Réduction de données - Néo-Narval

Génération de spectre - *generate_lane_spectrum.py*

Sylvain LUCAS

MàJ du 17 août 2017

1 Introduction

L'objectif de ce programme est de générer un spectre à partir d'une image d'une voie. C'est un outil générique capable de traiter des données issues à la fois de l'étoile, du Fabry-Perot, du Flat ou même du Thorium Argon. Ceci est possible en utilisant la matrice A précédemment créée. Ce rapport a ainsi pour vocation d'expliquer la méthode mise en place en détaillant la structure et la philosophie adoptées pour ce module. Notons que ce document ne se substitue en rien à la documentation interne au code, mais permet d'adopter un point de vue plus global vis-à-vis de ce dernier, en explicitant les choix effectués.

2 Idée générale

Comme cela a déjà été expliqué dans la documentation relative à la matrice A, on se place dans la situation suivante : On considère que l'on a l'image issue de la CCD d'une voie d'un ordre, que l'on met sous la forme d'un vecteur colonne Y. On suppose que l'on a également construit la matrice A pour cette voie. on cherche donc à obtenir le spectre X, sous la forme d'un vecteur colonne tel que :

$$AX = Y$$

En effet, la matrice A est construite comme le montre la figure suivante. Chaque ligne donne donc les différentes participation d'un pixel aux longueurs d'onde et on peut lire sur une colonne la participation des pixels à une longueur d'onde.

$$A = \begin{array}{c} \begin{array}{cccccc} \lambda_1 & \lambda_2 & \dots & \dots & \lambda_{n_{art}} \end{array} \\ \begin{pmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \end{pmatrix} \end{array} \begin{array}{c} pix_1 \\ pix_2 \\ \vdots \\ \vdots \\ pix_{n_{pix}} \end{array}$$

Avant de continuer davantage les explications concernant la matrice A, il est tout d'abord nécessaire de clarifier la construction du vecteur image Y. Le programme prend en effet en entrée une image complète (2D) issue de la CCD. Grâce à l'enveloppe de la voie fournie (voir *Données en entrée*), on peut parcourir l'image et copier les données dans un tableau 1D. Avec les conventions d'orientation décrite plus loin, le parcours se fait avec les y croissants puis les x croissants. On a donc le sens de parcours sur la voie donné par la figure 1.

On souhaite donc obtenir le spectre X. Il suffirait simplement d'inverser le système de manière à obtenir $X = A^{-1}Y$. Cependant, deux obstacles s'opposent à cela. Le premier est que la matrice A n'est pas carrée mais pas rectangulaire (par exemple pour Narval elle sera environ de dimension $(7000; 118000)$ avec deux voies par ordre et à $1.5 art/pix$). Cependant on peut quand même déterminer une pseudo-inverse en cherchant notamment les valeurs singulières de la matrice. Vient alors le second obstacle : la complexité du calcul. En effet, au vu des dimensions et du nombre de valeurs singulières à chercher (près de 7000 dans le cas précédent), le calcul pour une unique voie (sachant que l'image en compte 80) sera extrêmement long. On choisit donc de prendre un chemin détourné qui nous permet d'obtenir X sans pour autant calculer explicitement A^{-1} en implémentant la méthode des équations normales tel que décrite dans [1].

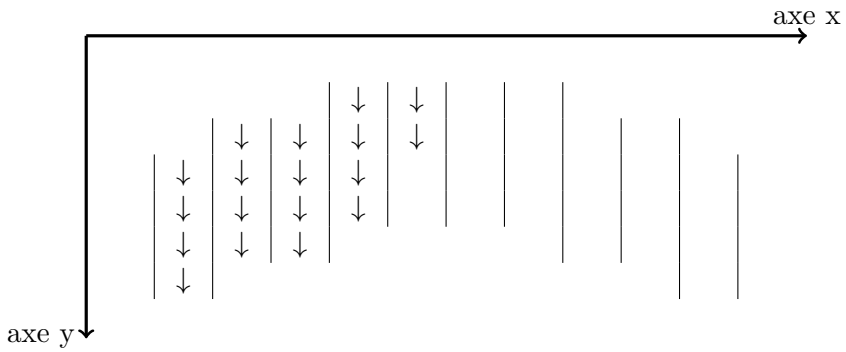

FIGURE 1 – Sens de parcours de la voie (construction du vecteur Y)

3 L'algorithme en détail

3.1 Initialisation (`create_spectrum`)

Le programme charge tout d'abord par le biais de cette fonction l'ensemble des fichiers et informations nécessaires à la réalisation du spectre. C'est également cette fonction qui s'occupe d'enregistrer au bon format le spectre une fois qu'il est calculé.

3.2 Génération du spectre (`generate_spectrum`)

Après avoir chargé la matrice A en mémoire, cette fonction s'occupe tout d'abord de la couper au niveau des limites de l'interpolation (nécessaire pour la factorisation de Cholesky). L'image de la voie est ensuite créée sous forme du vecteur Y.

On utilise ensuite la méthode des équations normales, c'est-à-dire :

— On calcule $C = A^tA$, matrice symétrique définie positive dont seule les valeurs diagonales sont non nulles.

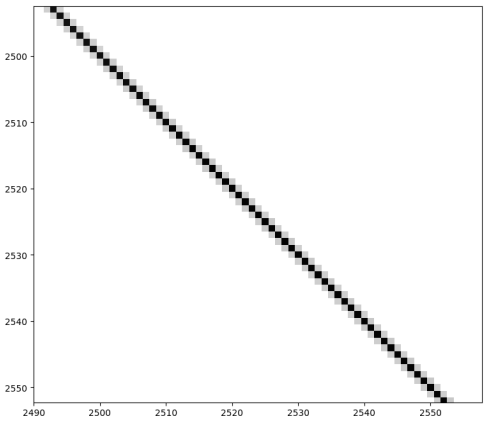

FIGURE 2 – Matrice C (partielle)

— Factorisation de Cholesky de C telle que $C = GG^t$

— Création du vecteur d : $d = A^tY$.

— On résout $Gy = d$ puis $G^tx = y$. On trouve alors x le spectre recherché.

Il nous suffit ensuite de compléter le vecteur créé pour lui donner sa longueur normale (longueur de la matrice A si elle n'avait pas été coupée aux limites de l'interpolation). Le spectre est ainsi complété à la valeur -1 sur ces portions, comme le montre la figure 4.

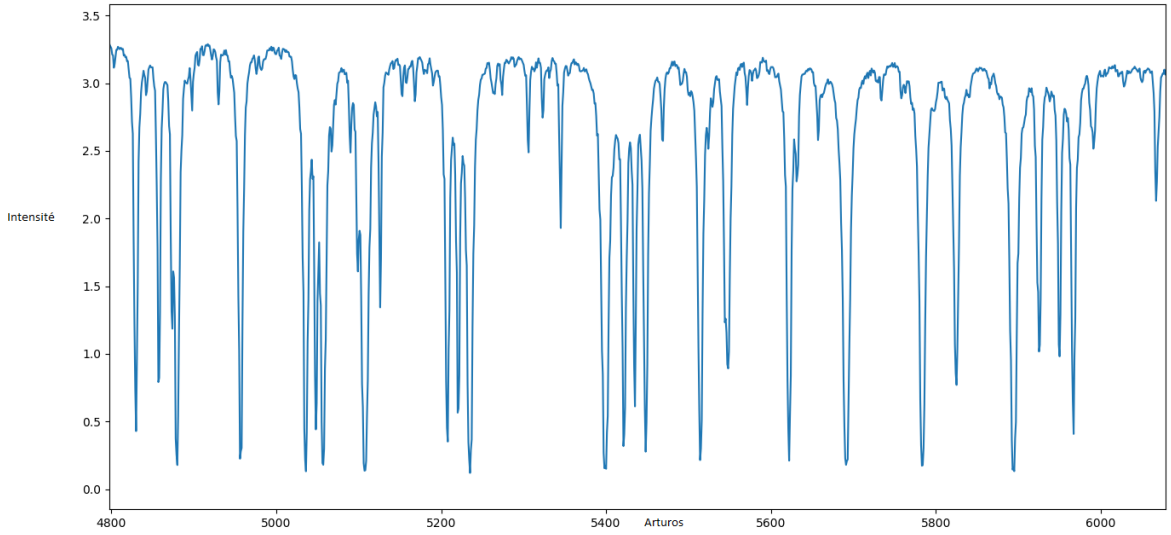

FIGURE 3 – Spectre partiel d'une étoile (1.5 art/pix)

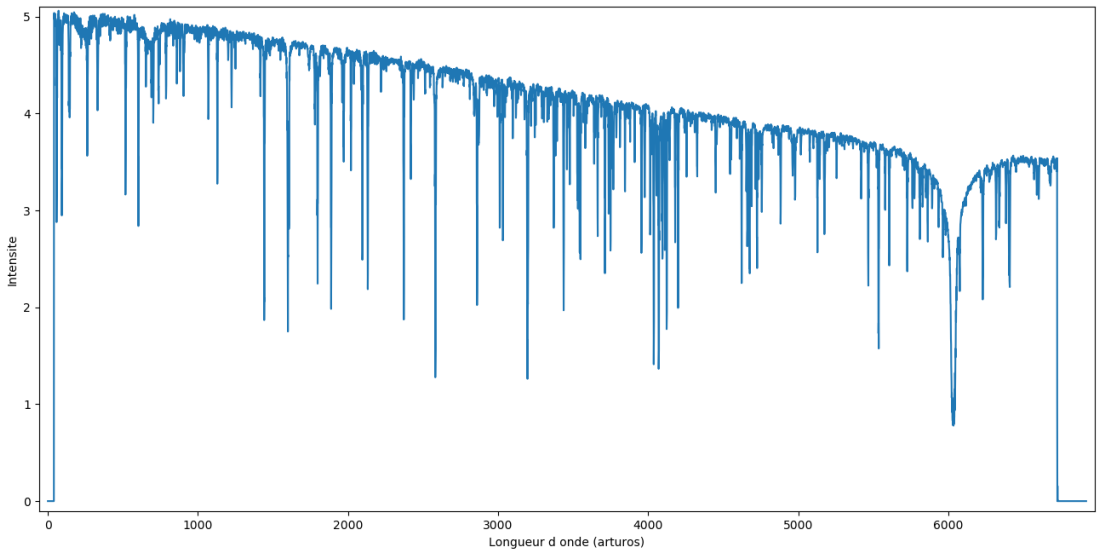

FIGURE 4 – Spectre complet (1 voie, 1.5 art/pix)

4 Spectres

4.1 Normalisation

Attention : L'ensemble des spectres présentés ici sont issus d'une normalisation à l'aide d'un spectre de flat. En effet, lors de la génération d'un spectre seul, la construction de la matrice A introduit des "erreurs" sur le spectre, qui sont reproduites sur le spectre du flat. La normalisation permet donc de s'en débarrasser.

4.2 Résultats

Lorsqu'on augmente le nombre d'arturos par pixel ($> 1 art/pix$), du "bruit" apparaît. On peut s'en débarrasser à l'aide d'un filtre passe-bas. Il est cependant nécessaire d'être prudent lors de l'utilisation d'un tel filtre afin de ne pas perdre d'information. La figure 5 présente un exemple de ce filtrage. Cette fonctionnalité n'est cependant pas directement implémentée dans `generate_lane_spectrum.py`, mais est présente dans le traitement `concatenate_toFITS.py`.

Références

[1] Gene H Golub and Charles F Van Loan. *Matrix computations*, chapter 5.3.2, page 238. JHU Press, third edition, 1996.

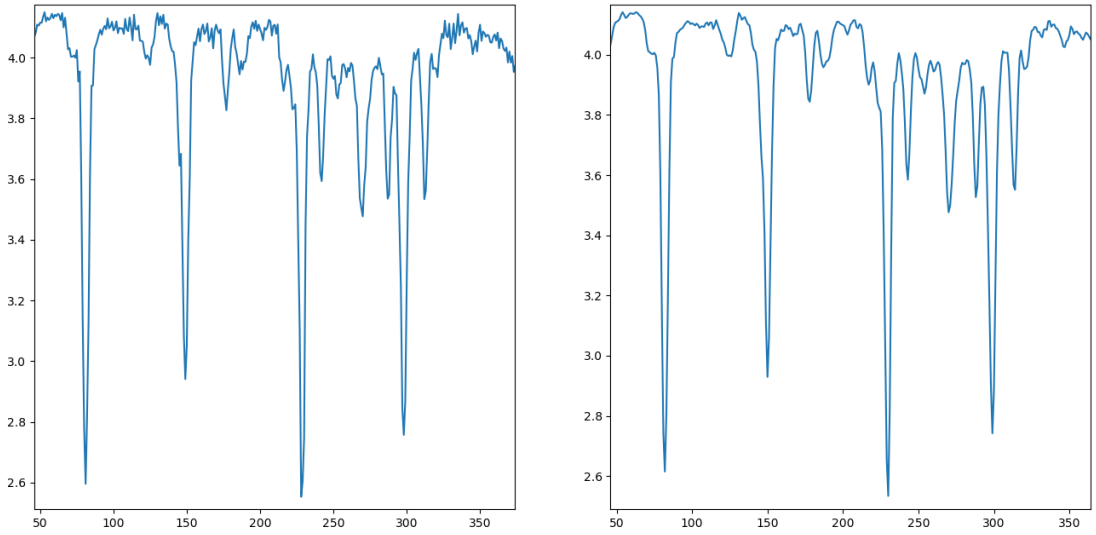

FIGURE 5 – Comparaison avant/après application d'un filtre passe-bas (1.5 art/pix)