DECONVOLUTION PAR LA METHODE DE MAXIMUM D'ENTROPIE MULTIECHELLE

François Levrier et Mourad Hamidouche

3 Mars 1999

1 L'observation interférométrique

Ces rappels sont basés essentiellement sur les "proceedings of the third NRAO Synthesis Imaging Summer School, June 1988" [1, 2, 3, 6, 7, 9].

1.1 La propagation de l'information

L'information sur un phénomène astrophysique se produisant en $\mathbf{R}$ est obtenue en mesurant le champ électromagnétique résultant en $\mathbf{r}$. Pour simplifier les équations, nous ne tiendrons compte que du champ électrique $\mathbf{E}(\mathbf{R}, t)$, étant donné que le champ magnétique peut s'en déduire à partir de la théorie de Maxwell. De plus, nous ne considérerons qu'une composante du champ. En raison de la linéarité des équations de Maxwell, nous pouvons également nous épargner les difficultés d'un champ variant dans le temps $E(\mathbf{r}, t)$ en ne considérant que ses composantes quasi-monochromatiques complexes $E_\nu(\mathbf{r})$

$$E(\mathbf{r}, t) = \frac{1}{\sqrt{2\pi}} \int_0^\infty E_\nu(\mathbf{r}) e^{2i\pi\nu t} d\nu. \tag{1}$$

Par ailleurs, cette même linéarité permet d'appliquer un théorème de superposition

$$E_\nu(\mathbf{r}) = \iiint \mathcal{P}_\nu(\mathbf{R}, \mathbf{r}) E_\nu(\mathbf{R}) d^3\mathbf{R}, \tag{2}$$

où $\mathcal{P}_\nu(\mathbf{R}, \mathbf{r})$ est la fonction de propagation. Vu les distances des sources astronomiques, nous ne pouvons espérer décrire leur structure dans la troisième dimension. Plus précisément, nous ne pouvons mesurer que leur distribution de brillance superficielle

$$\mathcal{E}_\nu(\mathbf{n}) = \int E_\nu(\mathbf{R}) d|\mathbf{R}|, \tag{3}$$

où l'intégration est effectuée le long de la ligne de visée de vecteur unitaire $\mathbf{n} = \mathbf{R}/|\mathbf{R}|$.
Il n'est pas inutile de se représenter une source comme projetée sur une "sphère céleste" de rayon $|\mathbf{R}|$. Si nous supposons qu'il n'y a pas d'autre source d'émission dans le volume que cette sphère délimite, le principe d'Huyghens-Fresnel donne une forme simple pour la fonction de propagation, et (2) devient

$$E_\nu(\mathbf{r}) = \iint \mathcal{E}_\nu(\mathbf{n}) \mathcal{A}_\nu(\mathbf{n}) \frac{e^{2i\pi\nu|\mathbf{R}-\mathbf{r}|/c}}{|\mathbf{R}-\mathbf{r}|} dS. \tag{4}$$

Dans cette équation, $\mathcal{A}_\nu(\mathbf{R})$, appelé lobe principal, prend en compte le fait que les antennes ne sont pas des sondes ponctuelles, mais qu'elles ont une sensibilité dépendant de la direction. Il s'agit donc d'une fonction valant un au voisinage immédiat d'une certaine direction principale et tombant rapidement à zéro ailleurs. Nous supposerons que toutes les antennes d'un même réseau ont le même lobe principal.

1.2 La fonction de cohérence spatiale

L'objet d'un interféromètre est de mesurer la corréraltion entre les champs en deux points distincts, disons $\mathbf{r_1}$ et $\mathbf{r_2}$, au travers de la fonction de cohérence spatiale ou fonction de visibilité

$$V(\mathbf{r_1}, \mathbf{r_2}) =< E_\nu(\mathbf{r_1}) E_\nu^*(\mathbf{r_2}) > . \tag{5}$$

Ici, $< . >$ est une espérance statistique et * indique la conjugaison complexe. D'après (4), et en supposant que les différents points de la source sont incohérents,

$$V(\mathbf{r_1}, \mathbf{r_2}) = \int \left\langle |\mathcal{E}_\nu(\mathbf{n})|^2 \right\rangle |\mathbf{R}|^2 \mathcal{A}_\nu^2(\mathbf{n}) \frac{e^{2i\pi\nu|\mathbf{R}-\mathbf{r_1}|/c}}{|\mathbf{R}-\mathbf{r_1}|} \frac{e^{-2i\pi\nu|\mathbf{R}-\mathbf{r_2}|/c}}{|\mathbf{R}-\mathbf{r_2}|} dS. \tag{6}$$

Enfin, puisque $|\mathbf{r}| \ll |\mathbf{R}|$, nous obtenons une expression très simple de $V(\mathbf{r_1}, \mathbf{r_2})$, qui fait apparaître l'intensité observée $I(\mathbf{n}) = \left\langle |\mathcal{E}_\nu(\mathbf{n})|^2 \right\rangle |\mathbf{R}|^2$

$$V(\mathbf{r_1}, \mathbf{r_2}) = \int I(\mathbf{n}) {\mathcal{A}_\nu}^2(\mathbf{n}) e^{-2i\pi\nu \mathbf{n}.(\mathbf{r_1}-\mathbf{r_2})/c} d\Omega. \tag{7}$$

Le point important à noter ici, c'est que $V(\mathbf{r_1}, \mathbf{r_2})$ ne dépend pas de $\mathbf{r_1}$ et $\mathbf{r_2}$ indépendamment, mais uniquement de la ligne de base $\mathbf{r_1} - \mathbf{r_2}$. De plus, puisque la correction du lobe principal est une simple division qui peut s'effectuer à la fin du processus de réduction des données, nous allons simplifier nos notations en supposant

$$I(\mathbf{n}) = I(\mathbf{n}) {\mathcal{A}_\nu}^2(\mathbf{n}). \tag{8}$$

1.3 Synthèse d'ouverture et transformées de Fourier

Bien entendu, ce qui va nous intéresser maintenant, c'est l'inversion de l'équation (7) en vue d'obtenir l'intensité observée $I(\mathbf{n})$. Ceci peut être fait dans un cas particulier, dans lequel nous nous plaçerons dorénavant. Supposons que la source que nous souhaitons observer soit limitée à une petite portion du ciel autour de $\mathbf{n_0}$. Alors, $\mathbf{n} = \mathbf{n_0} + \delta\mathbf{n}$ avec $|\delta\mathbf{n}| \ll 1 = |\mathbf{n_0}|$, et il est évident que $\delta\mathbf{n}.\mathbf{n_0} = 0$. Dans un système de coordonnées approprié, $\mathbf{n_0} = (0, 0, 1)$ et $\mathbf{r_1} - \mathbf{r_2} = \lambda(u, v, w)$ (λ étant la longueur d'onde). L'équation (7) devient

$$V'(u, v, w) = e^{-2i\pi w} \iint I(l, m) e^{-2i\pi(ul+vm)} dl dm, \tag{9}$$

où l et m sont les coordonnées célestes vers lesquelles pointe $\mathbf{n}$. Il est habituel d'intégrer le facteur exponentiel dans la fonction de visibilité, de sorte à pouvoir considérer une fonction de deux variables seulement (opération de blocage des franges)

$$V(u, v) = e^{2i\pi w} V'(u, v, w) = \iint I(l, m) e^{-2i\pi(ul+vm)} dl dm. \tag{10}$$

Il s'agit là d'une simple relation de transformée de Fourier, et nous avons

$$I(l, m) = \iint V(u, v) e^{2i\pi(ul+vm)} du dv. \tag{11}$$

1.4 Le problème de la déconvolution

Malheureusement, la fonction de cohérence spatiale n'est jamais connue partour dans le plan $u - v$, mais uniquement en certains points, puisque nous ne disposons que d'un nombre fini d'antennes et d'un temps

d'observation limité. Nous pouvons modéliser cet effet en introduisant une fonction de transfert optique ou fonction d'ouverture $P(u,v)$ qui est non-nulle uniquement aux points où une mesure a été effectuée.

$$P(u,v) = \sum_k D_k \delta(u-u_k, v-v_k). \tag{12}$$

Par ailleurs, les mesures n'étant jamais parfaites, il nous faut considérer qu'elles sont entachées d'un bruit $N(u,v)$. Si maintenant nous inversons l'équation comme précédemment, nous n'obtenons pas la vraie intensité $I(l,m)$, mais ce que les radioastronomes appellent l'image sale :

$$O(l,m) = \iint (V(u,v) + N(u,v))P(u,v)e^{2i\pi(ul+vm)}dudv. \tag{13}$$

Introduisant le lobe sale, ou fonction d'étalement de point (PSF, pour "point-spread function" en anglais)

$$B(l,m) = \iint P(u,v)e^{2i\pi(ul+vm)}dudv, \tag{14}$$

il est clair que nous avons une relation de convolution :

$$O = (I + \hat{N}) * B. \tag{15}$$

où $\hat{N}$ est la transformée de Fourier de N. Cette reconstruction O par transformée de Fourier présente de nombreux artefacts liés à la forme de B, qui peuvent rendre l'image totalement illisible et conduire à des effets non physiques (par exemple des intensités négatives). L'artefact le plus courant est l'oscillation produite par l'étendue limitée de B dans le plan $u-v$. Les méthodes de déconvolution usuelles introduisent des hypothèses qui permettent de donner des valeurs à O là où B est nul. La simple transformée de Fourier met à zéro $O(l,m)$ là où aucune mesure n'a été faite, et c'est ce qui provoque les artefacts sur les images. Les méthodes de maximum d'entropie font une autre hypothese.

2 Les méthodes de maximum d'entropie

2.1 Le principe

Comme nous venons de le voir, le problème des observations astronomiques, c'est qu'elles sont nécessairement incomplètes. Aux endroits du plan de Fourier $u-v$ où P est nulle, nous allons devoir "inventer" des mesures afin de reconstruire une image propre.
Le principe qui est à la base des méthodes de maximum d'entropie (MEM) consiste à supposer qu'en ces points, toutes les valeurs sont également probables. Bien entendu, celà est arbitraire et il existe d'autres méthodes, telles que CLEAN, partant d'autres points de vue.
En d'autres termes, un algorithme MEM va rechercher, parmi toutes les images reconstruites possibles, celle qui "rajoutera le moins d'information", tout en restant compatible avec l'image de départ, c'est-à-dire que, passée au travers du filtre P, elle s'en écarte de moins que le bruit $\hat{N}$.
Il devient sans doute plus clair maintenant pourquoi ces méthodes sont appelées "maximum d'entropie": Dans l'espace des images reconstruites possibles, celle que nous recherchons, celle qui rajoute le moins d'information, c'est celle qui maximise une fonction S, de type entropique.
Comment prendre en compte la contrainte selon laquelle l'image reconstruite, filtrée, doit redonner "à peu près" l'observation ? Pour le voir, il est intéressant de considérer le théorème de Bayes. Celui ci fournit la probabilité d'une image I, étant donnée une observation O:

$$P(I|O) = \frac{P(O|I).P(I)}{P(O)} \tag{16}$$

La vraisemblance $P(O|I)$ est précisément une estimation du bruit "nécessaire" au passage de I à O. S'il s'agit d'un bruit gaussien de variance σ_I, on peut écrire

$$P(O|I) = \exp\left(-\sum_{pixels} \frac{(O - B * I)^2}{2\sigma_I^2}\right) = \exp\left(-\chi^2(I)\right) \tag{17}$$

Quant à l'entropie, elle intervient naturellement dans la probabilité a priori $P(I)$ sous la forme

$$P(I) = \exp\left(\alpha S(I)\right) \tag{18}$$

Comme $P(I|O) \propto P(O|I)P(I)$, il devient alors (presque) évident que les algorithmes de maximum d'entropie vont avoir pour tâche de minimiser la fonction $J(I)$:

$$J(I) = \chi^2(I) - \alpha S(I), \tag{19}$$

le paramètre α permettant d'ajuster les poids relatifs de la régularisation (S) et de la fidélité (χ^2).
Les diverses formes qui ont été envisagées pour la fonction S sont résumées par Pantin & Starck [5]:

$$S(I) = -\sum_{pixels} \ln(I) \tag{20}$$

$$S(I) = -\sum_{pixels} I \ln(I) \tag{21}$$

$$S(I) = \sum_{pixels} I - m - I \ln(I/m) \tag{22}$$

Les méthodes de MEM restent peu employées. Elles rencontrent des problèmes pour restaurer des images qui comportent des échelles très différentes. On a pu montrer (revue par Narayan [4]) que MEM ne fonctionne bien qu'en présence d'un objet formé de quelques sources ponctuelles. Il semble que dans ce cas, c'est surtout le critère de positivité de la restauration qui permet de gagner en pouvoir séparateur. Dans les autres cas, on obtient peu d'amélioration, ou même un résultat très inférieur à la simple TF. Le choix du paramètre α (19) se révèle très délicat.

2.2 Les ondelettes

Dans l'article de Pantin & Starck [5], le principe du maximum d'entropie est appliqué à une image en distinguant différentes plages d'échelles spatiales. Celà devrait a priori permettre d'améliorer les résultats des algorithmes classiques, lesquels peinent parfois face à des données s'étalant sur une large gamme d'échelles. Les ondelettes sont l'outil mathématique qui se révèle approprié à une telle décomposition du signal. Très grossièrement, les ondelettes sont une généralisation de la décomposition de Fourier. Typiquement, une image I pourra s'écrire

$$I = \sum_{k=1}^{n} I^{(k)}. \tag{23}$$

Il faut noter que cette expression n'est rigoureusement valable que si l'image d'origine I ne contient que des échelles représentées dans les n ondelettes. En effet, les composantes $I^{(k)}$ sont le résultat d'une convolution de I par des fonctions "lissantes" $W^{(k)}$

$$I^{(k)} = I * W^{(k)} \tag{24}$$

de sorte qu'on a en toute rigueur[1]

$$\sum_{k=1}^{n} I^{(k)} = I * (\sum_{k=1}^{n} W^{(k)}) = I * W. \tag{25}$$

Dans notre travail, nous nous arrangerons pour que W soit le plus proche possible du lobe B.

2.3 L'entropie multiéchelle

Utilisant la décomposition en ondelettes que nous venons de présenter, Pantin & Starck [5] proposent une forme "multiéchelle" du principe de maximum d'entropie, en posant

$$S(I) = \frac{1}{\sigma_I} \sum_{k=1}^{n} \sum_{pixels} A^{(k)} \sigma^{(k)} (I^{(k)} - m^{(k)} - |I^{(k)}| \ln \frac{|I^{(k)}|}{m^{(k)}}) \tag{26}$$

Dans cette expression, les $m^{(k)}$ sont des images a priori, et on les prendra uniformes avec pour valeurs les $\sigma^{(k)}$, lesquels sont des bruits aux différentes échelles, dont on donnera dans la section suivante une méthode d'estimation. Ces valeurs permettent d'en déduire un cube de booléens appelé support multirésolution

$$M^{(k)}(x,y) = \begin{cases} 1 & \text{si } I^{(k)}(x,y) \geq 3\sigma^{(k)} \\ 0 & \text{si } I^{(k)}(x,y) < 3\sigma^{(k)} \end{cases} \tag{27}$$

Les fonctions $A^{(k)}$ sont alors simplement reliées au support multirésolution par $A^{(k)} = 1 - M^{(k)}$. L'idée derrière l'utilisation de ce procédé est de dire que la régularisation effectuée par l'entropie ne portera que sur le points pour lesquels $A^{(k)}$ vaut un, c'est-à-dire là où il n'y a a priori que peu de signal par rapport au niveau du bruit. On garde ainsi intactes les structures qui apparaissent à chaque échelle k bien au dessus du seuil correspondant $3\sigma^{(k)}$.

2.4 Et dans la pratique ?

Partant d'une image I ressemblant très grossièrement à celle que nous souhaitons obtenir, par exemple la composante floue de l'observation, nous pouvons trouver une image sur laquelle la fonction J atteint un minimum en utilisant une simple méthode de gradient.
L'image I^{n+1} à l'itération $n+1$ se déduisant de I^n par

$$I^{n+1} = I^n - \gamma \nabla(J(I^n)), \tag{28}$$

Intuitivement, on comprend bien qu'en procédant ainsi nous obtenons des images pour lesquelles J est de plus en plus petit. Lorsque I^n ne varie pratiquement plus, c'est que le gradient est nul et nous avons atteint un minimum. Il faut bien noter qu'il n'y a pas nécessairement de minimum unique, et rien ne nous assure que nous trouverons toujours le "bon". Empiriquement, si l'image de départ n'est pas trop irréaliste, on conçoit que celà puisse donner un résultat convaincant. Le calcul explicite de l'image "gradient" de J sera donnée dans la section suivante.
Il existe d'autres méthodes de recherche d'extremum d'une fonction de plusieurs variables, comme par exemple l'algorithme dit "recuit-simulé" qui se base sur le comportement d'un réseau de spins. Elles sont beaucoup plus lourdes à mettre en oeuvre, et nous nous sommes limités ici, comme Pantin et Starck, à la méthode du gradient.

[1] Tout ceci se conçoit en fait beaucoup mieux dans l'espace de Fourier, où les $W^{(k)}$ ont des transformées dont le support compact représente exactement les échelles conservées et où la convolution se ramène à une simple multiplication.

3 Les algorithmes

3.1 Le modèle

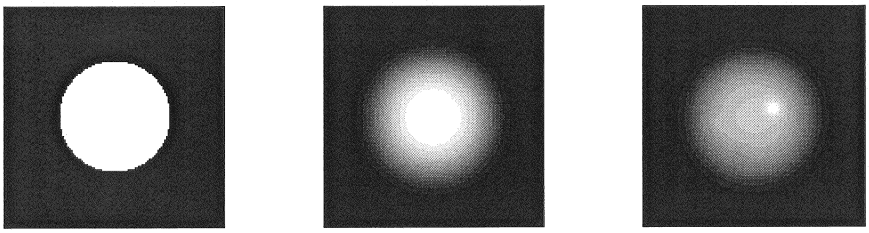

Figure 1: De droite à gauche: le disque uniforme de départ, le disque convolué par une gaussienne, et le modèle final

Nous nous sommes donnés comme but de reconstruire des images astronomiques, et pour tester nos méthodes nous avons choisi de considérer un modèle d'image solaire. Ce choix peut s'expliquer en partie par une volonté d'avoir une image ni trop simple (source ponctuelle), ni inutilement complexe (galaxie). Le modèle présenté est donc construit en prenant un disque uniforme

$$D(x,y) = \begin{cases} 1 & \text{si } x^2+y^2 \leq R^2 \\ 0 & \text{si } x^2+y^2 > R^2 \end{cases} \tag{29}$$

On le convolue par une gaussienne de façon à éliminer la discontinuité des bords, et on normalise le résultat:

$$D_s = \frac{1}{\int G}(D * G). \tag{30}$$

Enfin on rajoute des gaussiennes quasi ponctuelles:

$$M(x,y) = D_s(x,y) + \sum_{i=1}^{p} A_p \exp\left\{-\frac{(x-x_p)^2+(y-y_p)^2}{2\sigma_p^2}\right\}. \tag{31}$$

Ce modèle correspond assez bien à l'émission radio de la couronne solaire. Il est particulièrement résistant aux algorithmes de déconvolution habituels.

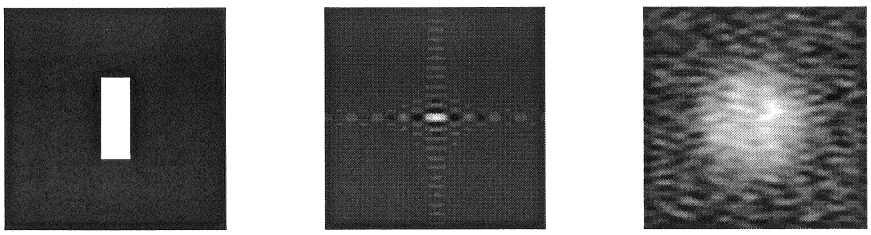

Figure 2: De droite à gauche: La fonction d'ouverture, la fonction d'étalement de point (PSF) et l'observation simulée de notre modèle.

Pour obtenir une observation simulée, nous passons ensuite cette image au travers d'une FTO très simple puisqu'elle est représentée dans l'espace de Fourier par un rectangle de "1" s'étalant de -8 à +8 en x et de

-23 à +23 en y. C'est celle qui correspond à un réseau d'antennes en T (par exemple le radiohéliographe de Nançay) utilisé sans synthèse d'ouverture.Ce dernier cas conduit à des FTO plus complexes, et ne sera pas abordé ici. On ajoute simultanément un bruit gaussien. Il faut noter que pour les opéraions dans l'espace de Fourier, il est nécessaire de translater les tableaux. La fonction d'ouverture, par exemple, doit être centrée sur (0,0) alors que, par commodité, on l'a construite centrée en (64,64). Remarquons que, comme les objets observés sont réels , il n'y a besoin en toute rigueur que d'un demi plan pour la FTO. Nous en avons gardé la totalité, car le programme de FFT fourni est uniquement complexe - complexe.

3.2 Les filtres

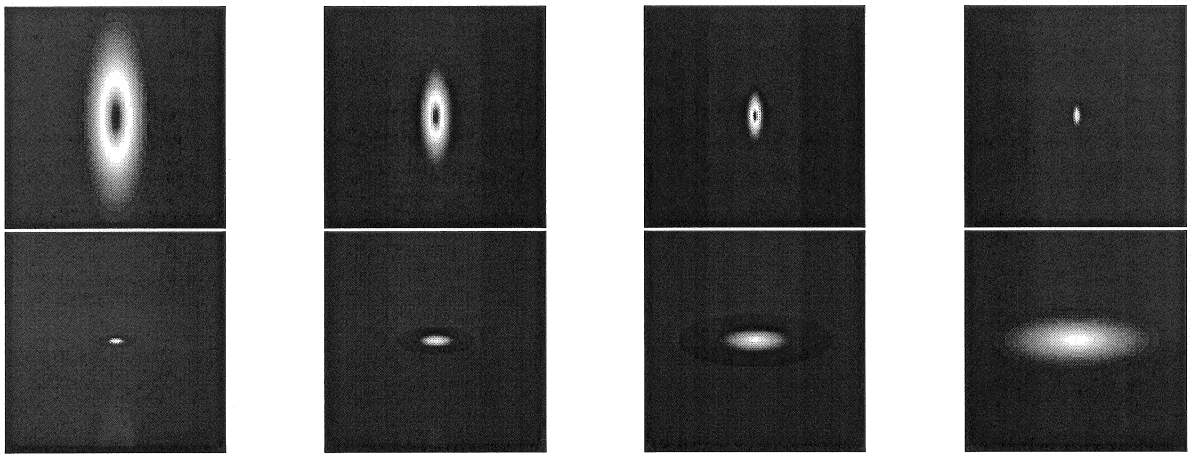

Figure 3: Les filtres utilisés dans notre expérience. De droite à gauche, on sélectionne les échelles spatiales de plus en plus grandes. En haut, les filtres en Fourier et en bas leurs contreparties dans l'espace réel

Nous avons vu que la décomposition en ondelettes supposait de prendre un ensemble de filtres $W^{(k)}$ dont la somme serait proche du lobe B. Travaillant dans l'espace de Fourier, celà revient à considérer des fonctions $F^{(k)}$ dont la somme serait proche de la fonction d'ouverture. Nous avons choisi de prendre des filtres annulaires gaussiens dont le rapport des axes serait égal à 8/23, comme c'est le cas pour celle-ci. Les filtres choisis sont inspirés de la transformée ondelette "à trous" (Starck et Bijaoui 1995). Pour garder une forme proche de la FTO dans le plan $u-v$, ils sont générés à partir d'une fonction qui présente la même anisotropie. Nous avons choisi d'utiliser le lobe propre au sens de l'algorithme CLEAN, c'est à dire une fonction gaussienne dont les largeurs à $1/e$ sont les dimensions du rectangle FTO. Si $LP(u,v)$ est ce lobe propre, les différents filtres sont

$$F^{(k)}(u,v) = LP(2^{k-1}u, 2^{k-1}v) - LP(2^k u, 2^k v). \tag{32}$$

Le dernier filtre est uniquement passe bas

$$F^{(n)}(u,v) = LP(2^n u, 2^n v). \tag{33}$$

3.3 L'estimation du bruit

L'expression de l'entropie multiéchelles fait apparaître des bruits aux différentes échelles $\sigma^{(k)}$, que nous avons choisi d'estimer en utilisant la méthode décrite par Starck et Murtagh [8]. Le principe en est relativement simple et consiste à étudier le comportement typique du bruit à chaque échelle. Pour celà, on construit une image entièrement bruitée avec une variance de 1, dont on calcule la transformée en ondelettes. On obtient

donc un "bruit caractéristique" $\sigma_e^{(k)}$ pour chacune de nos échelles. Les $\sigma^{(k)}$ se déduisent alors du bruit σ_I sur l'image d'origine I, en utilisant une propriété des ondelettes selon laquelle on a $\sigma^{(k)} = \sigma_e^{(k)}\sigma_I$. Il reste évidemment à déterminer le bruit σ_I, lequel peut être estimé sur l'ensemble des pixels de l'image "qui ne contiennent que du bruit". Cette assertion est mise en forme en utilisant le support multirésolution déjà défini. A partir de I et d'une estimation initiale σ_I^0 du bruit sur celle-ci, on calcule la transformée ondelettes puis le support multirésolution $\{M^{(k)}\}$. L'ensemble $\mathcal{N}$ des pixels (x, y) pour lesquels tous les $M^{(k)}$ sont nuls sont supposés ne contenir aucun signal, et on n'utilise qu'eux pour calculer une nouvelle estimation du bruit, σ_I^1. On en déduit un nouveau support multirésolution, donc un nouvel ensemble $\mathcal{N}$, et ainsi de suite jusqu'à ce qu'il y ait convergence de la suite des σ_I^n.

3.4 L'algorithme MEM

Comme nous l'avons dit plus haut, l'application pratique de l'algorithme de maximum d'entropie multi-échelles requiert de calculer "l'image gradient" de J en I. Cette image, que nous noterons $\nabla(J(I))$ a pour éléments

$$\nabla(J(I))_{lm} = \frac{\partial J}{\partial I_{lm}} \tag{34}$$

Or, avec la discrétisation, les expressions deviennent

$$J(I) = \frac{1}{2\sigma_I^2}\sum_{ij}(O_{ij} - \sum_{rs} B_{rs}I_{i-rj-s})^2 - \alpha S(I) \tag{35}$$

et pour l'entropie

$$S(I) = \frac{1}{\sigma_I}\sum_{k=1}^{n}\sum_{ij} A_{ij}^{(k)}\sigma^{(k)}(\sum_{rs} W_{rs}^{(k)}I_{i-rj-s} - m^{(k)} - |\sum_{rs} W_{rs}^{(k)}I_{i-rj-s}| \ln\frac{|\sum_{rs} W_{rs}^{(k)}I_{i-rj-s}|}{m^{(k)}}) \tag{36}$$

La dérivation de ces expressions se fait sans difficulté majeure et on trouve que $\nabla(J(I))_{lm}$ s'écrit simplement[2]:

$$\nabla(J(I))_{lm} = -\frac{1}{\sigma_I^2}\sum_{ij}((O_{ij} - \sum_{rs} B_{rs}I_{i-rj-s})B_{i-lj-m}) - \alpha\nabla(S(I))_{lm} \tag{37}$$

$$\nabla(S(I))_{lm} = \frac{1}{\sigma_I}\sum_{k=1}^{n}\sum_{ij}[A_{ij}^{(k)}\sigma^{(k)}(1 - sgn(\sum_{rs} W_{rs}^{(k)}I_{i-rj-s})(1 + \ln\frac{|\sum_{rs} W_{rs}^{(k)}I_{i-rj-s}|}{m^{(k)}}))W_{i-lj-m}^{(k)}] \tag{38}$$

On peut condenser ces résultats de manière à obtenir une expression plus lisible, en posant

$$B_{ij}^* = B_{-i-j} \quad \text{et} \quad W_{ij}^{*(k)} = W_{-i-j}^{(k)} \tag{39}$$

Alors on peut écrire l'image gradient de J sous la forme simplifiée:

$$\nabla(J(I)) = -\frac{1}{\sigma_I^2}(O - I * B) * B^* - \frac{\alpha}{\sigma_I}\sum_{k=1}^{n}[A^{(k)}\sigma^{(k)}(1 - sgn(I^{(k)})(1 + \ln\frac{|I^{(k)}|}{m^{(k)}}))] * W^{*(k)} \tag{40}$$

C'est cette forme du gradient que nous avons décidé d'implémenter dans nos expériences numériques.

[2]Remarquons que l'expression donnée dans [5] contient une légère erreur dans cette dérivation...

4 Les résultats

4.1 L'estimation du bruit

Dans leur article [8], Starck & Murtagh semblent très confiants en ce qui concerne l'algorithme d'estimation automatique du bruit à partir du support multirésolution, et en particulier sur sa "robustness", c'est-à-dire sa fiabilité. Nous avons eu l'occasion de voir autre chose, en vérité. Nous présentons sur la figure le résultat d'un test effectué sur notre simulation. Le graphique représente le bruit déduit de leur algorithme en fonction de celui que nous avons introduit - et qui est bien sûr connu. On voit que la corrélation est plutôt bonne, même s'il y a une légère sous estimation systématique. Celà semble donc indiquer que nous avons correctement implémenté leur algorithme et nous ne mettons donc pas en cause la valeur de celui-ci... quand il converge! En effet, ce que nous avons constaté, c'est que dans certaines conditions (sur l'image entièrement bruitée, sur le niveau de bruit dans l'image, etc...) que nous n'avons malheureusement pas pu déterminer précisément, faute de temps, il n'y a pas convergence. En réalité nous sommes assez régulièrement tombés sur des cycles d'ordre 2, 3 ou plus, sans que nous puissions en trouver la raison. Dans la suite, nous n'avons bien entendu utilisé que des résultats convergents.

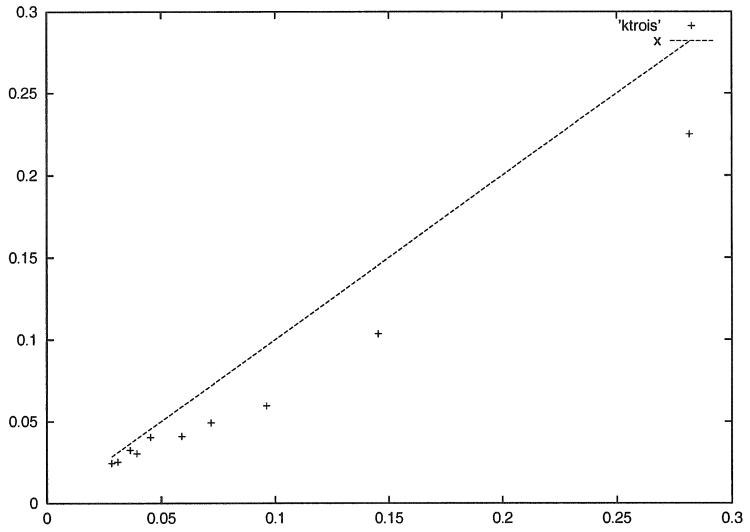

Figure 4: Résultats d'un test de l'algorithme d'estimation du bruit. On a porté en abscisse le bruit introduit et en ordonnées le bruit estimé. La droite $y = x$ est là pour référence.

4.2 Le maximum d'entropie

Nous avons tenté d'appliquer notre algorithme au modèle construit, en partant d'une image initiale quelconque, à savoir la composante "smooth" de l'observation. Pour être honnêtes, nous devons admettre que la méthode ne donne pas de bons résultats. En fait, lorsque nous prenons une petite valeur de α - à la limite nulle, elle reconstruit l'observation, bruit compris, ce qui n'est en soi pas étonnant puisqu'alors on cherche à minimiser le premier terme de J, qui n'est autre qu'un χ^2 (voir figures 5 et 6). Cette constatation rassurante faite, les choses se compliquent pourtant quand on introduit la contribution entropique au gradient de J. La régularisation qui est censée se produire n'a pas lieu: On observe, bien au contraire, des variations sur l'image lorsque α devient suffisamment grand pour ne plus être négligeable. Cependant, si l'on efface le terme en χ^2, et qu'on calcule la valeur de l'entropie sur l'image à chaque itération, celle-ci augmente bien, conformément à nos attentes, mais l'image qu'on obtient - au lieu d'être uniforme comme elle le devrait -

présente une structure extrêmement curieuse, dont nous n'avons pu déterminer l'origine. On a donc bien un algorithme qui maximise l'entropie telle qu'elle a été définie...mais celà ne régularise en rien l'image. Nous donnons ci-dessous les résultats de certaines de nos expériences.

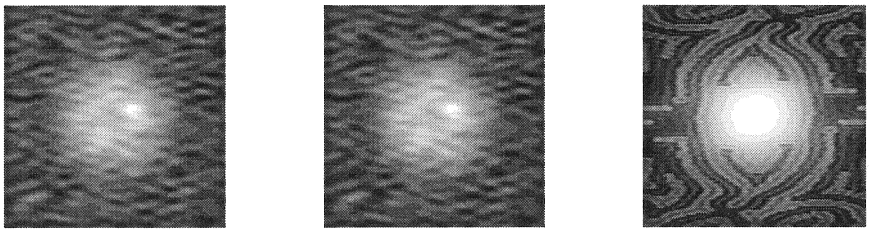

Figure 5: A gauche, l'observation telle que donnée par obs.pro. Au centre, la "reconstruction" de l'image pour $\alpha = 0$, c'est-à-dire quand seul le terme en χ^2 est considéré. A droite, la situation inverse, lorsque seul le terme entropique est pris en compte. Il est clair que si la contrainte est respectée, la régularisation manque à l'appel...

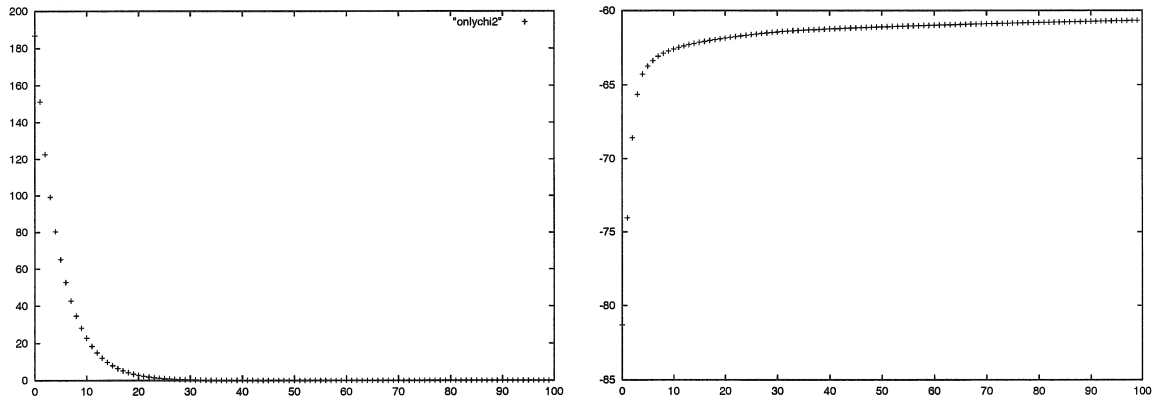

Figure 6: Evolution des termes conservés (χ^2 à gauche et entropie à droite) dans les deux expériences de la figure précédente, en fonction du temps.

Nous avons également testé la situation extrêmement simplifiée d'une seule source ponctuelle. Après avoir pris en compte un critère de positivité sur l'image reconstruite, en l'occurence en seuillant les itérations de l'algorithme par un ϵ, on obtient un résultat qui est cette fois satisfaisant, comme le montre la figure 8.

4.3 Conclusions

De ce que nous avons vu, il n'y a donc pas grand chose de positif à garder. Les détracteurs des méthodes de maximum d'entropie affirment que celles-ci au mieux ne font rien et au pire détériorent l'image. Nous ne sommes pas loin de penser la même chose. Il semble cependant assez clair (voir la figure 6) que nous avons effectivement construit un algorithme, basé sur la décomposition en ondelettes, maximisant la fonctionnelle définie par Pantin et Starck. Il se trouve que nous ne trouvons pas leurs résultats. Il nous paraît que cet algorithme de MEM modifié ne permet toujours pas de traiter de façon satisfaisante des objets correspondants au modèle traité, dont la particularité essentielle est d'avoir la quasi totalité de son flux concentré dans une source très étendue et qui n'a donc que très peu de points non nuls dans le plan $u - v$.

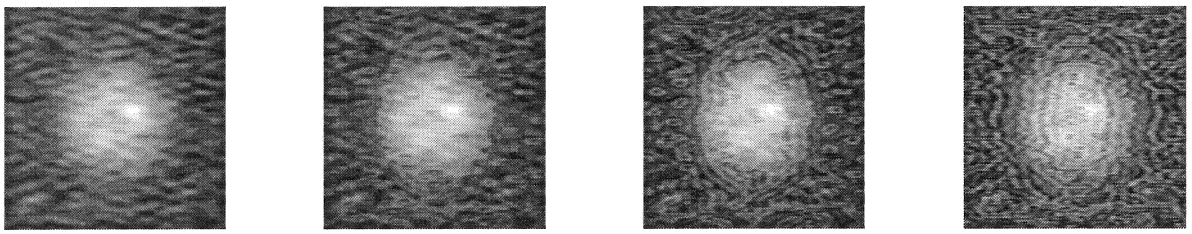

Figure 7: Images "reconstruites" pour différentes valeurs de α: de gauche à droite, $\alpha = 0.5$, $\alpha = 3$, $\alpha = 8$, et $\alpha = 20$.

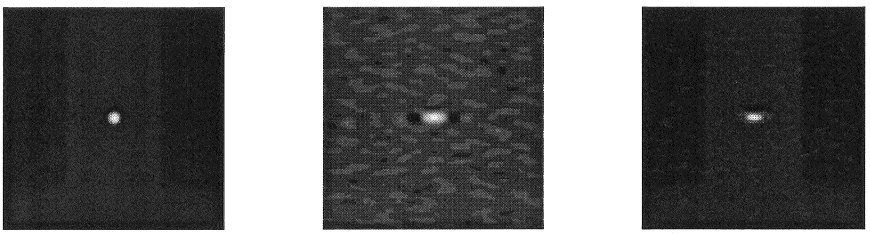

Figure 8: Résultats de l'algorithme pour une source ponctuelle. De droite à gauche: le modèle, l'observation, et l'image reconstruite.

Bien entendu, si l'on nous donne le droit de critiquer les articles sur lesquels nous nous sommes basés, il convient de rester prudents car il est toujours possible - voire probable - que nous nous soyons fourvoyés dans l'implémentation et que nous ne nous en soyions jamais rendus compte. Pour vérification, les codes IDL de nos programmes sont donnés en annexe.

References

[1] Clark, B.G. in *Synthesis Imaging In Radio Astronomy*, A.S.P.C.S., **6**, 1989

[2] Cornwell, T.J., & Braun, R. in *Synthesis Imaging In Radio Astronomy*, A.S.P.C.S., **6**, 1989

[3] Fomalont, E.B., & Perley, R.A. in *Synthesis Imaging In Radio Astronomy*, A.S.P.C.S., **6**, 1989

[4] Narayan, R., & Nityananda, R. *Ann. Rev. Astron. Astrophys.*, **24**, 127, 1986

[5] Pantin, E., & Starck, J.-L. A&A Supplement Series, **118**, 575, 1996

[6] Roelfsema, P. in *Synthesis Imaging In Radio Astronomy*, A.S.P.C.S., **6**, 1989

[7] Sramek, R.A., & Schwab, F.R. in *Synthesis Imaging In Radio Astronomy*, A.S.P.C.S., **6**, 1989

[8] Starck, J.-L., & Murtagh, F. P.A.S.P., **110**, 193, 1998

[9] Thompson, A.R., in *Synthesis Imaging In Radio Astronomy*, A.S.P.C.S., **6**, 1989

Annexe: Codes IDL

Nous présentons ici les codes que nous avons écrits en IDL. Il sera ainsi plus facile de débusquer nos fautes et de nous condamner aux galères pour les expier...

Fichier disque.pro, retournant le modèle solaire

```
Function disque,N

R=32L
F=3
p=3

objet=dblarr(N,N)
objetF=dblarr(N,N)
gauss=dblarr(N,N)
gaussF=dblarr(N,N)
objetsmouze=dblarr(N,N)
objetsmouzeF=dblarr(N,N)
aux=dblarr(N,N)
ponct=dblarr(N,N)
A=dblarr(7)
B=dblarr(4*p)

;Modelisation d'un objet

for i = N/2-R, N/2+R do begin
   for j= N/2-R, N/2+R do begin
      Rd=double(R)
      delta2=(i-N/2)^2+(j-N/2)^2
      if (double(delta2) lt Rd^2) then begin
        objet(i,j)=1
      endif
   endfor
endfor

;Construction d'une gaussienne circulaire de largeur 16

A=[0,1,9,9,N/2-1,N/2-1,0]
X= FINDGEN(N) # REPLICATE(1.0,N)
Y= REPLICATE(1.0,N) # FINDGEN(N)
U= ((X-A[4])/A[2])^2 + ((Y-A[5])/A[3])^2
gauss= A[0] + A[1] * EXP(-double(U)/2)

;Transformee de Fourier

objetF=FFT(objet,-1)
gaussF=FFT(gauss,-1)
```

```
;Produit

objetsmouzeF=objetF*gaussF

;Transformee inverse

aux=FFT(objetsmouzeF,1)*N^2/20*3.14159265/(A[2]*A[3])
objetsmouze=SHIFT(aux,N/2,N/2)

;ajout de sources ponctuelles

B=[-0.1,R/6,55*N/128,42*N/128,0.35,R/8,75*N/128,68*N/128,0.1,R/7,40*N/128,59*N/128]
ponct(*,*)=0.D
for i=0,p-1 do begin
   X=FINDGEN(N) # REPLICATE(1.0, N)
   Y=REPLICATE(1.0, N) # FINDGEN(N)
   V=double(((X-B[4*i+2])/B[4*i+1])^2 + ((Y-B[4*i+3])/B[4*i+1])^2)
   ponct = ponct + B[4*i+0] * EXP(-V/2)
   modele=objetsmouze+ponct*objetsmouze
endfor
return,modele

end
```

Fichier obs.pro, retournant l'observation à partir du modèle

```
Function obs,modele

N=128L
R=32L
F=3

FOaux=dblarr(N,N)
FO=dblarr(N,N)
BruitReel=dblarr(N,N)
BruitImaginaire=dblarr(N,N)
BruitComplexe=complexarr(N,N)
observ=dblarr(N,N)
observF=dblarr(N,N)
modeleF=dblarr(N,N)

;construction de la fonction d'ouverture

for i=N/2-8,N/2+8 do begin
  for j=N/2-23,N/2+23 do begin
     FOaux(i,j)=1.D
  endfor
endfor
```

```
FO=shift(FOaux,-N/2,-N/2)

;bruit normal

BruitReel=randomn(64295759L,N,N)/(300)
BruitImaginaire=randomn(6429207L,N,N)/(300)
BruitComplexe=dcomplex(BruitReel,BruitImaginaire)

;Relation modele-observation

modeleF=FFT(modele,-1)
observF=FO*(modeleF+BruitComplexe)
observ=FFT(observF,1)
bruit=stdev(abs(FFT(FO*BruitComplexe,1)))
print,bruit
return,reelle(observ)

end
```

Fichier convo.pro, retournant le produit de convolution de deux tableaux

```
Function convo,tab1,tab2

N=128L

;Transformee de Fourier

tab1F=dblarr(N,N)
tab2F=dblarr(N,N)
tab1F=N*FFT(tab1,-1)
tab2F=N*FFT(shift(tab2,-N/2,-N/2),-1)

;Produit des TF

resF=dblarr(N,N)
resF=tab1F*tab2F

;Transformee inverse

res=dblarr(N,N)
res=FFT(resF,1)
return,reelle(res)

end
```

Fichier reelle.pro, retournant la partie réelle d'un tableau complexe

```
function reelle,tab

partiereelle=double(tab)
```

```
return,partiereelle

end
```

Fichier stdev.pro, retournant l'écart-type d'un tableau

```
function stdev,Image

N=128L
R=32L
F=4

  Imcarre=dblarr(N,N)
  Imcarre=Image^2
  sd=sqrt(double(total(Imcarre)/(N^2)-(total(Image)/(N^2))^2))
  return,sd

end
```

Fichier filter.pro, retournant les filtres utilisés dans la décomposition en ondelettes

```
function filter,N

F=3

A=dblarr(7)
gauss=dblarr(F+1,N,N)
fil=dblarr(F+1,N,N)
filtresh=dblarr(F+1,N,N)

A=[0,1,8.,23.,N/2,N/2,0]
X=FINDGEN(N) # REPLICATE(1.0,N)
Y=REPLICATE(1.0,N) # FINDGEN(N)

for k=0,F do begin
   U=((X-A[4])/(A[2]/(2.^k)))^2 + ((Y-A[5])/(A[3]/(2.^k)))^2
   gauss(k,*,*) = A[0] + A[1] * EXP(-double(U)/2)
endfor

for k=0,F-1 do begin
   fil(k,*,*)=gauss(k,*,*)-gauss(k+1,*,*)
endfor
fil(F,*,*)=gauss(F,*,*)
filtresh=shift(fil,0,-N/2,-N/2)
return,filtresh

end
```

Fichier decomp.pro, retournant la transformée ondelettes d'un tableau

```
function decomp,objet,filsh

N=128L
F=3

objetF=dblarr(N,N)
objetFmulti=dblarr(F+1,N,N)
objetmulti=dblarr(F+1,N,N)
objetFfiltre=dblarr(F+1,N,N)
objetfiltre=dblarr(F+1,N,N)
aux=dblarr(N,N)
auxF=dblarr(N,N)

objetF=FFT(objet,-1)
for k=0,F do begin
   objetFmulti(k,*,*)=objetF(*,*)
endfor

objetFfiltre=filsh*objetFmulti
for k=0,F do begin
   auxF(*,*)=objetFfiltre(k,*,*)
   aux=FFT(auxF,1)
   objetfiltre(k,*,*)=aux(*,*)
endfor

return,objetfiltre

end
```

Fichier onlynoise.pro, retournant une image bruitée de variance 1

```
function onlynoise,N

F=3

filtresh=filter(N)
simul=dblarr(N,N)
standdev=dblarr(F+1)
aux=dblarr(N,N)
simultrans=dblarr(F+1,N,N)

simul=randomn(5382719L,N,N)
simultrans=decomp(simul,filtresh)
for k=0,F do begin
   aux(*,*)=simultrans(k,*,*)
   standdev(k)=stdev(aux)
endfor

return,standdev
```

```
end
```

Fichier bruit.pro, retournant l'estimation des bruits à chaque échelle sur l'image

```
function bruit,filtresh,Image,simulnoise

N=128L
F=3
epsilon=2.D-3

;Definitions

Imsharp=dblarr(N,N)
Imagefiltre=dblarr(F+1,N,N)
Imagefiltre=decomp(Image,filtresh)
Sigma=dblarr(F+1)
Sigmulti=dblarr(F+1,N,N)
M=dblarr(F+1,N,N)
Int=dblarr(N,N)
zero=dblarr(N,N)
S=dblarr(N,N)
A=dblarr(N,N)

; calcul de l'image diminuee de la composante "smooth"

Imsharp(*,*)=Image(*,*)-Imagefiltre(F,*,*)

; Initialisation de l'estimation du bruit

SigmaI=0.D
SigmaInew=stdev(Imsharp)
Var=abs(SigmaInew-SigmaI)/SigmaInew

; Boucle d'estimation du bruit

While (Var GT epsilon) do begin
   SigmaI=SigmaInew
   Sigma=SigmaI*simulnoise

; Calcul du MRS

   for k=0,F do begin
         Sigmulti(k,*,*)=Sigma(k)
   endfor
   M(where(abs(Imagefiltre) GT 3*Sigmulti))=1.D

; Calcul de l'ensemble des pixels domines par le bruit a toutes les echelles
```

```
   for k=0,F-1 do begin
      Int(*,*)=Int(*,*)+((2^k)*M(k,*,*))
   endfor
   S(*,*)=1.D
   S(where(Int GT zero))=0.D
   p=total(S)

; Selection des pixels de bruit dans l'image et calcul du sigma

   A=Imsharp*S
   If (p NE 0) then begin
     Imcarre=A^2
     SigmaInew=sqrt(double((total(Imcarre)/p)-(total(A)/p)^2))
   endif
   print,SigmaInew
   Var=abs(SigmaInew-SigmaI)/SigmaInew
Endwhile

SigmaI=SigmaInew
Sigma=SigmaI*simulnoise
Sigma(0)=SigmaI

return,Sigma

end
```

Fichier grad.pro, retournant l'image gradient de J

```
function grad,Image,Noise,filtresh,observ,realPSF,alpha

N=128L
F=3

; Definitions

Imagefiltre=dblarr(F+1,N,N)
Imagefiltre=decomp(Image,filtresh)
A=dblarr(F+1,N,N)
M=dblarr(F+1,N,N)
Un=dblarr(F+1,N,N)
Sigmulti=dblarr(F+1,N,N)
gradient=dblarr(N,N)
terme1=dblarr(N,N)
interm=dblarr(N,N)
sgn=dblarr(N,N)
aux=dblarr(N,N)
auxF=dblarr(N,N)
sum=dblarr(N,N)
```

```
;calcul du MRS

for k=0,F do begin
   Sigmulti(k,*,*)=3*Noise(k)
endfor
M(where(abs(Imagefiltre) GT Sigmulti))=1.D

;calcul du premier terme du gradient

Un(*,*,*)=1.D
A=Un-M
interm=observ-convo(realPSF,Image)
terme1=-convo(realPSF,interm)
gradient=terme1
sum=0.5D*(interm^2)

;calcul des termes a chaque echelle

for k=0,F do begin
   for i=0,N-1 do begin
      for j=0,N-1 do begin
         if (Imagefiltre(k,i,j) NE 0) then begin
           sgn(i,j)=abs(Imagefiltre(k,i,j))/Imagefiltre(k,i,j)
         endif
      endfor
   endfor
   interm(*,*)=A(k,*,*)*Noise(k)*(1-sgn(*,*)*(1+ALOG(abs(Imagefiltre(k,*,*))/Noise(k))))
   auxF(*,*)=filtresh(k,*,*)
   aux=reelle(shift(FFT(auxF,1),-N/2,-N/2))/(N^2)
   gradient=gradient-(alpha)*(convo(interm,aux))
   sum(*,*)=sum(*,*)+A(k,*,*)*Noise(k)*(Imagefiltre(k,*,*)-(Noise(k))-abs(Imagefiltre(k,*,*))*
endfor
print,total(sum)

return,gradient

end
```

Fichier main.pro, programme principal

```
PRO main,observ

N=128L
F=3
eps=1.D-3

FOaux=dblarr(N,N)
FO=dblarr(N,N)
PSF=dblarr(N,N)
```

```
Im=dblarr(N,N)
Imnew=dblarr(N,N)
Imrec=dblarr(N,N)

alpha=0.5D
gamma=0.1D

k=0

for i=N/2-8,N/2+8 do begin
  for j=N/2-23,N/2+23 do begin
     FOaux(i,j)=1.D
  endfor
endfor
FO=shift(FOaux,-N/2,-N/2)
PSF=reelle(shift(FFT(FO,1),-N/2,-N/2))/(N^2)

filtresh=filter(N)
objetfiltre=dblarr(F+1,N,N)
objetfiltre=decomp(observ,filtresh)
simulnoise=onlynoise(N)
standdev=bruit(filtresh,observ,simulnoise)

Imnew(*,*)=objetfiltre(F,*,*)

while (k LT 100) do begin
  Im=Imnew
  tvscl,Im
  gr=grad(Im,standdev,filtresh,observ,PSF,alpha)
  Imnew=Im-gamma*gr
  k=k+1
for i=0,N-1 do begin
  for j=0,N-1 do begin
    if (Imnew(i,j) LT eps) then begin
       Imnew(i,j)=0.D
    endif
  endfor
endfor
endwhile
Imrec=Imnew

end
```