par

```
wsech2.moments <- function(x,z){
  # This subprogram evaluates the implicit form of the two equations
  # relating the sine and cosine moments of order n = x[1] for the
  # mixed wrapped sech distribution, the parameters of which are in
  # z[1:5], to those derived from the data, which are in x[2:3].
  #
  g <- vector(length=2)
  n <- x[1]
  Sn <- x[2]
  Cn <- x[3]
  a1 <- z[1]
  b1 <- z[2]
  a2 <- z[3]
  b2 <- z[4]
  Y <-z[5]
  X <- (1+tanh(Y))/2
  arg1 <- pi/(2*a1)
  arg2 <- pi/(2*a2)
  g[1] <- X*sin(n*b1)/cosh(n*arg1)+(1-X)*sin(n*b2)/cosh(n*arg2)-Sn
  g[2] <- X*cos(n*b1)/cosh(n*arg1)+(1-X)*cos(n*b2)/cosh(n*arg2)-Cn
  return(g)
}
```

L'optimisation moindres-carrés est exécutée par le programme `gnls`

```
gnls <- function(funk,nf,X0,Sigmax,ndata,z){
  # This subprogram performs a Generalized Noninear Least Squares
  # fit of the nf-dimensional vector-valued function funk to a vector
  # X0 of regressors containing ndata data with nx variables at each
  # point, for which the block diagonal variance - covariance matrix is
  # Sigmax, supplied as a rectangular matrix with ndata*nx rows and
  # nx columns. Also supplied is a vector z[1:nz] containing the
  # initial estimates of the parameters.
  #
  # Iteration parameters and key dimensions
  itmax <- 20
  tol <- 1.e-10
  nz <- length(z)
  nx <- ncol(Sigmax)
  #
  # Arrays for the whole data set
  A <- matrix(nrow=nz,ncol=nz)        #Normal eqn. coefficient matrix
  B <- vector(length=nz)              #Normal eqn. right hand side
  Deltax <- vector(length=ndata*nx)   #Regressor increments
  Deltaz <- vector(length=nz)         #Parameter increments
  F <- vector(length=(nf*ndata))      #Constraints
  Fx <- matrix(nrow=nf*ndata,ncol=nx) #x-Jacobian
  Fz <- matrix(nrow=nf*ndata,ncol=nz) #z-Jacobian
  G <- vector(length=nf*ndata)        #f-grad_xfr+grad_zfDelta z
  R <- vector(length=(nx*ndata))      #Residuals
  W <- matrix(nrow=nf*ndata,ncol=nf)  #Weighting matrix
  X <- vector(length=(nx*ndata))      #Regressors
  #
  # Arrays for individual points
  a <- matrix(nrow=nz,ncol=nz)        #Normal eqn. coefficient matrix
  b <- vector(length=nz)              #Normal eqn. right hand side
  deltax <- vector(length=nx)         #Regressor increments
  f <- vector(length=nf)              #Constraints
  fx <- matrix(nrow=nf,ncol=nx)       #x-Jacobian
  fz <- matrix(nrow=nf,ncol=nz)       #z-Jacobian
  g <- vector(length=nf)              #f-grad_xfr+grad_zfDelta z
  lambda <- vector(length=nf)         #Lagrange multipliers
  r <- vector(length=nx)              #Residuals
  sigmax <- matrix(nrow=nx,ncol=nx)   #Variance-ovariance matrix
  w <- matrix(nrow=nf,ncol=nf)        #Weighting matrix
  x <- vector(length=nx)              #Regressors
  #
  # Initial regressor and parameter vectors
  X <- X0
```

```
R[1:(nx*ndata)] <- 0
#
# Main iteration loop:
cat("Iterating...","\n")
for(iter in 1:itmax){
  # Initialization
  for(k in 1:nz){
    A[k,] <- 0
    B[k] <- 0
  }
  F <- NULL
  Fx <- NULL
  Fz <- NULL
  G <- NULL
  W <- NULL
  #
  # Weighting matrix, normal equations
  for(i in 1:ndata){
    #
    # Calculation of single-point arrays:
    r <- R[(nx*(i-1)+1):(nx*i)]
    x <- X[(nx*(i-1)+1):(nx*i)]
    sigmax <- Sigmax[(nx*(i-1)+1):(nx*i),]
    sigmax <- sigmax + 1.e-15 * diag(nx)
    f <- funk(x,z)
    fx <- xjacob(funk,nf,x,z)
    fz <- zjacob(funk,nf,x,z)
    g <- f - fx %*% r
    w <- solve(fx %*% (tcrossprod(sigmax,fx)))
    a <- crossprod(fz, (w %*% fz))
    b <- crossprod(fz, (w %*% g))
    #
    # Accumulation of whole-set arrays:
    A <- A+a
    B <- B+b
    F <- rbind(F,f)
    Fx <- rbind(Fx,fx)
    Fz <- rbind(Fz,fz)
    G <- rbind(G,g)
    W <- rbind(W,w)
  }
  # Solution for parameter increment vector
  A <- solve(A)
  Deltaz <- -A %*% B
  G <- G + Fz %*% Deltaz
```

```
  Deltax <- NULL
  #
  # Lagrange multipliers and regressor increment vector
  for(i in 1:ndata){
    for(j in 1:nx){
      r[j] <- R[nx*(i-1)+j]
      sigmax[j,] <- Sigmax[nx*(i-1)+j,]
    }
    for(k in 1:nf){
      fx[k,] <- Fx[nf*(i-1)+k,]
      g[k] <- G[nf*(i-1)+k]
      w[k,] <- W[nf*(i-1)+k,]
    }
    lambda <- w %*% g
    deltax <- -(r + sigmax %*% crossprod(fx,lambda))
    Deltax <- rbind(Deltax,deltax)
  }
  #
  # Convergence assessed on relative error of parameter estimates
  relerr <- sqrt(crossprod(Deltaz)/crossprod(z))
  cat(iter,relerr,z,"\n")
  # Converged - exit loop and return
  if(relerr < tol){break}
  # Not yet converged - update z and R and try again
  z <- z+Deltaz
  R <- R+Deltax
  X <- R+X0
}
#
# Weighted sum of squared residuals etc.
nu <- -nz
chisq <- 0
for(i in 1:ndata){
  for(j in 1:nx){
    r[j] <- R[nx*(i-1)+j]
    sigmax[j,] <- Sigmax[nx*(i-1)+j,]
  }
  for(k in 1:nx){
    test <- ceiling(sigmax[k,k])
    if(test >= 1){nu <- nu+1} #Fixed regressors excluded from d.o.f.
  }
  sigmax <- sigmax + 1.e-15 * diag(nx)
  chisq <- chisq + crossprod(r,solve(sigmax)) %*% r
}
cat("Degrees of freedom = ",nu,"\n")
```

```
  prob <- pchisq(chisq,df=nu,lower.tail=FALSE)
  #
  # Standard errors in parameter estimates
  sigmaz <- vector(length=nz)
  for(k in 1:nz){
    sigmaz[k] <- sqrt(A[k,k]*chisq/nu)
  }
  #
  # Assembly of results in a list:
  results <- list(z,sigmaz,X,chisq,prob)
  return(results)
}
xjacob <- function(funk,n,x,y,z){
  # This subprogram calculates the Jacobian matrix of a vector
  # valued function funk of m variables with n components,
  # with respect to x, for given vectors of fixed and
  # adjustable parameters y and z, respectively. The required
  # partial derivatives are approximated as central difference
  # quotients.
  #
  scale <- (.Machine$double.eps)^(1/3)
  m <- length(x)
  xplus <- vector(length=m)
  xminus <- vector(length=m)
  fplus <- vector(length=n)
  fminus <- vector(length=n)
  aj <- matrix(nrow=n,ncol=m)
  for(j in 1:m){
    xplus <- x
    xminus <- x
    if(x[j]==0){
      h=scale
    } else {
      h=scale*abs(x[j])
    }
    xplus[j] <- xplus[j]+h
    xminus[j] <- xminus[j]-h
    fplus <- funk(xplus,y,z)
    fminus <- funk(xminus,y,z)
    for(i in 1:n){
      aj[i,j] <- (fplus[i]-fminus[i])/(2*h)
    }
  }
  return(aj)
}
```

```
zjacob <- function(funk,n,x,y,z){
  # This subprogram calculates the Jacobian matrix of a vector
  # valued function funk of m variables with n components,
  # with respect to adjustable parameters z, for given vectors
  # of fixed parameters y and regressors x. The required
  # partial derivatives are approximated as central difference
  # quotients.
  #
  scale <- (.Machine$double.eps)^(1/3)
  m <- length(z)
  zplus <- vector(length=m)
  zminus <- vector(length=m)
  fplus <- vector(length=n)
  fminus <- vector(length=n)
  aj <- matrix(nrow=n,ncol=m)
  for(j in 1:m){
    zplus <- z
    zminus <- z
    if(z[j]==0){
      h=scale
    } else {
      h=scale*abs(z[j])
    }
    zplus[j] <- zplus[j]+h
    zminus[j] <- zminus[j]-h
    fplus <- funk(x,y,zplus)
    fminus <- funk(x,y,zminus)
    for(i in 1:n){
      aj[i,j] <- (fplus[i]-fminus[i])/(2*h)
    }
  }
  return(aj)
}
```

et finalement la figure est crée par le programme 'conducteur'

```
XX <- read.table("junk.data3")
theta <- XX[[1]]
n <- 5
nx <- 3
ndata <- n
X <- vector(length=(ndata*nx))
Sigmax <- matrix(nrow=(nx*ndata),ncol=nx)
SC <- SCmoments(n,theta)
for(i in 1:ndata){
```

```
  X[nx*(i-1)+1] <- i
  X[nx*(i-1)+2] <- SC[i,1]
  X[nx*(i-1)+3] <- SC[i,2]
  Sigmax[nx*(i-1)+1,] <- c(0,0,0)
  Sigmax[nx*(i-1)+2,] <- c(0,1,0)
  Sigmax[nx*(i-1)+3,] <- c(0,0,1)
}
nz <- 5
z <- vector(length=nz)
# a1 <- 1.5
# b1 <- 2.5
# a2 <- 2.1
# b2 <- 4.8
# ymix <- 0.15
# a1 <- 2
# b1 <- 1.6
# a2 <- 1
# b2 <- 4.5
# ymix <- 0
# Parameters estimated from first moments
a1 <- 1.535092
b1 <- 2.513274
a2 <- 1.464376
b2 <- 4.712389
ymix <- 0
z[1] <- a1
z[2] <- b1
z[3] <- a2
z[4] <- b2
z[5] <- ymix
ftol <- 1.e-02
zrange <- c(0.1,0.5,0.1,0.5,0.01)
# z <- dsm.min(wsech2.ssf,X,z,zrange,ftol)
nf <- 2
results <- gnls(wsech2.moments,nf,X,Sigmax,ndata,z)

hist(theta,freq=FALSE,col="grey",breaks=((0:20)*pi/10),
xlim=c(0,2*pi),ylim=c(0,0.4))
a1 <- 1.5
b1 <- 2.5
a2 <- 2.1
b2 <- 4.8
ymix <- 0.15
curve(wsech2.pdf(a1,b1,a2,b2,ymix,x),from=0,to=2*pi,add=TRUE)
z <- results[[1]]
```

```
a1 <- z[1]
b1 <- z[2]
a2 <- z[3]
b2 <- z[4]
ymix <- z[5]
curve(wsech2.pdf(a1,b1,a2,b2,ymix,x),from=0,to=2*pi,col="red",add=TRUE)
```

A.5 Figure 7

Les données de direction du vent analysées par la méthode des moments sont contenues dans le fichier `JFA.d04.qq`.

```
qqf <- read.table("JFA.d04.qq")
npoints <- nrow(qqf)-1
wdmeas <- vector(length=npoints)
wdcalc <- vector(length=npoints)
for(k in 2:npoints){
  wdcalc[k-1] <- as.numeric(as.character(qqf[k,5]))*pi/180
  wdmeas[k-1] <- as.numeric(as.character(qqf[k,6]))*pi/180
}
n <- 5
nx <- 3
ndata <- n
X <- vector(length=(ndata*nx))
Sigmax <- matrix(nrow=(nx*ndata),ncol=nx)
SC <- SCmoments(n,wdcalc)
for(i in 1:ndata){
  X[nx*(i-1)+1] <- i
  X[nx*(i-1)+2] <- SC[i,1]
  X[nx*(i-1)+3] <- SC[i,2]
  Sigmax[nx*(i-1)+1,] <- c(0,0,0)
  Sigmax[nx*(i-1)+2,] <- c(0,1,0)
  Sigmax[nx*(i-1)+3,] <- c(0,0,1)
}
nz <- 5
z <- vector(length=nz)
a1 <- 2
b1 <- 1.6
a2 <- 1
b2 <- 4.5
ymix <- 0
z[1] <- a1
z[2] <- b1
z[3] <- a2
z[4] <- b2
```

```
z[5] <- ymix
ftol <- 1.e-04
zrange <- c(0.1,0.5,0.1,0.5,0.01)
# Generation of an initial estimate by Nelder-Mead minimization of the
# sum of squares of the moment equations
z <- dsm.min(wsech2.ssf,X,z,zrange,ftol)
nf <- 2
results <- gnls(wsech2.moments,nf,X,Sigmax,ndata,z)

hist(wdcalc,freq=FALSE,col="grey",breaks=((0:20)*pi/10),
xlim=c(0,2*pi),ylim=c(0,0.6))
z <- results[[1]]
a1 <- z[1]
b1 <- z[2]
a2 <- z[3]
b2 <- z[4]
ymix <- z[5]
curve(wsech2.pdf(a1,b1,a2,b2,ymix,x),from=0,to=2*pi,col="red",add=TRUE)

SC <- SCmoments(n,wdmeas)
for(i in 1:ndata){
  X[nx*(i-1)+1] <- i
  X[nx*(i-1)+2] <- SC[i,1]
  X[nx*(i-1)+3] <- SC[i,2]
  Sigmax[nx*(i-1)+1,] <- c(0,0,0)
  Sigmax[nx*(i-1)+2,] <- c(0,1,0)
  Sigmax[nx*(i-1)+3,] <- c(0,0,1)
}
results <- gnls(wsech2.moments,nf,X,Sigmax,ndata,z)

hist(wdmeas,freq=FALSE,col="grey",breaks=((0:20)*pi/10),
xlim=c(0,2*pi),ylim=c(0,0.6))
z <- results[[1]]
a1 <- z[1]
b1 <- z[2]
a2 <- z[3]
b2 <- z[4]
ymix <- z[5]
curve(wsech2.pdf(a1,b1,a2,b2,ymix,x),from=0,to=2*pi,col="red",add=TRUE)
```

Pour cette figure, le point du départ de l'ajustement moindres-carrés est généré par la minimisation de la somme des fonctions carrées pour la forme implicite des équations des moments, évaluée par le sous-programme suivant `wsech2.ssf`:

```
wsech2.ssf <- function(X,z){
```

```
  # This subprogram evaluates the implicit form of the two equations
  # relating the sine and cosine moments of order n = X[nx*(i-1)+1]
  # for the mixed wrapped sech distribution, the parameters of which
  # are in z[1:5], to those derived from the data, which are supplied
  # in X[nx*(i-1)+2] (sine moment) and X[nx*(i-1)+3] (cosine moment).
  # It returns the squared norm of the vector of function values.
  #
  nf <- 2
  nx <- 3
  ndata <- length(X)/nx
  f <- vector(length=nf)
  x <- vector(length=nx)
  ssf <- 0
  for(i in 1:ndata){
    for(j in 1:nx){
      x[j] <- X[nx*(i-1)+j]
    }
  f <- wsech2.moments(x,z)
  ssf <- ssf+crossprod(f)
  }
}
```