TP APPLI
SAFT : Synthetic Aperture Focusing Technique

Orane VERNET
Diane CHOUNLAMOUNTRY

Février 2024

1 Introduction

Dans ce TP, on s'intéresse à l'imagerie SAFT pour le contrôle par ultrasons. On travaille avec un bloc d'aluminium contenant des petits trous et une sonde multi-élément en contact direct avec la pièce. On considère que tout se passe dans le plan XZ et le jeu de données acquis est un ensemble de signaux temporels (A-scans) qui sont regroupés dans la matrice A (colonne i de A correspond au signal reçu par le capteur u_i). Le but de l'imagerie est de calculer une image discrète O, matrice correspondant aux dimensions spatiales x et z.

Dans le cadre de ce TP, nous allons mettre en oeuvre une méthode d'imagerie temporelle implémentée dans un premier temps en Matlab. Ensuite, on convertira ce code en fonction MEX (exécution depuis Matlab de fonctions en langage C). Finalement, on comparera les temps de calcul entre ces deux mises en oeuvre.

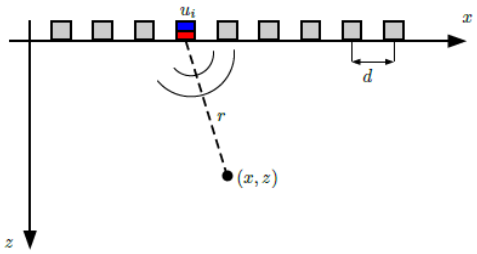

Figure 1: Principe de l'imagerie SAFT

Sommaire

2 Ouverture du fichier

On ouvre le fichier dataSAFT_exp.mat qui contient la matrice A, la fréquence d'échantillonnage temporel F_s, l'espacement $d = 0.5mm$ entre deux capteurs et la vitesse des ondes longitudinales $c = 6300m/s$.
On a $N_t = 1500$ et $N_{el} = 128$ qui correspondent respectivement au nombre de lignes et de colonnes de la matrice A.
Après avoir construit les variables t et u qui correspondent aux points de discrétisation des éléments de A, on peut afficher l'image des données :

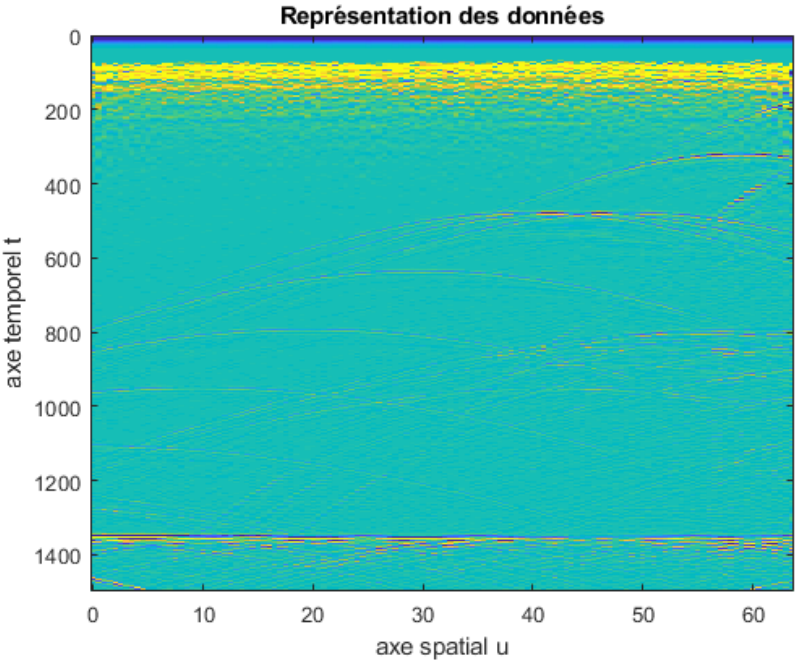

Figure 2: Image initiale des données

Remarque : cette image n'est pas exploitable car il y a du bruit. Les démarquations des différentes zones ne sont pas claires et visuelles.

On affiche un A-scan en fonction du temps, ici on a pris la première colonne de A donc le signal correspond à celui reçu par le premier capteur. On observe que les ordonnées vont de -150 à 150, il faut que N_{bits} soit tel que $N_{bits} = 2^p :> 300$: on en déduit que $N_{bits} = 512 = 2^9$ sont nécessaires pour quantifier les données.

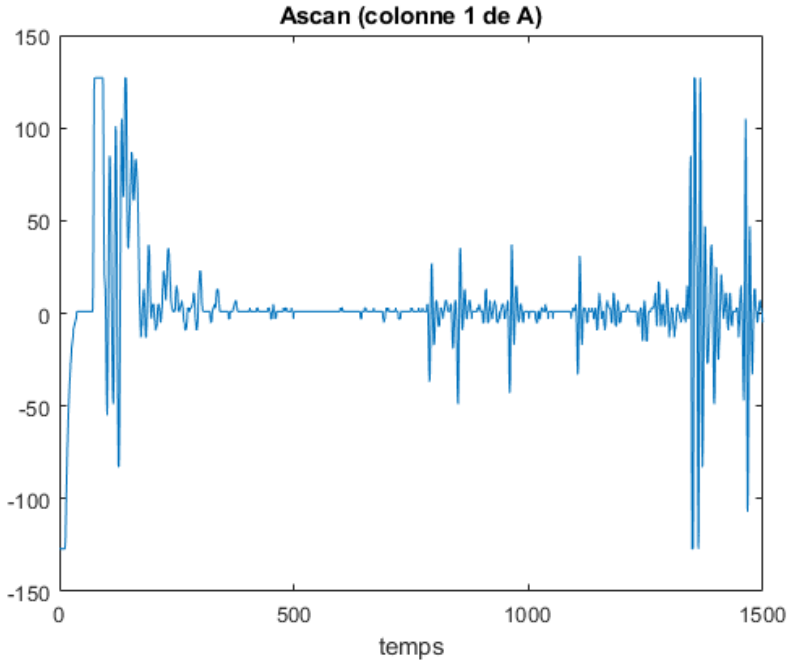

Figure 3: A-scan (1ère colonne de A)

3 Définition de la grille de reconstruction

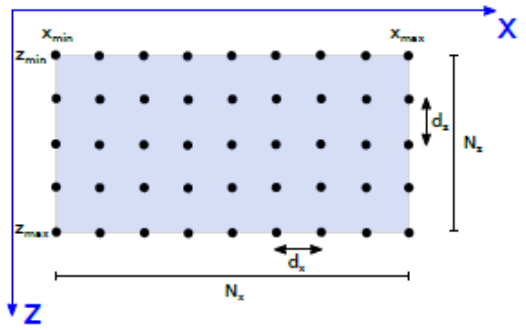

Figure 4: Schéma de la grille de reconstruction dans le plan XZ

Dans cette partie on doit juste implémenter la grille de reconstruction représentée par la figure ci-dessus.
On choisit un nombre de points N_x arbitraire (puissance de deux) puis on définit les extrémités de la sonde : $x_{min} = 0$ et $x_{max} = u(N_{el})$ pour ensuite construire le vecteur de positions $x = linspace(x_{min}, x_{max}, N_x)$. On fait pareil pour l'axe z sachant que $dz = \frac{c}{2F_s}$ et qu'on a $N_z = N_t$.

4 Reconstruction de l'image par une méthode temporelle

Il est temps de reconstruire l'image O en "focalisant directementen chaque point (x,z)" c'est-à-dire sommant sur tous les éléments des échantillons pris au bons moments. Le point (x, z) est reconstruit via :

$$O(x,z) = \sum_{i=1}^{N_{el}} a\Big(\tau(x,z,u_i), i\Big)$$

Sachant qu'on a $r = sqrt(z^2 + (x-u_i)^2$ (obtenu simplement avec Pythagore, cf Figure1) et que le temps de propagation correspondant à l'aller-retour est $\tau(x,z,u_i) = \frac{2r}{c}$. On obtient une image reconstruite en $t_{calcul} = 1,9s$:

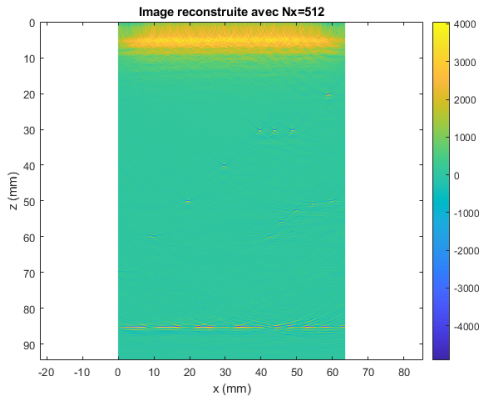

Figure 5: Image reconstruite avec $N_x = 512$

Commentaires : Grâce à cette reconstruction, on arrive maintenant à distinguer l'emplacement des petits trous (ce qui n'était pas du tout le cas avant) mais ils ne sont pas bien mis en avant, notamment avec le jeu de couleur l'image mise sur le rapport ne permet pas encore vraiment d'observer les trous.

On peut faire varier N_x pour observer l'évolution de la qualité de reconstruction et le temps de calcul (plus en détails dans la partie 7).

5 Post-traitement de l'image

Pour que l'image soit encore plus pratique à analyser, on effectue un post-traitement. On calcule d'abord l'enveloppe de l'image avec la commande *abs(hilbert(O))* puis on normalise par rapport au maximum de l'image. On obtient la nouvelle image suivante :

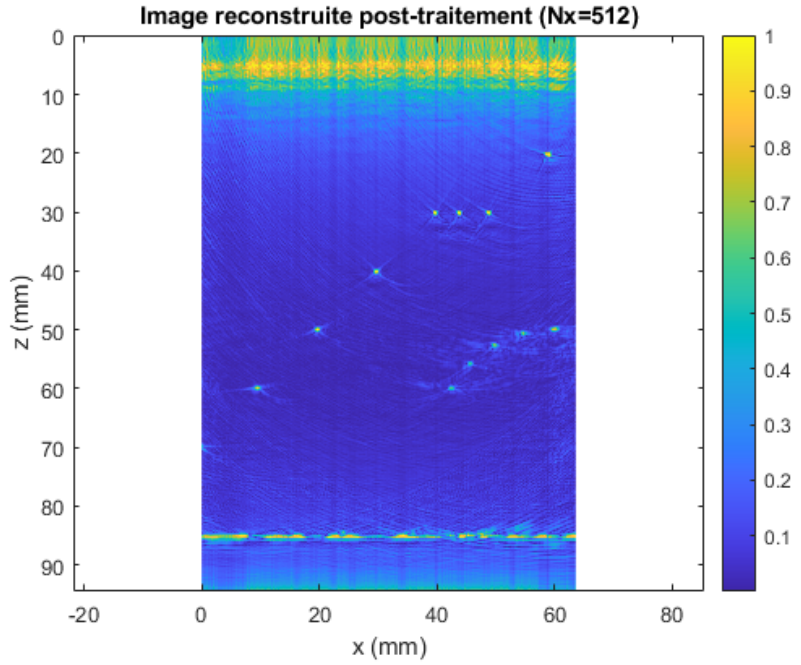

Figure 6: Image reconstruite post-traitement (avec $N_x = 512$)

Commentaires : Désormais, les trous sont vraiment visibles et leurs emplacements sont très précis. La reconstruction est très satisfaisante.

6 Avec une fonction MEX

Les fonctions MEX sont des fonctions en langage C qui sont compilées pour pouvoir être utilisées avec Matlab. On dispose dans ce TP du fichier *MEX_SAFT.c* qui contient la fonction d'appel *mexFunction* à ne pas modifier et la fonction *MEX_SAFT* que nous devons compléter. Pour se faire, il a fallu traduire le code Matlab en langage C, en faisant attention aux petites différences de syntaxe et en prenant en compte que les matrices A et O sont "vectorisées" (colonnes mises bout à bout pour former un vecteur d'une dimension et non plus une matrice).

Une fois une compilation sans erreur, on peut appeler la fonction MEX dans Matlab afin de reconstruire l'image O. Après redimensionnement et post-traitement, on obtient l'image suivante :

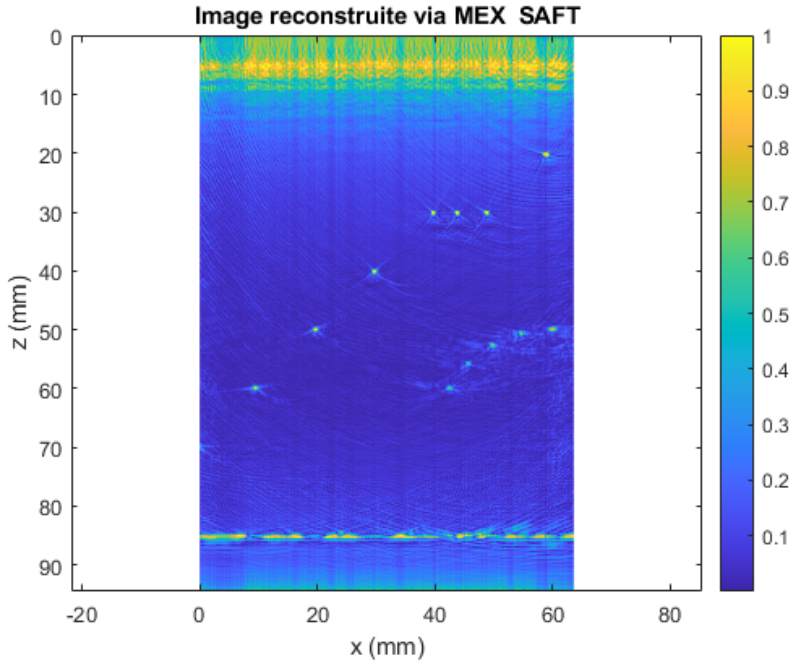

Figure 7: Image reconstruite avec $N_x = 512$ via la fonction MEX

Commentaire : Les trous sont bien visibles. Le temps de calcul est de $t_{calcul} = 0,9s$.

7 Comparaison des deux mises en oeuvre

Voici les résultats obtenus avec la fonction matlab et la fonction mex respectivement :

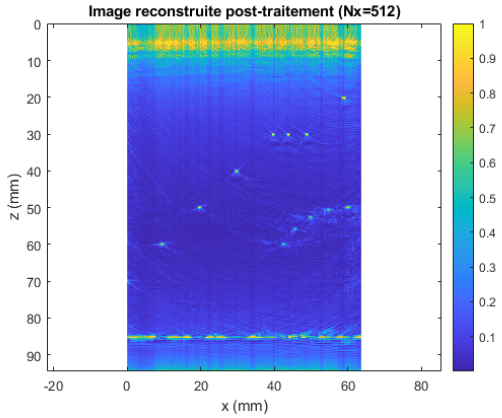

Figure 8: Image reconstruite via Matlab

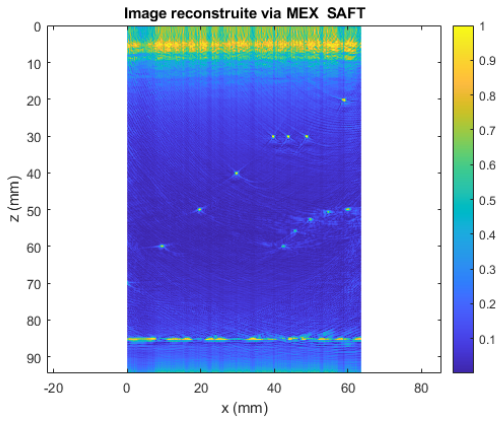

Figure 9: Image reconstruite via la fonction MEX

A l'oeil nu on ne remarque pas de différence entre les images. En revanche, lorqu'on lance les algorithmes (on a pris ici $N_x = 512$), on constate uen différence notoire dans les temps de calcul : 1.9s pour la fonction matlab contre seulement 0.9s pour la fonction mex.
Cela se remarque lorsqu'on trace le temps de calcul en fonction de N_x :

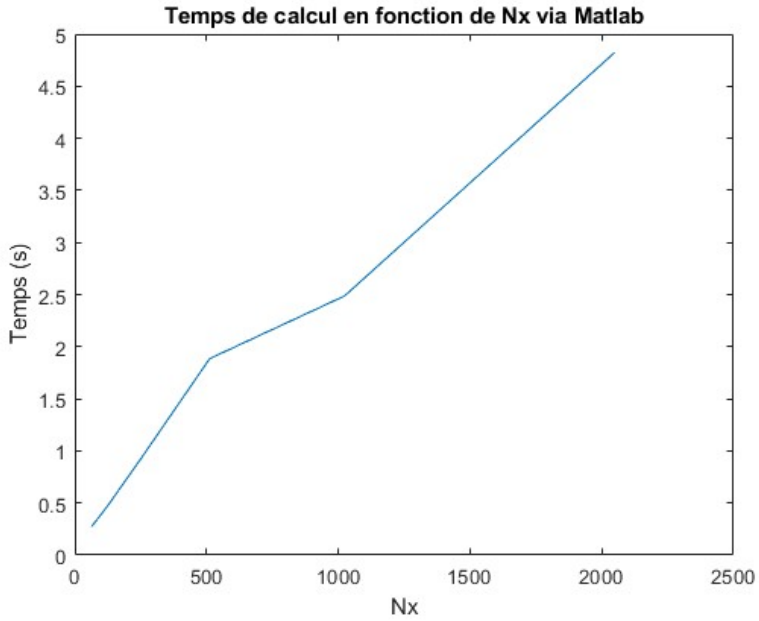

Figure 10: Temps de calcul en fonction de N_x pour Matlab

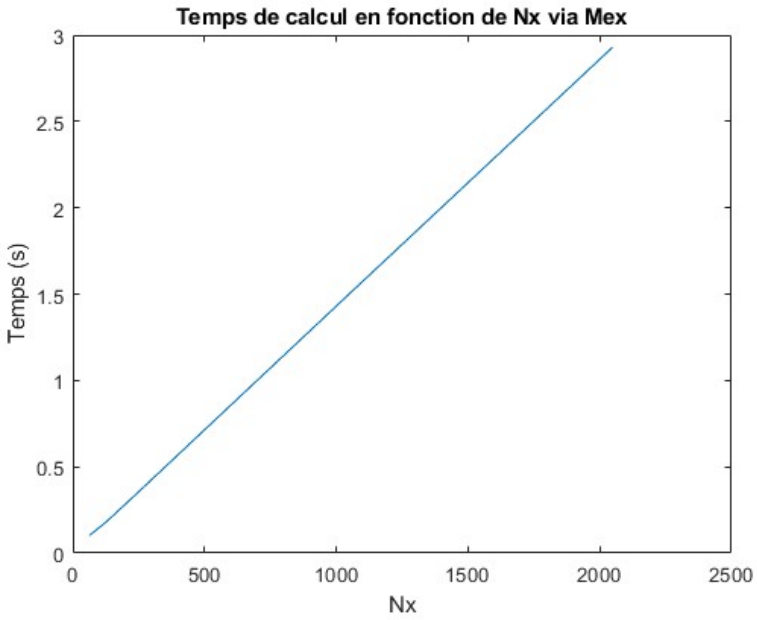

Figure 11: Temps de calcul en fonction de N_x pour MEX

L'accélération entre MAT et MEX (rapport MAT/MEX) est d'environ 2.5 pour $N_x = 64, 128, 256, 512$ et d'environ 1.5 pour $N_x = 1024, 2048$ (MEX plus rapide).

On trace ensuite la ligne z=30mm qui correspond aux trois trous alignés pour les deux images reconstruites (Matlab et MEX). Voici ce qu'on obtient :

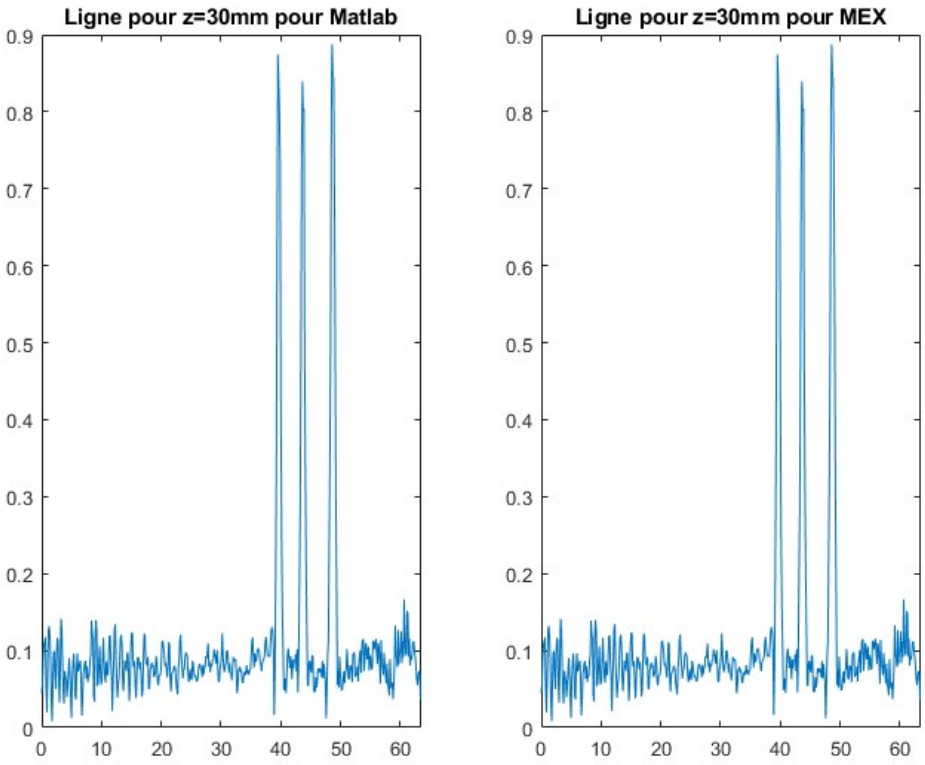

Figure 12: Signaux correspond à z=30mm en fonction de x

On visualise bien les trois trous qui correspondent aux trois pics sur le signal. Les deux premiers trous sont espacés d'une distance d'environ 4.1mm selon x et le trou 2 et 3 sont espacés d'environ 5mm selon x (même distances pour les deux images).

On trace ensuite l'erreur qui correspond à la différence entre le signal de la ligne z=30mm de l'image reconstruite via Matlab et le signal de la ligne z=30mm de l'image reconstruite via MEX :

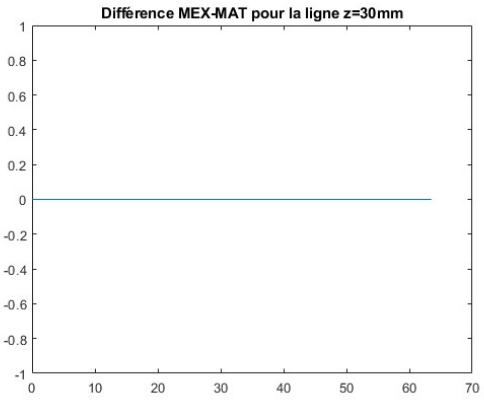

Figure 13: Erreur MEX-Mat pour z=30mm

On constate que celle-ci est nulle. Les performances en terme de précision de ces deux algorithmes sont donc similaires.

8 Conclusion

Dans ce TP nous avons pu reconstruire des images depuis des signaux temporels (A-scan) grâce à des fonctions implémentées en Matlab et en MEX qui englobe la définition d'une grille de reconstruction, l'implémentation d'une méthode temporelle et un post-traitement. Pour un nombre de points N_x donné, on constate que la précison en terme de reconstruction est plutôt similaire pour les deux algorithmes. En revanche le temps de calcul varie grandement entre les deux méthodes puisqu'on observe un temps de calcul 2.5 fois moins important via MEX pour les plus petites valeurs et 1.5 fois moins important lorsque la valeur de N_x augmente.