NOTES SUR LA DISPERSION SPECTRALE

Ariane LE CARDINAL, 30 mars 2021

1 Notations

Symbole	Description
ρ	fonction de pénalisation (Cauchy ou moindre carrés)
f	gaussienne
δ	carte des bad pixels (BP)
s^{cal}	facteur d'échelle
A	amplitude $\propto \frac{1}{2\pi\sigma^2}$
c_h^k	coefficients de la loi polynomiale (pixel)
c_h^λ	coefficients de la loi polynomiale (spectres)
x, y	position de la microlentille
xtrue, ytrue	paramètres vrais donnes a l'image simulée
xopt, yopt	paramètres optimisés
xinit, yinit	paramètres a l'initialisation
F(X)	valeur de la fonction de vraisemblance
‖G(X)‖	norme du gradient de la fonction de vraisemblance
$\lambda_{laser} = [\lambda_1, \lambda_2, \lambda_3, \lambda_4]$	tableau des longueurs d'onde de calibration
λ_0	moyenne de λ_{laser}

TABLE 1 – Notations

Le facteur d'échelle s^{cal} sert à ce que les interspectres ne soient pas détectés comme "outsiders" en diminuant la valeur d'ensemble des résidus. La fonction de pénalisation est ainsi robuste, car les pixels détectés comme défectueux sont très éloignés de la loi.

Le nombre d'acquisitions lasers n dépend du mode de calibration des bandes :

- YJ-mode : $n_\lambda^{cal} = 3$
- YH-mode : $n_\lambda^{cal} = 4$

Longueurs d'onde de calibration laser utilisées :

- $\lambda_1 = 987.72$ nm
- $\lambda_2 = 1123.71$ nm
- $\lambda_3 = 1309.37$ nm
- $\lambda_4 = 1545.10$ nm

2 Equations

On cherche à minimiser le terme de fidélité aux données $f(c_h^k, c_h^\lambda, \gamma_h, \Lambda)$ pour avoir une différence entre le modèle et les données la plus petite possible. Pour cela, on cherche les paramètres x, y, σ et A qui minimisent la fonction par ajustement.

Fonction de pénalisation : plusieurs choix sont possibles

- fonction de Cauchy : $\rho(r) = \frac{\gamma}{2} log(1 + \frac{r^2}{\gamma^2})$
  $\gamma = 2.385$

Modèle utilisé :

Gaussienne :
$f(k_1, k_2, \sigma) = Aexp(\frac{(x_0-x)^2+(y_0-y)^2}{2\sigma^2})$

Calibration :
Data fidelity term :
$L(c_h^k, c_h^\lambda, \gamma_h, \Lambda) = \sum_{k=1}^{b} \rho\delta_k^{cal}(s^{cal-}1(d_p - \sum_{n=1} \xi_k(k_1, k_2, \sigma)))$

Position selon n $\in [1, 4]$:
$x(\lambda_n) = c_0 + c_1(\lambda_1 - \lambda_0) + c_2(\lambda_n - \lambda_0)^2$
$y(\lambda_n) = c_0' + c_1'(\lambda_1 - \lambda_0) + c_2'(\lambda_n - \lambda_0)^2$

Coefficients polynomiaux :
C0 = [6.2 10.0 0.0; 25.0 80.0 0.0] : la première partie correspond a l'écartement en y de la loi spectrale, la seconde partie correspond a l'écartement en x.
6.2 : centre, 10 et 0 : écartement.

Normalisation par λ_0 :
normalisation par λ_0 pour que l'optimsateur mette moins de temps. Mieux conditionne.
A l'origine : problème dans les unités (λ_0 en micron, donc coefficients en micron^{-1}).
Pas d'optimisation : c_0 sans unité, c_1 en 10^{-6}, c_2 en 10^{-12}.
$c_0 + c_1(\lambda_1 - \lambda_0) + c_2(\lambda_n - \lambda_0)^2$
Désormais : c_0 en pixel
$c_0 + c_1(\frac{\lambda_1-\lambda_0}{\lambda_0}) + c_2(\frac{\lambda_n-\lambda_0}{\lambda_0})^2$

2.1 Modèle

modèle : $m_k = \sum_{i=1}^{3} \alpha g_i(\theta_i, \sigma_i)$ avec

- g lq gaussienne
- θ la position
- σ la largeur a mi hauteur
- α l'amplitude de la gaussienne a la position θ
- k les pixels
- i le nombre de spots

On cherche a minimiser la fonction de coût $f(\theta, \sigma) = \sum_k \| m_k - d_k \|^2$ avec d_k les données. Pour un pixel k, on minimise l'écart entre le modèle et les données.

Cas avec une seule gaussienne à fitter : Norme quadratique : $f = \|\alpha g - d\|_W^2$ fonction de coût pondérée par la matrice de poids W, où αg correspond au modèle m.
$f = [\alpha g - d]^T W[\alpha g - d] = \alpha^2 g^T W g - 2\alpha g^T W d + d^T W d$ f est une fonction quadratique dont le minimum est atteint quand la dérivée s'annule. On cherche l'amplitude de la gaussienne qui s'ajuste le mieux aux données, ainsi :
$\frac{\partial f}{\partial \alpha} = \alpha g^T W g - 2 g^T W d = 0$

$\alpha^+ = [g^T W g]^{-1} g^T W d$ est l'amplitude optimale. Le terme $g^T W d$ correpond à la corrélation entre la gaussienne g et les données d.
Avec l'optimiseur vmlmb, la minimisation de f se fait par rapport au gradient θ :
$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial \alpha(\theta)} \frac{\partial \alpha(\theta)}{\partial \theta} g(\theta) + \frac{\partial f}{\partial g(\theta)} \frac{\partial g(\theta)}{\partial \theta} \alpha^+(\theta)$
Le premier terme s'annule car $\frac{\partial f}{\partial \alpha(\theta)} = 0$. Ainsi la fonction de coût considère l'amplitude optimisée, et pour une position et un écart-type donnée, on considère l'amplitude comme constante, et α n'a plus à être optimisée à chaque itération pendant le fit des gaussiennes.

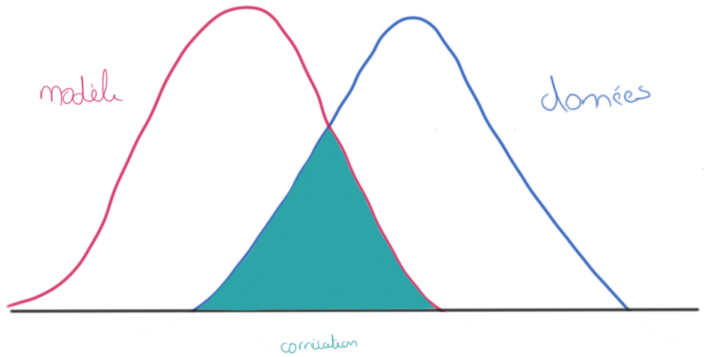

FIGURE 1 – Illustration de la corrélation entre la gaussienne des données et du modèle.

Cas avec plusieurs gaussiennes à fitter :
Il faut prendre en compte la corrélation du modèle d'un spot avec lui même et la corrélation des modèles des spots entre eux, c'est-à-dire de considérer l'ambiguïté due à la proximité des spots.
$f = \|G\alpha - d\|^2_W$ avec :

- α un vecteur
- G une matrice de dimension 3 x k
- d un vecteur

Fonction de coût : $f = [G\alpha - d]^T W [G\alpha - d] = \alpha^T G^T W G \alpha - \alpha^T G^T W d - d^T W G \alpha + d^T W d$
Minimisation : $\frac{\partial f}{\partial \theta} = 2G^T W G \alpha - G^T W d - G^T W d = 0$
Amplitude optimisée : $\alpha^+ = [G^T W G]^- 1 G W d$

Finalement pour avoir une auto-corrélation des gaussiennes avec les données, on inverse la matrice $[G^T W G]^- 1 = A$.

Cette méthode permet de diminuer le nombre d'itérations dans la boucle de fit des gaussiennes.

3 Modèle simulé

Packages utilisés :
TwoDimensional, Zygote, StatsBase, Plots, OptimPackNextGen, FITSIO

Fonctions et structures utilisées :

- Likelihood : renvoie $\sum$(données - modèle)^2
- Gradient : Dérivée de likelihood, soit la dérivée de l'écart entre données et modèle.
  ∇cost = 0 au centre
  ∇cost > 0 pour fwhm + 2
- BoundingBox : délimite un cadre autour de la microlentille.

3.1 Simulation pour une microlentille

Méthode de simulation :
Comparaison d'une image générée avec les paramètres vrais (xtrue, ytrue,atrue, fwhmtrue) et une image finale data de paramètres optimises (xopt, yopt, aopt, fwhmopt). Si les images sont identiques, alors la simulation est correcte.

Bounding Box utilisée : bbox = BoundingBox(xmin=1, ymin=1, xmax=15, ymax=50).

	Jeu de données
Simulation	a0 = 1. .- 0.2 .* randn(Float64,length(laser)); fwhm0= 5. .- 2. .* rand(Float64,length(laser)); C0[1 :2,1 :2] = [[6.2 10]; [25 80]]
Initialisation	ainit = a0 .+ (rand(Float64,laser.n) .- 0.5); fwhminit = fwhm0 .+ (rand(Float64,laser.n) .- 0.5); cinit[1 :2,1 :2] = [[6.2 0]; [25 60]]; xinit = vcat([ainit[:],fwhminit[:],cinit[:]]...)
Optimisation	aopt = xopt[1 :(laser.n)]; fwhmopt = xopt[(laser.n+1) :(2*laser.n)]; copt = reshape(xopt[(2*laser.n+1) :(4*laser.n)]; xopt = vmlmb(likelihood, xinit; verb=50, ftol=(0.0,0),gtol = (0.0,0))

TABLE 2 – Valeurs des coefficients en input

La variation de la taille des gaussiennes est limitée a +/- 2 pixels ici, donc on n'observe pas de changements majeurs pour ce paramètre.

Résultats : On compare les trois paramètres initiaux a0, fwhm0 et Cinit aux paramètres optimisés aopt, fwhmopt. On remarque qu'après optimisation, on retrouve des paramètres proches de ceux du modèle initial, donc l'algorithme fonctionne correctement.
On cherche a définir les limites de convergence du modèle en ajoutant plus ou moins de bruit.

	a	fwhm	C
Simulation	a0 = 1.5748436239455692, 1.1028489598746334, 0.9971761539965861	fwhm0 = 4.158250103938821, 4.986886656324037, 4.041399489366643	C0 = 6.2 10.0 0.0; 25.0 80.0 0.0
Initialisation	ainit = 1.60502548954363, 0.893006548858263, 0.5680885279296609	fwhminit = 2.183734571219645, 3.5775420888950498, 2.487535287595205	cinit = 6.2 0.0 0.0; 27.0 60.0 0.0
Optimisation	aopt = 1.559135041006382, 1.0865996443408623, 1.0055210604597469	fwhmopt = 4.208823580900754, 5.030848856537249, 4.094025131484664	copt = 6.164321794306867 9.906787793407572 3.1203846103944337; 25.05497140495707 80.08155464550333 -2.494673184039535

TABLE 3 – Comparaison des paramètres de la simulation aux paramètres raffinées.

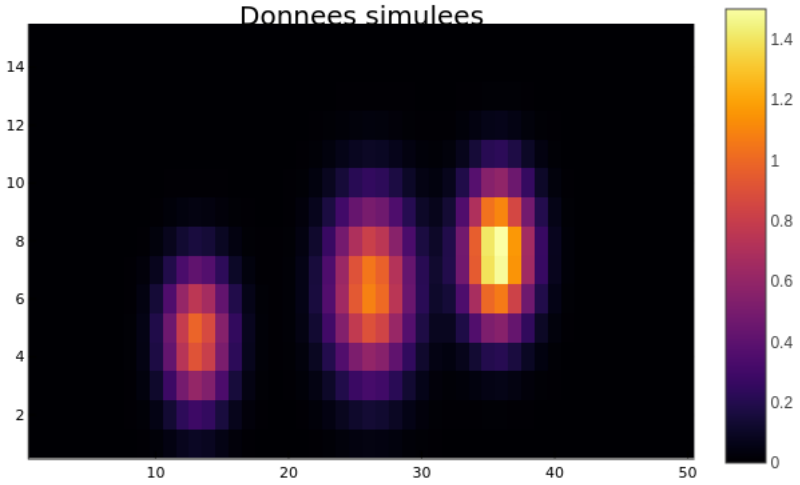

FIGURE 2 – Dans cette simulation d'une microlentille dans le mode YJ, on cherche a ajuster la loi de dispersion au trois spots gaussiens.

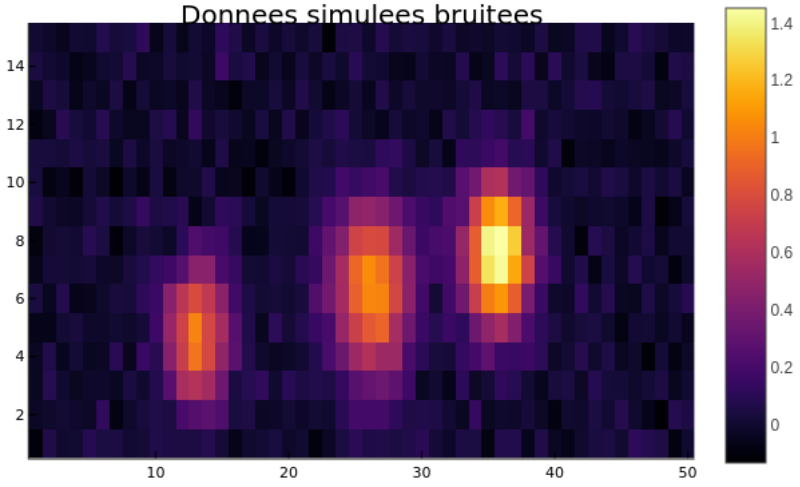

FIGURE 3 – Simulation de la microlentille avec ajout de bruit. On ajoute du bruit a la simulation pour chercher a retrouver les coefficients initiaux après optimisation.

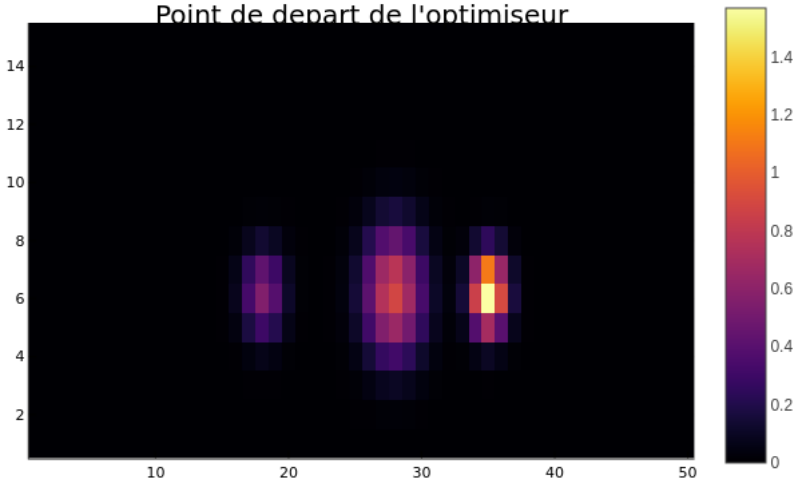

FIGURE 4 – Initialisation de l'optimisation.

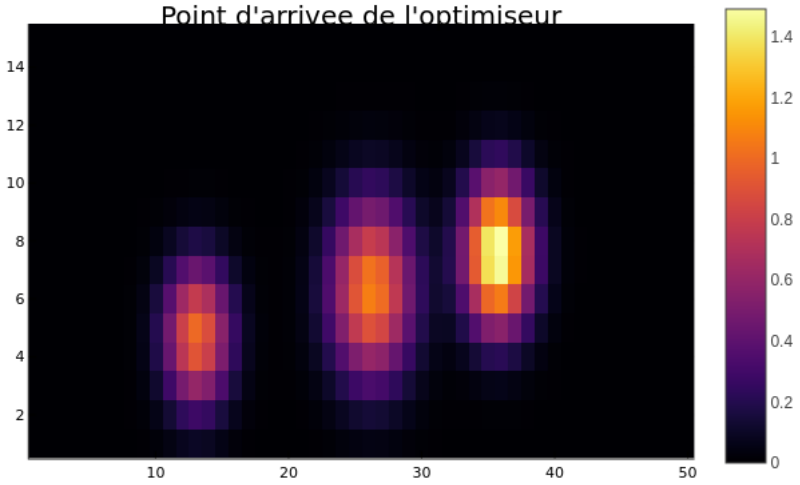

FIGURE 5 – Après optimisation des paramètres. La taille, la position et l'amplitude des trois gaussiennes correspond bien au modèle simulé de départ, donc l'algorithme a bien convergé.

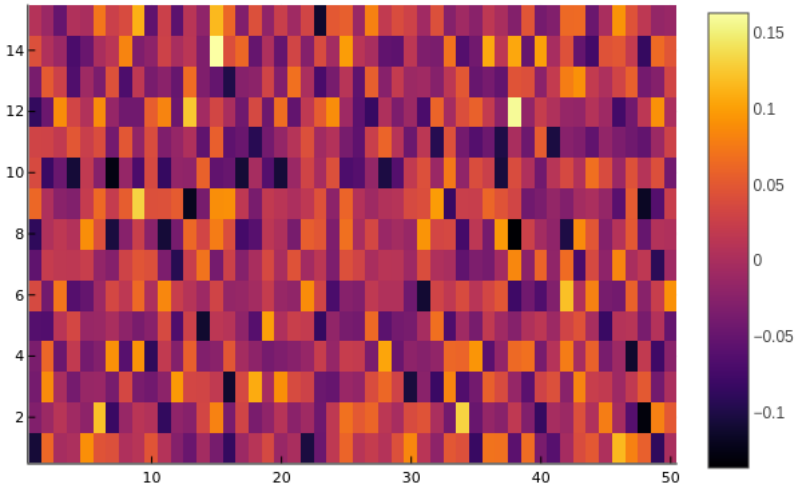

FIGURE 6 – Carte des résidus.

4 Modèle a partir de donnees réelles pour une microlentille

4.1 Méthode

A partir des données réelles, on définit une bounding box et des paramètres le plus proche de la réalité possible. Les paramètres ajustes "a l'oeil" sont attribues aux paramètres d'initialisation de l'optimiseur ainit, fwhminit et cinit..

4.2 Paramètres

Bounding Box utilisée associe au fichiers fits (IFS_ calib_ wave_corrected.fits) :
bbox = BoundingBox(xmin=1, ymin=1, xmax=7, ymax=45).

	a	fwhm	C
Initialisation	ainit = [300.0, 400.0, 500.0]	fwhminit = [2.0, 2.0, 2.0]	cinit = [4.0 0.0 0.0; 22.0 -100.0 0.0]
Optimisation	aopt = [965.046137521718, 583.3811831426702, 424.4586979198295]	fwhmopt = [2.3304777053184464, 2.3977989106373707, 2.826759346117514]	copt = [3.7518467577672254 0.5513220963378493 -1.0473553244754272; 23.52781199044731 -102.72177816785967 -69.26642744587515]

TABLE 4 – Résultats de la simulation

4.3 Extraction d'une microlentille des fichiers de calibration HR4796-HD95086

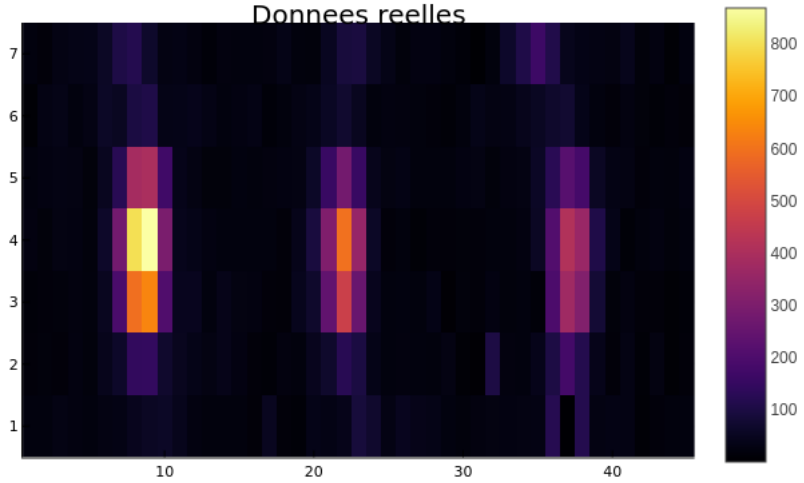

FIGURE 7 – On peut voir en plus des trois gaussiennes d'intérêt au centre de l'image, les gaussiennes des microlentilles voisines en haut et en bas, ainsi qu'un pixel défectueux a cote de la troisième gaussienne.

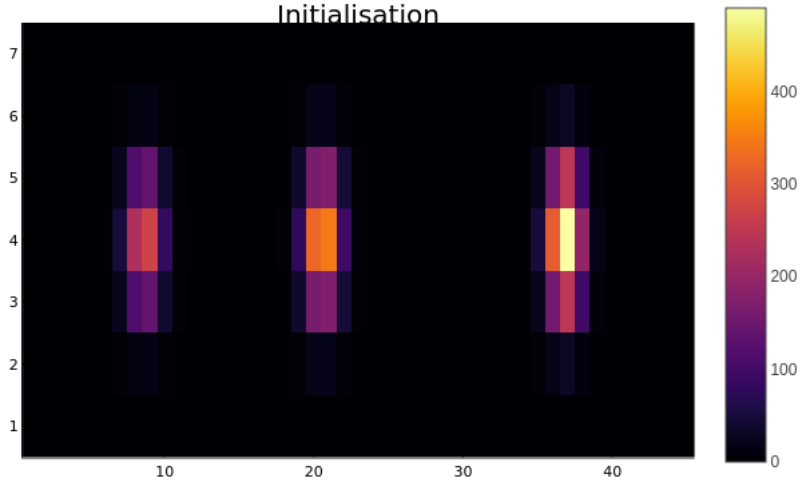

FIGURE 8 – On ajuste les paramètres pour démarrer l'optimisation au plus proche du résultat a obtenir : ainit, fwhminit et cinit. On remarque par ailleurs que la première et la dernière gaussienne sont inverses, ce qui est du a la normalisation du polynôme par λ

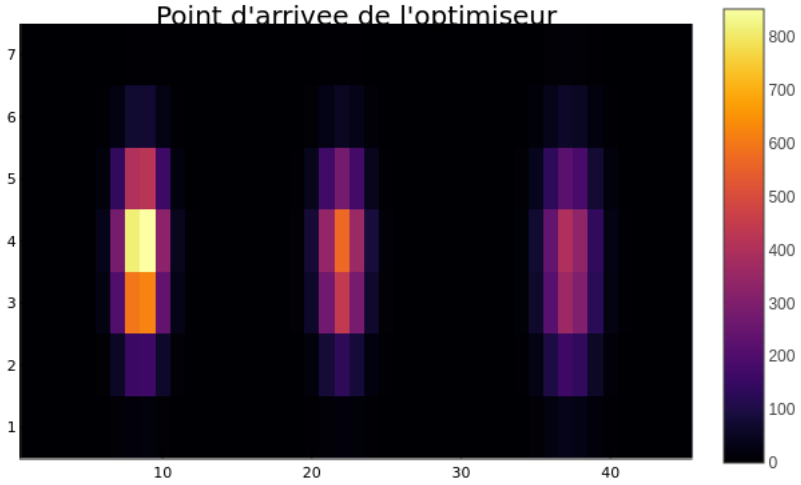

FIGURE 9 – Le modèle a correctement convergé.

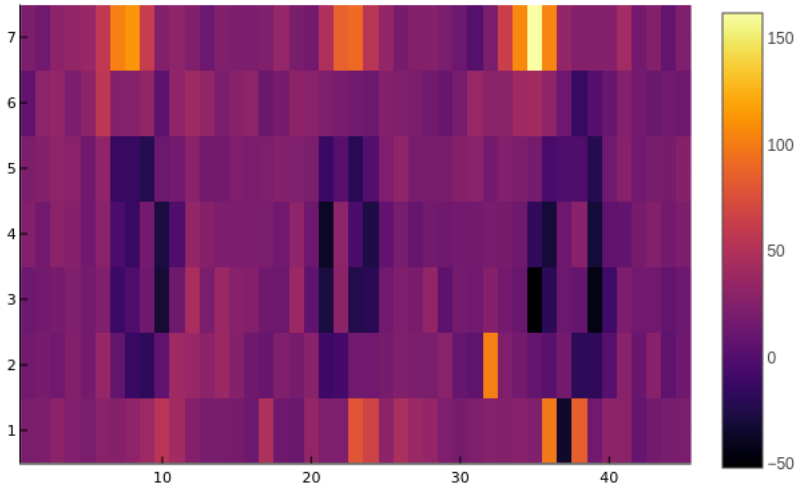

FIGURE 10 – Les gaussiennes voisines n'ont pas été fittées, donc apparaissent sur la carte des résidus.

4.4 Limites de convergence du modèle :

En changeant l'écartement en x de la loi dans cinit, on s'appercoit que le modele ne converge pas correctement.

	a	fwhm	C
Initialisation	ainit = [300.0, 400.0, 500.0]	fwhminit = [2.0, 2.0, 2.0]	cinit = [[4.0 0 0] ; [21 -80 0]]
Optimisation	aopt = [957.6297839687048, 400.62381558386863, 64.22784451117751]	fwhmopt = [2.1836557307254205, 0.0031358875260142144, 39.11348073906345]	copt = [2.8101047188847366 6.744296319149254 5.123873994749534 ; 17.878876830708894 -69.65150227202648 -1.183979082406328]

TABLE 5 – Comparaison des parametres mis en input (Initialisation) et obtenus apres raffinement (Optimisation)

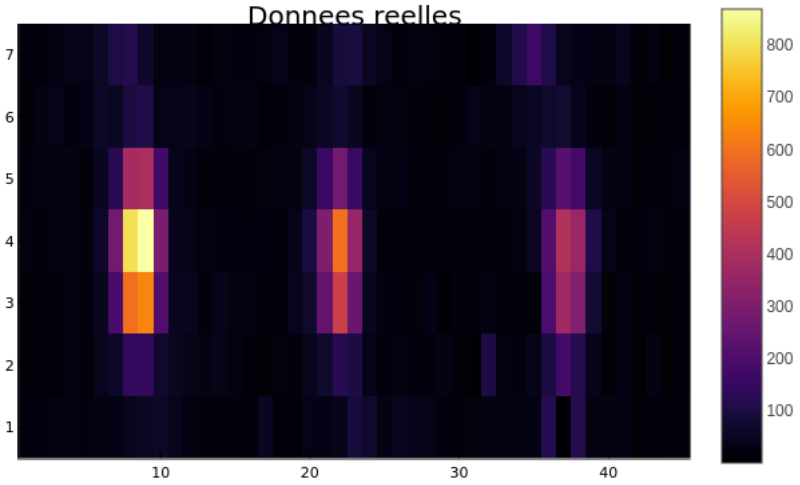

FIGURE 11 – Données réelles.

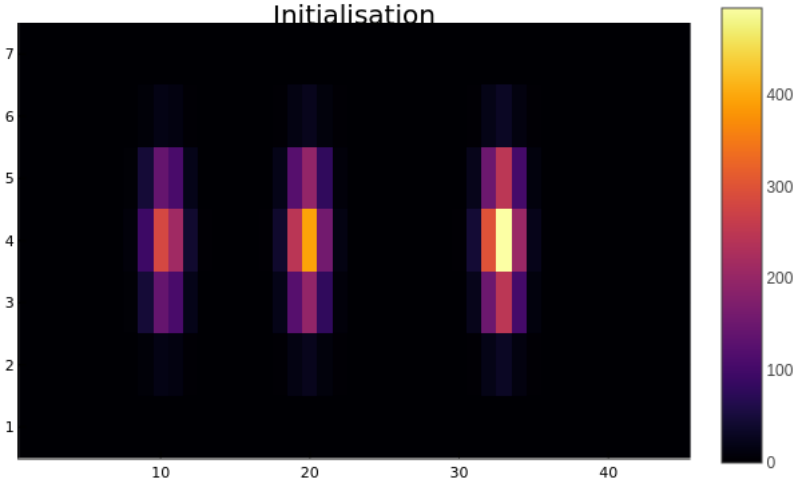

FIGURE 12 – On remarque le décalage de la position des gaussiennes en x du a la modification de cinit.

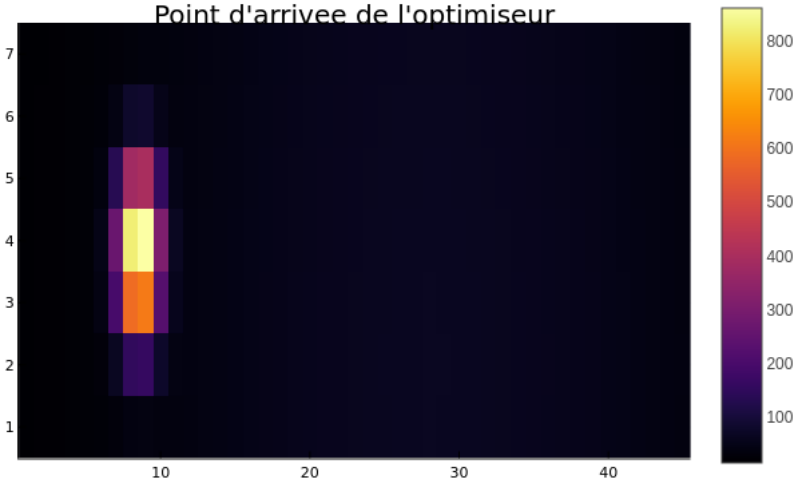

FIGURE 13 – Les gaussiennes n'ont pas été correctement fittées.

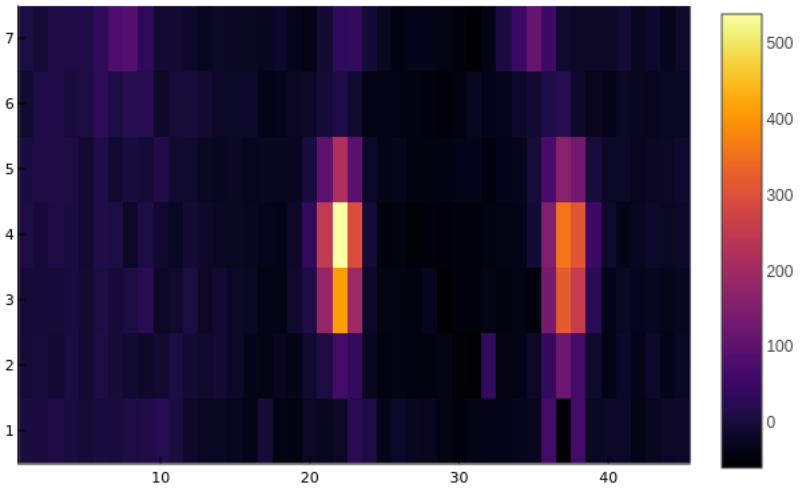

FIGURE 14 – Résidus. On retrouve les deux gaussiennes.

4.5 Fit sur une microlentille :

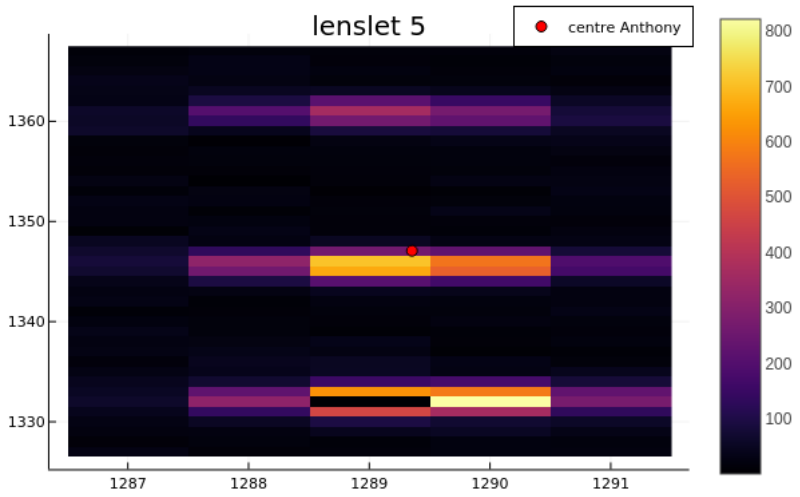

FIGURE 15 – Exemple de la lenslet n°5.

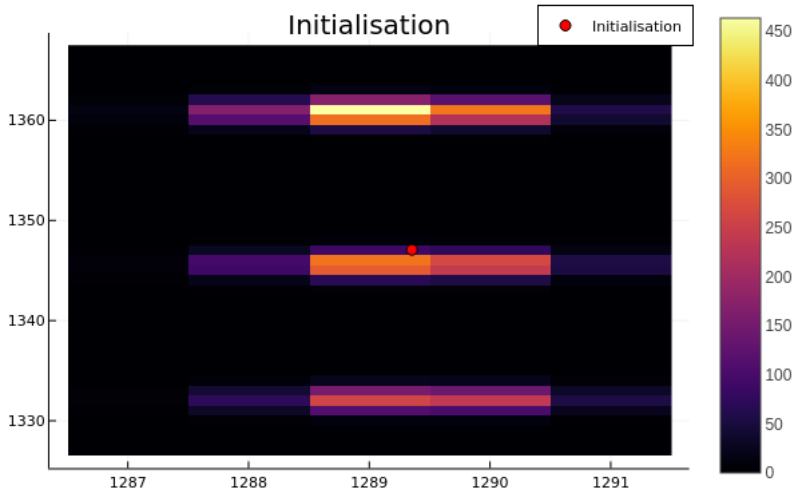

FIGURE 16 – Initialisation du fit.

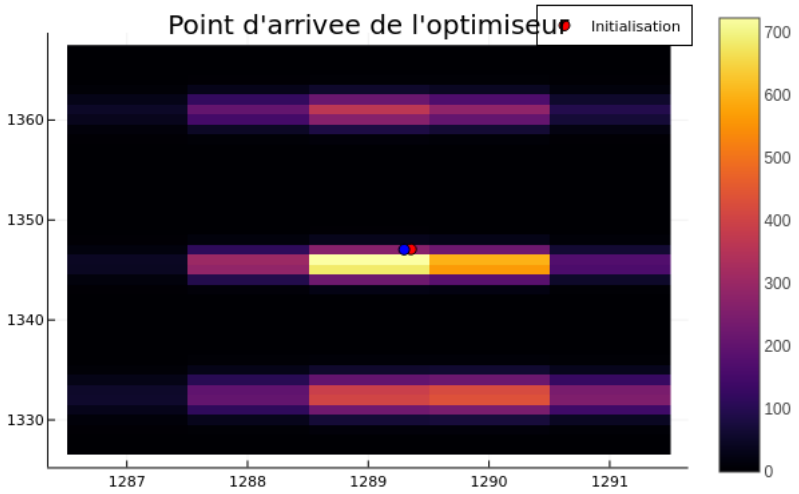

FIGURE 17 – Après optimisation.

	a	fwhm	C
Initialisation	ainit = [300.0, 400.0, 500.0]	fwhminit = [2.0, 2.0, 2.0]	Cinit = [999.5222834913245 -0.6843665411452149 -0.41446582217274436; 1073.4407472368255 102.60070991107642 -68.43665668411037]
Optimisation	aopt = [978.4291799788813, 677.3126304628319, 619.4889053279672]	fwhmopt = [2.254523487266258, 2.3741053651236075, 2.4887114132663366]	copt = [999.517706523665 -0.639341072434858 -1.803139952491615; 1073.4624158780084 102.87429999286442 -68.06074633439525]

TABLE 6 – Résultats de la simulation

On remarque aue les amplitudes ne sont pas

4.6 Fit sur l'ensemble des microlentilles :