# Traitement d'images hyperspectrales

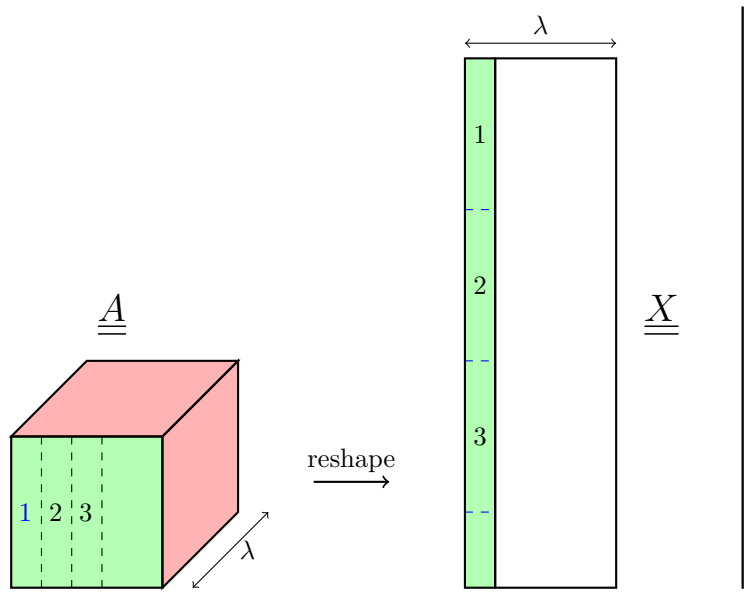
Yang Yi, XU Kaiyuan

17 Janvier 2025

## Table des matières

# 0 Introduction & Visualusation des données

Dans ce TP, on s'intéresse à l'étude des images hyperspectrales, qui sont des images $2D$ associées à une séquence de longueurs d'onde $\lambda$. Plus simplement, pour une image RGB, on peut considérer que l'on a pris trois images à $\lambda_{blue} = 450\ nm$, $\lambda_{green} = 550\ nm$, $\lambda_{red} = 700\ nm$. Mais pour les images hyperspectrales, $\lambda$ peut couvrir une large gamme de valeurs.




**Description du schéma :**

— $A$ est une image hyperspectrale. Pendant ce TP, on a choisi `Reslice_Stack_NAU2-60X-1.tif` comme $A$ ; `size(A)` donne les dimensions de A, qui sont $249 * 696 * 457$.

— $X$ est un nuage de `n_pix` est le nombre de pixels et $n_\lambda$ le nombre de longeurs d'ondes ; `size(X)` donne les dimensions de X, qui sont $173304 * 457$.

La commande permettant de transformer une matrice 3D $A$ en une matrice 2D $X$ « dépilée » est :

```
X = reshape(A,[size(A,1)*size(A,2), size(A,3)]);
```

où chaque pixel de l'image correspond à une ligne de $X$.

⇝ Images obtenues après le découpage de l'image 3D $A$ aux positions $n = 1, 299, 475$ avec la longueur d'onde associée $\lambda = 399\ nm, 684\ nm, 1000\ nm$

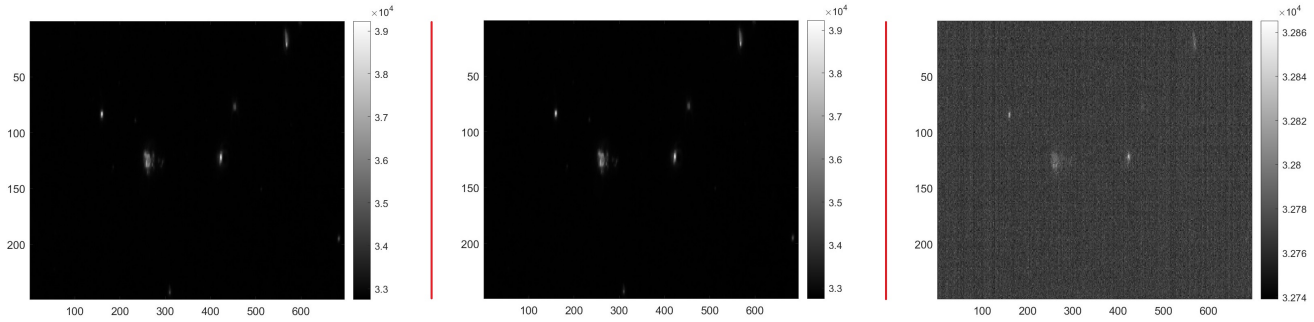


FIGURE 0.0.1 – Pile d'images d'images $n_\lambda = 1|299|475$ avec la longueur d'onde associée $\lambda = 399\ nm \mid 684\ nm \mid 1000\ nm$

Le noir en arrière-plan ne signifie pas l'absence de données, mais simplement une intensité beaucoup plus faible que l'intensité maximale.

⇝ On peut aussi effectuer une sommation de toutes les piles d'images avec la commande `sum(A,3)` :

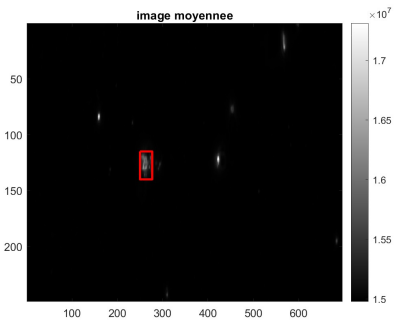



FIGURE 0.0.2 – Image moyenne $S(x, y)$.

soit

$$\boxed{S(x,y) = \sum_{\lambda=1}^{n_\lambda} f(x,y,\lambda)}$$

$\rightsquigarrow$ Concernant la 2D matrice $X$, on peut la visualiser en ajoutant un seuil - soit pour les valeurs des pixels supérieures à ce seuil, on les remplace par cette valeur limite :

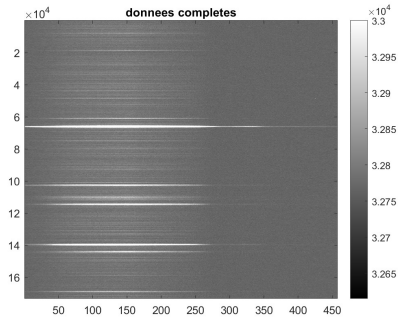



FIGURE 0.0.3 – Données completes

$\rightsquigarrow$ On peut également tracer au hasard 3 exemples de signaux à partir d'une ligne de la matrice 2D $X$, chaque figure représentant l'évolution de l'intensité du signal en fonction de la longueur d'onde :

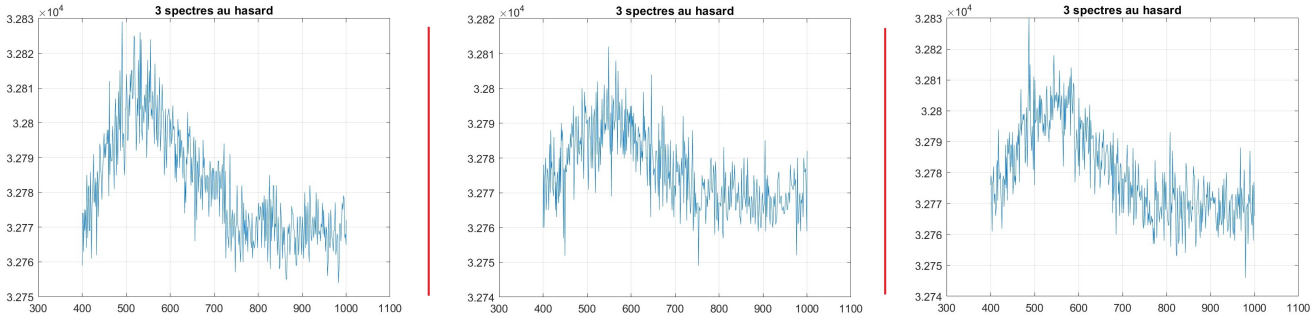



FIGURE 0.0.4 – L'intensité du signal en fonction de la longueur d'onde à partir d'une ligne choisie au hasard de $X$

Chaque ligne de la matrice 2D $X$ represnete un pixel dans la matrice 3D $A$ sur l'ensemble des longueurs d'onde.

# 1 Pretraitement : retrait d'une ligne de base

## 1.1 Noise

Pour calculer `X_bruit_zoom`, on sélectionne dans $A$ les colonnes $1:50$ et les lignes $1:50$ avec toutes les valeurs de $\lambda$ , puis on utilise la fonction `reshape` pour obtenir le résultat. Chaque colonne de la matrice `X_bruit_zoom` représente une image $2D$ de la zone sélectionnée précédemment pour une certaine valeur de $\lambda$. Nous utilisons simplement la commande `baseline = mean(X_bruit_zoom)`, qui calcule la moyenne de chaque colonne et la stocke dans la variable `baseline`, soit chaque élément de `baseline` est la moyenne des valeurs de la zone sélectionnée précédemment dans $A$ pour une certaine valeur de $\lambda$ :

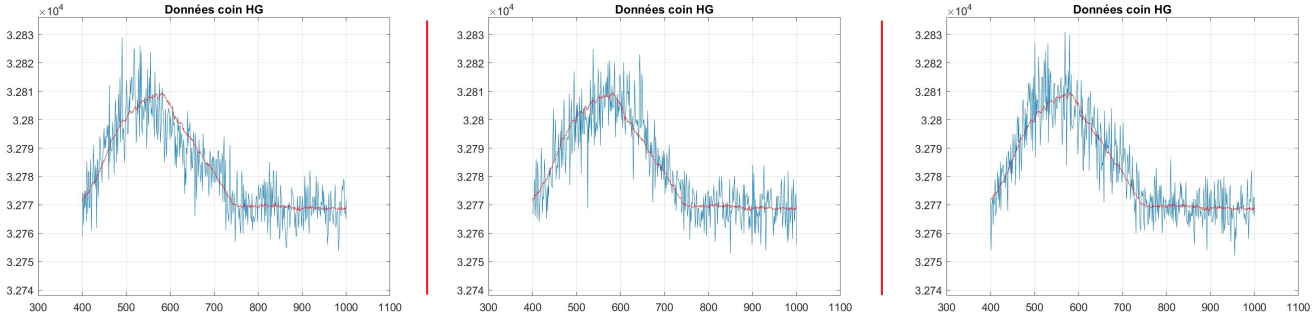
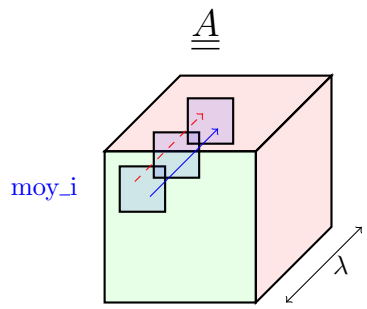



FIGURE 1.1.1 – L'intensité du signal zoomé en fonction de la longueur d'onde à partir d'une ligne (`nb_pix` $= 1|626|1251$) choisie de `X_bruit_zoom`, avec la ligne rouge représentante l'intensité moyenne de chaque zoom choisi par $(x[1:50], y[1:50])$ dans $A(x,y,\lambda)$.




**Description du schéma :**

- Pour chaque pixcel dans le carré en bleu (sélectionné par $(x[1:50], y[1:50])$ dans $A(x, y, \lambda)$), on peut tracer son intensité en fonction de $\lambda$
- Pour chaque carré bleu, on calcule sa moyenne ;
- On trace le signal zoomé et la `baseline` dans FIGURE 1.1.1 .

`baseline = mean(X_bruit_zoom)` : « Noise »

## 1.2 Inforamtion utile

Même processus, cette fois on choisit le zoom ROI (Region of Interest cf. FIGURE 0.0.2 rectengle en rouge : sélectionné par $(x[115:140], y[250:277])$ dans $A(x, y, \lambda)$) nommé `A_zoom`.

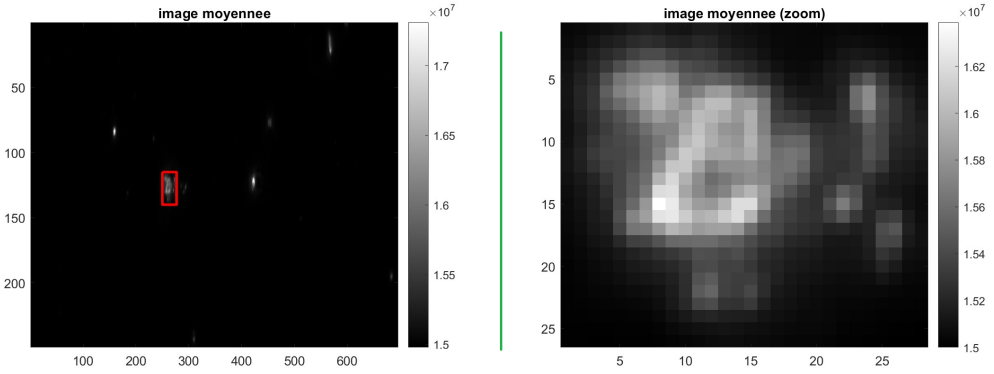



FIGURE 1.2.1 – Image moyenne $S(x, y)$ et `A_zoom` | Image moyenne de `A_zoom` : `sum(A_zoom,3)`

Pour passer de `A_zoom` à `X_zoom`, on utilise toujours la fonction `reshape` :

```
X_zoom = reshape(A_zoom,[size(A_zoom,1)*size(A_zoom,2), size(A_zoom,3)]);
```

Et l'information utile soit `X_zoom_preprocess` est `X_zoom` moins le « noise » `baseline` :

`X_zoom_preprocess` = `X_zoom` - `baseline` : « Inforamtion utile »

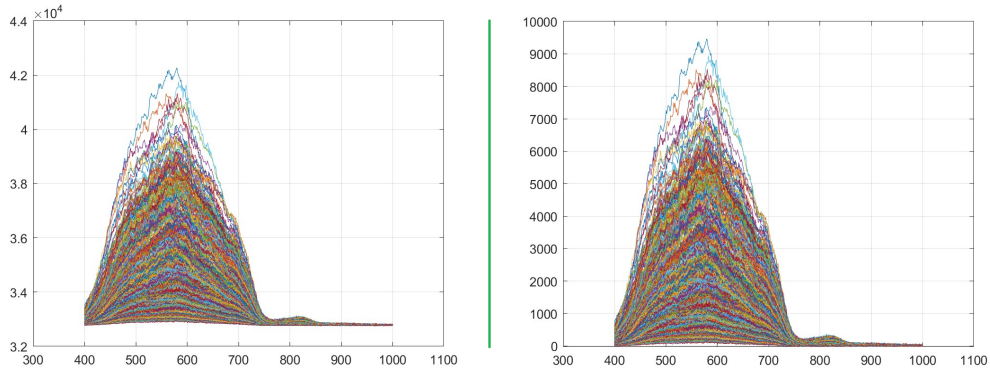


FIGURE 1.2.2 – L'intensité du signal en fonction de la longueur d'onde à partir de chaque ligne de `X_zoom` | | L'intensité du signal en fonction de la longueur d'onde à partir de chaque ligne de `X_zoom_preprocess`

## 2 Débruitage par Analyse Par Composantes Principales

Remarque que la dimension de `X_zoom_preprocess` est $728(= n_x * n_y = 26 * 28) \times 457(= n_z = n_\lambda)$, on voudrait utiliser la technique de l'ACP pour réduire la dimension jusqu'à $p \ll n_\lambda$ soit on va choisir les information le plus importante les informations les plus importantes, qui sont une **combinaison linéaire** des vecteurs propres de la matrice de haute dimension `X_zoom_preprocess`. Voici le processus :

⇝ On nomme `Z` la version centrée de `X_zoom_preprocess`.

```
Z = X_zoom_preprocess-ones(n,1)*mean(X_zoom_preprocess);
```

⇝ On applique SVD (décomposition en valeurs singulières) :

$$[U, S, V] = svd(Z, 0);$$

⇝ On représente $Z$ dans les nouvelles coordonnees où `C(:,k)` est la projection de `Z` sur la k-ième composante principle :

```
C = Z*V;
```

⇝ On peut obtenir la valeur propre $S$ à partir de la valeur singulière $s$ :

```
s = diag(S); val_propres = s.^ 2;
```

$s = [s_1, s_2, \cdots, s_N]^T$ où $s_1$ est la plus grande valeur singulière. On peut donc calculer la somme cumulée de la valeur prores $S = s^2$ :

$$\text{somme cumulée de la valeur prores } \lambda_i = \frac{\sum_{k=1}^{i}}{\sum_{k=1}^{N}}$$

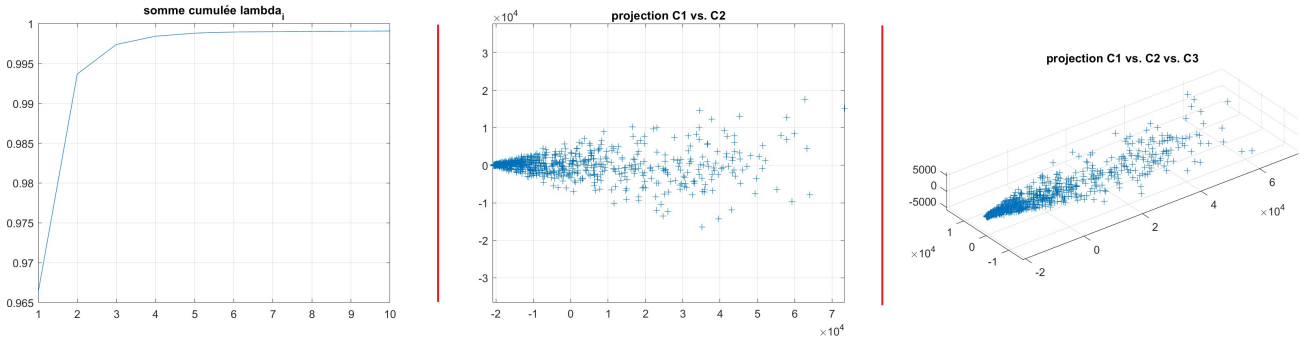



FIGURE 2.0.1 – Somme cumulée de la valeur prores $\lambda_i$ | La projection de `Z` sur les deux premières composantes principales `C(:,1)` et `C(:,2)` | La projection de `Z` sur les trois premières composantes principales `C(:,1)`, `C(:,2)`, `C(:,3)` .

⇝ La matrice des $\cos^2$ peut représenter la distance/le coefficient de projection de chaque point sur le sous-espace de dimension $\dim - 1$. Ce coefficient est de $1 - \cos^2(i, k)$, ce qui est l'information projetée en dehors de la $k$-ième composante principale. Avec $\cos^2(i, k)$ qui représente, pour un point $i$, la part du carré de sa projection sur la k-ième composante principale par rapport à la somme des carrés de ses projections sur toutes les composantes principales :

$$\cos^2(i, k) = \frac{C_{i,k}^2}{\sum_{j=1}^{p} C_{i,j}^2}$$

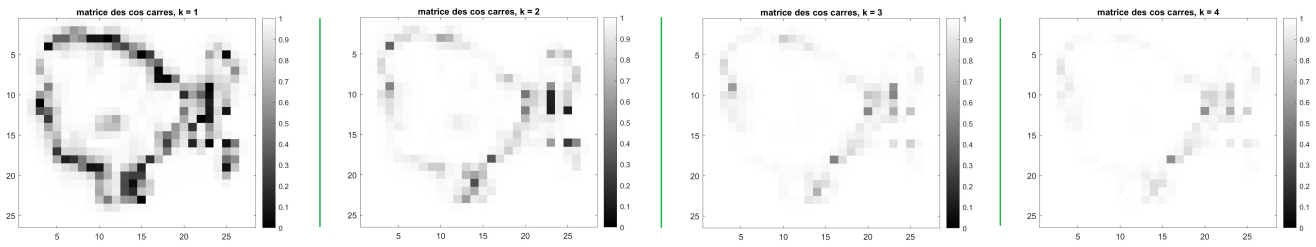



FIGURE 2.0.2 – La matrice de $\cos^2(i, k)$ `mat_coscarre` avec $k = 1, 2, 3, 4$.

## 3 Factorisation en Matrices Non-Négatives

Dans la partie suivante, $X$ correspond à la data `X_zoom_preprocess` !

On cherche à résoudre le problème de séparation de sources non-négatives, c'est-à-dire à décomposer les donnée sous la forme $X = HS$, où $H_{i,j} \geq 0, S_{i,j} \geq 0$ avec $H$ de dim $(n_x * n_y) \times p$, $S$ de dim $4 \times n_\lambda$. Ou l'image hyperspectrale $X$ peut être écrite sous la forme

$$\boxed{X(x, y, \lambda) = \sum_{i=1}^{p} H_i(x, y) S_i(\lambda)}$$

La méthode proposée (ALS pour Alternative Least Squares) consiste à minimiser $||X - HS||_F{}^2$ alternativement par rapport à $H \geq 0$ pour $S$ fixé, et par rappoet à $S$ pour $H$ fixé. [1]

**Algorithm 1:** ALS général

```
/* ALS général */
Input          : X, H, S, K
Output         : H^(K+1), S^(K+1)
Initialisation : H^(0), S^(0), k = 1
do
    minimize_A ||X − H S^(k)||_F^2 ;
    update H^(k+1);
    minimize_S ||X − H^(k+1) S||_F^2 ;
    update S^(k+1)
while k ≤ K and Stopping condition == False;
```

On peut considérer les conditions d'arrêt comme :

1. $f(x_k) - f(x_{k+1}) < \varepsilon_1$
2. $\|x_{k+1} - x_k\|_2 < \varepsilon_2$
3. $\|\nabla f(x_k)\|_2 < \varepsilon_3$
4. $k > K_{\max}$

avec $f(H, S) = \|X - HS\|_F{}^2, \quad x_k = [H^{(k)}, S^{(k)}]^T$

Prof. Charles Soussen a proposé d'utiliser la condition d'arrêt suivante $\frac{f(x_{k-1}) - f(x_k)}{f(x_{k-1})} \leq \epsilon = 0.001$ :

```
(f(h_previous, s_previous) - f(h_present, s_present)) / f(h_previous, s_previous) < 1e-3
```

## 3.1 Méthode 1 pour Updating

⇝ On rappelle la fonction coût à minimiser :

$$\|X - HS\|_F{}^2 = \sum_{i,j} (X_{ij} - (HS)_{ij})^2$$

ou sous la forme matricielle

$$\|X - HS\|_F{}^2 = \mathrm{Tr}((X - HS)(X - HS)^T) = \mathrm{Tr}(XX^T - XS^TH^T - HSX^T + HSS^TH^T)$$

On calcule son gradient :

$$\frac{\partial}{\partial H_{ik}} \|X - HS\|_F{}^2 = -2 \sum_j X_{ij} S_{kj} + 2 \sum_j (HS)_{ij} S_{kj}$$

ou sous la forme matricielle, sachant que $\frac{\partial \mathrm{Tr}(AB)}{\partial A} = B^T$ :

$$\frac{\partial}{\partial H} \|X - HS\|_F{}^2 = (0 - XS^T - XS^T + (HSS^T + HSS^T)) = -2XS^T + 2HSS^T = -2(+XS^T - HSS^T)$$

Ce qui donne la formule de mise à jour de H :

$$H \leftarrow H. * (XS^T)./(HSS^T)$$

plus précisément :

$$H^{(k+1)} = H^{(k)}. * (X^{(k)} S^{(k)T})./(H^{(k)} S^{(k)} S^{(k)T})$$

où $.*$ et $./$ sont les opérations de multiplication et division termes à termes.

$\rightsquigarrow$ En remarquant que $H$ et $S$ jouent des rôles symétriquement dans le problème (on pourra remarquer que la norme de Frobénius d'une matrice est égale à celle de sa transposée)

$$\|X - HS\|_F{}^2 = \|X^T - (HS)^T\|_F{}^2 = \|X^T - S^T H^T\|_F{}^2$$

$$S_{ji} = {S_{ij}}^T \leftarrow \frac{{S_{ij}}^T (X^T H)_{ij}}{(S^T H^T H)_{ij}} = \frac{{S_{ij}}^T (H^T X)_{ji}}{(H^T H S)_{ji}}$$

Ce qui donne la formule de mise à jour de S :

$$S \leftarrow S. * (H^T X)./(H^T H S)$$

plus précisément :

$$S^{(k+1)} = S^{(k)}. * (H^{(k+1)T} X^{(k)})./(H^{(k+1)T} H^{(k+1)} S^{(k)})$$

**Algorithm 2:** ALS 1

```
/* ALS 1 */
```

**Input** : $X, H, S, K$
**Output** : $H^{(K+1)}$, $S^{(K+1)}$
**Initialisation** : $H^{(0)}$, $S^{(0)}$, $k = 1$

$f(H,S) \coloneqq \|X - HS\|_F{}^2$;
**do**
  $H^{(k+1)} = H^{(k)}. * (X * S^{(kT)})./(H^{(k)} * S^{(k)} * S^{(kT)})$ ;
  $S^{(k+1)} = S^{(k)}. * (H^{(k+1)T} * X)./(H^{(k+1)T} * H^{(k+1)} * S^{(k)})$ ;
  $k = k + 1$
**while** $k \leq K$ ***and*** $(f(H^k, S^{(k)}) - f(H^{(k+1)}, S^{(k+1)}))/f(H^{(k)}, S^{(k)}) > 0.001$;

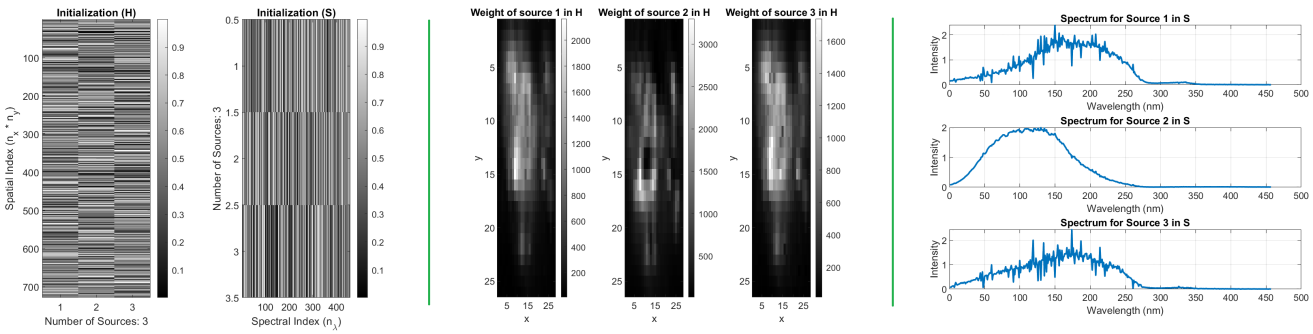



FIGURE 3.1.1 – Initialisation : $H^{(0)}$, $S^0$ | $H^{(K+1)}$ | $S^{(K+1)}$

L'image la plus à droite représente la source obtenue à l'aide de l'algorithme ALS 1 (correspondant à une ligne de $S^{K+1}$) : la première image correspond à la source 1 et la troisième image correspond à la source 3. La première et la troisième image présentent des similitudes, ce qui pourrait être dû à un choix inapproprié des conditions initiales de $H^0$ et $S^0$. En effet, dans cet algorithme, l'influence des conditions initiales n'a pas pu être efficacement éliminée. De plus, la fonction que nous cherchons à minimiser est non convexe. Si les valeurs initiales $H^{(0)}$ et $S^{(0)}$ sont mal choisies, l'algorithme risque de converger vers un optimum local au lieu d'un minimum global.

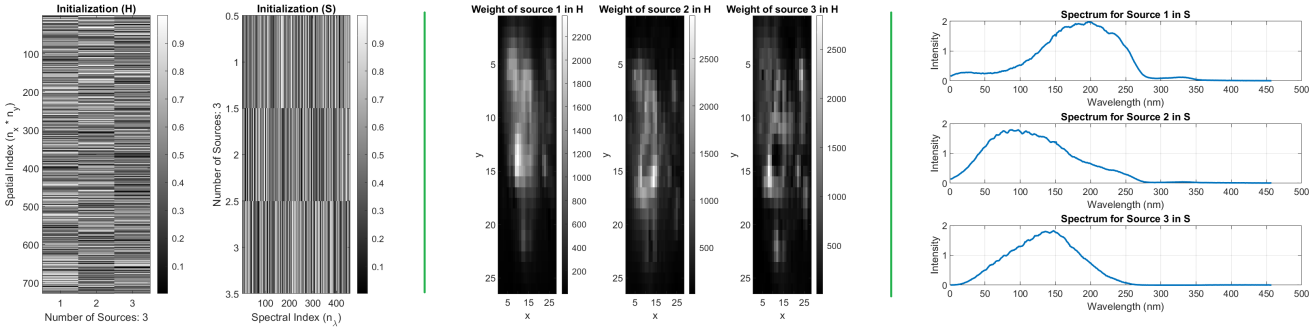



FIGURE 3.1.2 – Initialisation : $H^{(0)}$, $S^0$ | $H^{(K+1)}$ | $S^{(K+1)}$

Maintenant, après avoir modifié les conditions initiales $H^{(0)}$ et $S^{(0)}$, nous pouvons constater que les trois sources ont été efficacement distinguées.

### 3.2 Méthode 2 pour Updating

⇝ On rappelle que

$$\frac{\partial}{\partial H}\|X - HS\|_F^2 = -2XS^T + 2HSS^T = -2(+XS^T - HSS^T)$$

On pose cette équation égale à 0, ce qui donne :

$$XS^T = HSS^T$$

soit

$$H = XS^T(SS^T)^{-1}$$

Comme on est en train de faire la factorisation en matrices non-négatives soit $H \geq 0$, on introduit l'opérateur $[z]_+ = \max(z, 0)$ qui désigne la projection sur la projection sur l'orthant positif pour un saclaire z, et $[Z]_+$ est défini pour une matrice comme la matrice formée de la projection de chacun des éléments de Z sur l'orthant positif. [1]

$$H = [XS^T(SS^T)^{-1}]_+$$

plus précisément :

$$H^{(k+1)} = max(XS^{(k)T}(S^{(k)}S^{(k)T})^{-1}, 0)$$

⇝ De la même manière, la fonction coût à minimiser

$$\|X - HS\|_F{}^2 = \mathrm{Tr}((X - HS)(X - HS)^T) = \mathrm{Tr}(XX^T - XS^TH^T - HSX^T + HSS^TH^T)$$

On calcule son gradient, sachant que $\frac{\partial \mathrm{Tr}(AB)}{\partial A} = B^T$, $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ :

$$\frac{\partial}{\partial S}\|X - HS\|_F{}^2 = (0 - H^TX - H^TX + (H^THS + H^THS)) = -2H^TX + 2H^THS = -2(+H^TX - H^THS)$$

On pose cette équation égale à 0, ce qui donne :

$$H^TX = H^THS$$

soit

$$S = (H^TH)^{-1}H^TX$$

Comme on est en train de faire la factorisation en matrices non-négatives soit $S \geq 0$, on fait la projection de chaque des éléments de S sur l'orthant positif.

$$S = [(H^TH)^{-1}H^TX]_+$$

plus précisément :

$$S^{(k+1)} = max(H^{(k+1)T}(H^{(k+1)})^{-1}H^{(k+1)T}X, 0)$$

**Algorithm 3:** ALS 2

```
/* ALS 2 */
```

**Input** : $X, H, S, K$
**Output** : $H^{(K+1)}$, $S^{(K+1)}$
**Initialisation** : $H^{(0)}$, $S^{(0)}$, $k = 1$

$f(H, S) := \|X - HS\|_F{}^2$;
**do**
  $H^{(k+1)} = \max(XS^{(k)T}(S^{(k)}S^{(k)T})^{-1}, 0)$ ;
  $S^{(k+1)} = \max((H^{(k+1)T}H^{(k+1)})^{-1}H^{(k)T}X, 0)$ ;
  $k = k + 1$
**while** $k \leq K$ ***and*** $(f(H^k, S^{(k)}) - f(H^{(k+1)}, S^{(k+1)}))/f(H^{(k)}, S^{(k)}) > 0.001$;

$$\boxed{\frac{\partial \mathrm{Tr}(AB)}{\partial A} = B^T}$$

$$\boxed{\mathrm{Tr}(AB) = \mathrm{Tr}(BA)}$$

Nous remarquons que, dans la relation de mise à jour, $H^{(k)}$ n'affecte pas $H^{(k+1)}$, ce dernier dépend uniquement de $S^{(k)}$. De même, $S^{(k)}$ n'affecte pas $S^{(k+1)}$ , qui dépend uniquement de $H^{(k)}$.

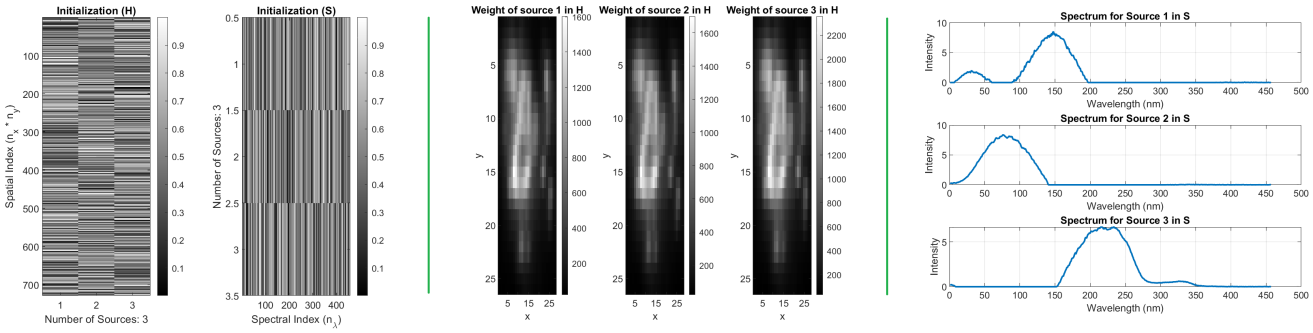



FIGURE 3.2.1 – Initialisation : $H^{(0)}$, $S^0$ | $H^{(K+1)}$ | $S^{(K+1)}$

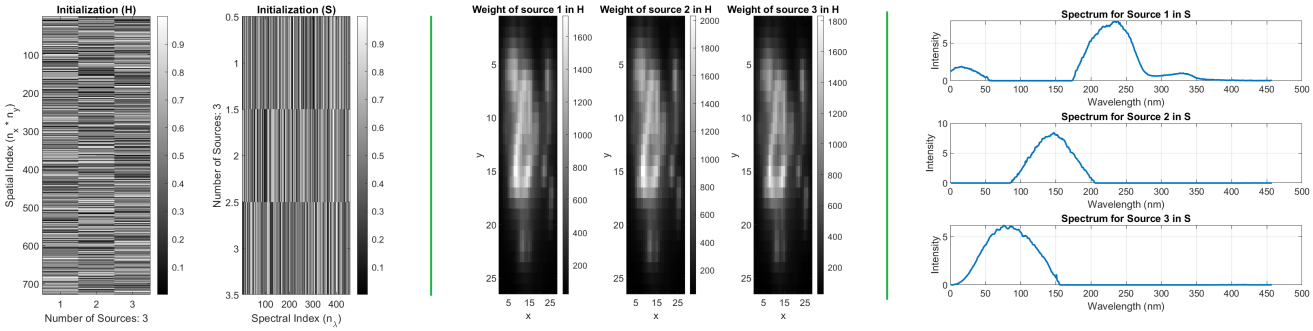



FIGURE 3.2.2 – Initialisation : $H^{(0)}$, $S^0$ | $H^{(K+1)}$ | $S^{(K+1)}$

Cet algorithme a bien fait son travail : on peut voir les composantes à 20 $nm$, 80 $nm$, 150 $nm$, 235 $nm$, 330 $nm$ , mais il ne sait pas à quel source appartient chaque composante. Toutefois, cet algorithme a tout de même réussi à bien séparer ces composantes. De plus, cet algorithme n'est pas influencé par les valeurs initiales $H^{(0)}$ et $A^{(0)}$.

# 4 Conclusion

Pour la méthode 1, le temps de calcul est faible, mais le résultat est influencé par les valeurs initiales $H^{(0)}$ et $A^{(0)}$. Il est adapté au traitement des données de grande dimension.

Pour la méthode 2, l'opération d'inversion est très coûteuse et le temps de calcul est long. Cependant, le résultat est meilleur. Il est adapté au traitement des données de petite dimension.

# Références

[1] C. Soussen. *Traitement d'images hyperspectrales.* Université Paris-Saclay, 2024-2025.

# A Table des Programmes

## Listings

# B Programmes

TP.py

Listing 1 – TP.py

```
% NMF

nb_sources = 3;

pause;

% 6. NMF

disp('Run NMF');

% ....A faire....

method  = 2;

p = nb_sources; %nb_sources = 3;
K = 1000;
H = abs(rand(728, p, K));
S = abs(rand(p, 457, K));

f = @(H, S) norm(X_zoom_preprocess - H * S, 'fro');

for k = 1 : K

        s_previous = S(:,:,k);
        h_previous = H(:,:,k);

        if method  == 1
        % Update H
        h_present = h_previous .* (X_zoom_preprocess * s_previous')./(h_previous * s_previous * s_previous');
        H(:, :, k+1) = h_present;

        % Update S
        s_present = s_previous .* (h_present' * X_zoom_preprocess)./(h_present'* h_present * s_previous);
        S(:, :, k+1) = s_present;
        end

        if method == 2
        % Update H
        h_present = max((X_zoom_preprocess * s_previous')*inv(s_previous * s_previous') , 0);
        H(:, :, k+1) = h_present;

        % Update S
        s_present = max(inv(h_present' * h_present) * (h_present' * X_zoom_preprocess), 0);
        S(:, :, k+1) = s_present;
        end
    if (f(h_previous, s_previous) - f(h_present, s_present)) / f(h_previous, s_previous) < 1e-3
        k_fin = k+1;
        break
    end

end

% 7. affichage des resultats

% ....A completer....

nx = 26; ny = 28;
disp('Displaying results...');

% H(k+1), S(k+1)
```

```
H_final = H(:, :, k_fin);
S_final = S(:, :, k_fin);

% H
figure;
for i = 1:p
    subplot(1, p, i);
    imagesc(reshape(H_final(:, i), nx, ny));
    colormap gray;
    colorbar;
    title(['Weight of source ', num2str(i), ' in H']);
    xlabel('x');
    ylabel('y');
end

% S
figure;
for i = 1:p
    subplot(p, 1, i);
    plot(S_final(i, :), 'LineWidth', 1.5);
    grid on;
    xlabel('Wavelength (nm)');
    ylabel('Intensity');
    title(['Spectrum for Source ', num2str(i), ' in S']);
end

% H S
X_reconstructed = H_final * S_final;
figure;
imagesc(X_reconstructed);
colormap gray;
colorbar;
title('Reconstructed Data (X)');
xlabel('Spectral Index n_{\lambda}');
ylabel('Spatial Index n_x * n_y');

% X zoom preprocess v.s. HS
figure;
subplot(1, 2, 1);
imagesc(X_zoom_preprocess);
colormap gray;
colorbar;
title('Original Data (X)');
xlabel('Spectral Index n_{\lambda}');
ylabel('Spatial Index n_x * n_y');

subplot(1, 2, 2);
imagesc(X_reconstructed);
colormap gray;
colorbar;
title('Reconstructed Data (X)');
xlabel('Spectral Index n_{\lambda}');
ylabel('Spatial Index n_x * n_y');

disp('Results displayed.');

% H(0), S(0)
% H(0)
figure;
subplot(1, 2, 1);
imagesc(H(:,:,1));
colormap(gray);
colorbar;
axis on;
title('Initialization (H)');
xlabel(['Number of Sources: ', num2str(nb_sources)]);
ylabel('Spatial Index (n_x * n_y)');

% S(0)
```

```
130 subplot(1, 2, 2);
131 imagesc(S(:,:,1));
132 colormap(gray);
133 colorbar;
134 axis on;
135 title('Initialization␣(S)');
136 xlabel('Spectral␣Index␣(n_{\lambda})');
137 ylabel(['Number␣of␣Sources:␣', num2str(nb_sources)]);
```