

Imagerie ultrasonore par ouverture synthétique

Ewen Carcreff

Compte rendu à envoyer pour le 09/02/2024 à ewen.carcreff@tpac-ndt.com

1 Présentation du TP

Ce TP concerne l'imagerie *Synthetic Aperture Focusing Technique* (SAFT) pour le contrôle par ultrasons. Nous nous intéressons au contrôle d'un bloc d'aluminium contenant des petits trous à l'aide d'une sonde multi-élément en contact direct avec la pièce, comme présenté sur la figure 1. Nous considérons la vitesse des ondes longitudinales constante $c = 6300 \text{ m/s}$ ¹.

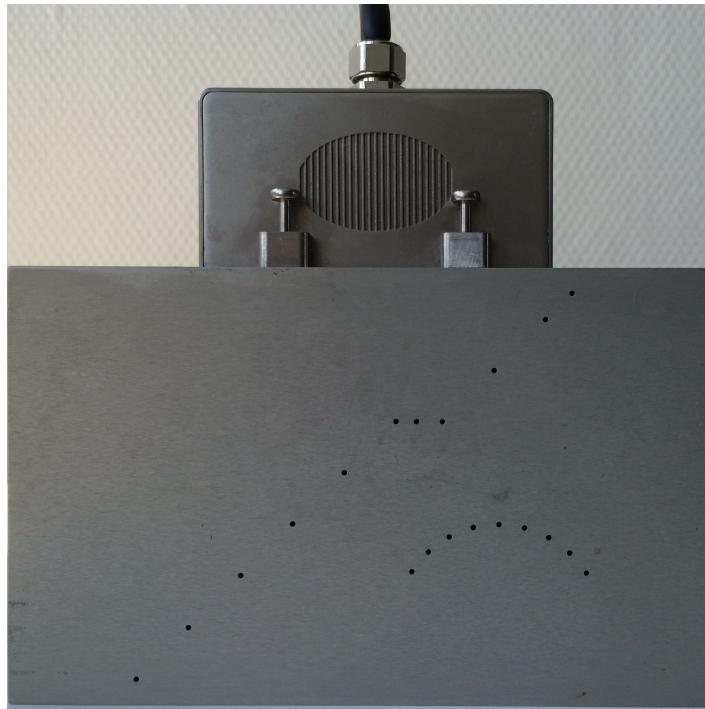


FIGURE 1 – Photo de l'inspection

Le principe est dans un premier temps d'acquérir les données en utilisant chaque élément d'un réseau de capteurs indépendamment. Sur la figure 2, on peut voir le principe de l'inspection. Nous considérons ici une sonde de fréquence centrale 5 MHz ayant $N_{el} = 128$ éléments et une distance inter-élément $d = 0.5 \text{ mm}$. La profondeur est notée z et la largeur x . Nous considérons que tout se passe dans le plan XZ et ne prenons pas en compte la dimension Y. Chaque élément i , pour $i = 1 \dots N$, est placé en $(u_i, 0)$ où $u_i = (i - 1)d$. Le jeu de données acquis est donc un ensemble de signaux temporels (dits A-scans) $a(t, i)$, correspondant aux signaux reçus pour chaque position u_i du capteur actif. Ces signaux sont regroupés dans la matrice $\mathbf{A} \in \mathbb{R}^{N_t \times N_{el}}$. N_t est le nombre d'échantillons temporels des données.

Le but de l'imagerie est de calculer une image discrète $\mathbf{O} \in \mathbb{R}^{N_x \times N_z}$, qui est une matrice correspondant aux dimensions spatiales x et z . Ce calcul se fait donc pour une grille de calcul (x, z) préalablement définie.

Dans ce TP, nous mettrons en œuvre une méthode d'imagerie dite temporelle car elle utilise directement les données en fonction du temps. Celle-ci sera implémentée dans Matlab à l'aide de boucles *for*. Ensuite, nous convertirons ce code en fonction MEX, qui permet d'exécuter des fonctions en langage C depuis Matlab. On comparera les temps de calcul entre ces deux mises en œuvre.

1. **N.B.** Pour des aspects pratiques, il est préférable d'exprimer les distances en mm, le temps en μs et la vitesse en $\text{mm}/\mu\text{s}$.

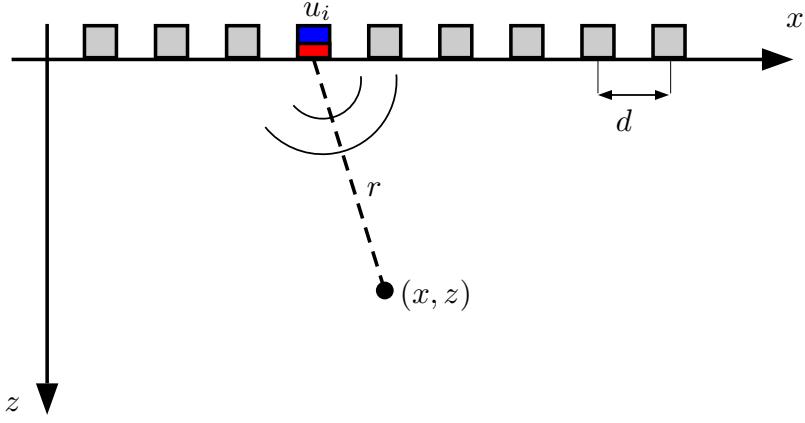


FIGURE 2 – Principe de l'imagerie SAFT. L'émission-réception est réalisée successivement pour chaque élément i .

2 Ouverture du fichier

- Créer un fichier `MAIN.m` qui contiendra votre programme principal.
- Ouvrir le fichier `dataSAFT_exp.mat` à l'aide de la fonction `load` (ce fichier contient les données \mathbf{A} , la fréquence d'échantillonnage temporel F_s , l'espacement d entre deux capteurs et la vitesse des ondes c).
- Observer la taille de la matrice et en déduire N_t et N_{el} .
- Construire les variables correspondant à l'axe temporel t et à l'axe spatial \mathbf{u} , correspondant aux points de discréttisation des éléments de la matrice \mathbf{A} .
- Afficher l'image des données en fonction de t et \mathbf{u} avec `imagesc`.
- Est-ce que cette image est exploitable ?
- Afficher un Ascan (*i.e.*, un signal temporel, correspondant à une colonne de \mathbf{A}) en fonction du temps.
- Déduire le nombre de bits utilisé pour la quantification.

3 Définition de la grille de reconstruction

Le but est ici de définir la grille de reconstruction (x, z) telle qu'elle est représentée sur la figure 3. Il faut donc construire les vecteurs \mathbf{x} et \mathbf{z} .

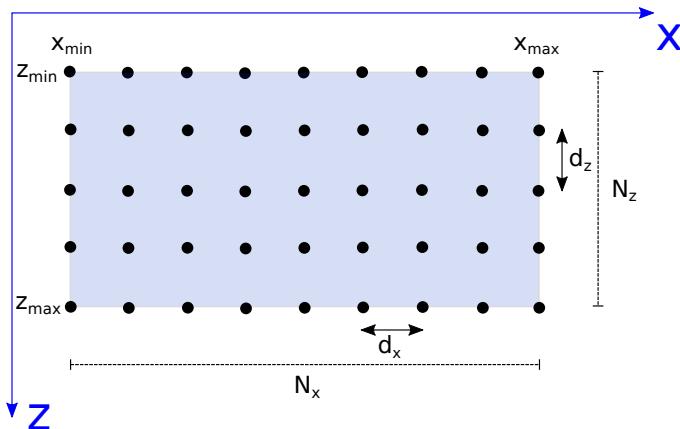


FIGURE 3 – Schéma de la grille de reconstruction dans le plan XZ. Chaque point noir est un point de calcul (x, z) .

- Définir un nombre de points N_x arbitraire (64, 128, 256, 512, 1024, 2048).
- Définir x_{min} et x_{max} comme étant les extrémités de la sonde.
- Construire le vecteur \mathbf{x} de positions selon l'axe horizontal.
- Définir le pas dz du vecteur z tel que $dz = c/(2F_s)$ (2 correspond à l'aller-retour).
- Définir z_{min} et z_{max} correspondant au extrémités du vecteur t .
- Construire le vecteur \mathbf{z} de positions selon l'axe vertical.

4 Reconstruction de l'image par une méthode temporelle

La reconstruction de l'image **O** peut s'effectuer en "focalisant directement en chaque point (x, z) ". Le principe est de faire la somme sur tous les éléments des échantillons pris aux bons instants temporels. La reconstruction pour le point (x, z) sera de la forme :

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

- D'après la figure 2, exprimer la distance aller r en fonction de x, z et u_i .
- Calculer le temps de propagation $\tau(x, z, u_i)$ correspondant à l'aller-retour ($2r$).
- Compléter l'équation de $O(x, z)$.
- Initialiser l'image **O** par : $O = \text{zeros}(Nz, Nx)$.
- Proposer une procédure qui reconstruit l'image **O**. La fonction `round` peut être nécessaire.
- Mesurer le temps de calcul de l'image reconstruite à l'aide de `tic ... toc`.
- Afficher l'image en fonction de x et z .
- Utiliser la commande `axis equal` pour mettre à l'échelle.
- Faire varier N_x , commenter la qualité de reconstruction et le temps de calcul.

5 Post-traitement de l'image

Dans cette partie, nous affinons l'image obtenu pour qu'elle soit plus pratique à analyser. Il faut :

- Calculer l'enveloppe de l'image avec $O = \text{abs}(\text{hilbert}(O))$.
- Normaliser par rapport au maximum de l'image.
- Afficher la nouvelle image en fonction de x et z , et commenter.

6 Accélération avec une fonction MEX

Les fonctions MEX sont des fonctions en langage C ou C++ compilées pour être utilisables dans l'environnement Matlab.

6.1 Exemple de fonction MEX

Toutes les fonctions MEX ont la même fonction d'appel `mexFunction`. Voici un exemple de la fameuse fonction "Hello World" dont le code est contenu dans un fichier C (nomDuFichier.c).

```
# include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    mexPrintf("Hello World !");
}
```

La fonction `mexFunction` est le point d'entrée du fichier MEX (à la manière du main en langage C). Elle prend toujours quatre arguments :

- `nlhs` : entier indiquant le nombre d'arguments de sortie.
- `plhs` : tableau de pointeurs contenant les **sorties**.
- `nrhs` : entier indiquant le nombre d'arguments d'entrée.
- `prhs` : tableau de pointeurs contenant les **entrées**. La déclaration `const` signifie que ces variables ne sont pas modifiables par le programme.

6.2 Utilisation des fonctions MEX

La compilation de la fonction se fait à la ligne de commande dans Matlab (`mex nomDuFichier.c`) et crée un fichier `nomDuFichier_mexw64`. Un fichier MEX s'utilise comme une fonction Matlab ordinaire. On appelle ainsi simplement ce fichier en lui passant des arguments d'entrée et en récupérant des arguments de sortie :

```
[sortie1, sortie2, ...] = nomDuFichier(entre1, entre2, ...);
```

Dans la fonction `mexFunction`, il faut tout d'abord récupérer les variables d'**entrée** :

- Pour un scalaire :

```
double c;
c = mxGetScalar(prhs[0]);
```
- Pour un vecteur :

```
double* pA;
pA = mxGetPr(prhs[1]);
```

Une variable de **sortie** de type vecteur ou matrice (de taille préalablement définie `NoutSize`) peut être affectée de la façon suivante :

```
double* pO;
plhs[0] = mxCreateNumericMatrix((mwSize)NoutSize, 1, mxDOUBLE_CLASS, 0);
pO = mxGetData(plhs[0]);
```

6.3 Travail à effectuer

- Ouvrir le fichier `MEX_SAFT.c`.
- Définir dans Matlab les entrées et sorties de la fonction MEX .
- Compléter la fonction `MEX_SAFT` qui effectue la reconstruction SAFT^{2 3 4}
- **Dans la fonction MEX, `p_A` et `p_O` doivent être des vecteurs. Pour ce faire, utiliser une indentation colonne par colonnes.**
- Compiler dans Matlab la fonction par la commande `mex MEX_SAFT.c`.
- Observer la bonne compilation (pas d'erreur).
- Appeler la fonction MEX dans Matlab pour reconstruire l'image **O**.
- Redimensionner l'image à l'aide de `reshape`.
- Afficher l'image obtenue en intégrant les post-traitements .
- Mesurer le temps de calcul à l'aide de `tic ... toc`.

7 Comparaison des deux mises en œuvre

- Pour la version Matlab et la version MEX, afficher les images reconstruites. Sont-elles identiques ?
- Faire varier N_x et tracer les temps de calcul dans une figure.
- Donner une estimation de l'accélération entre les codes MEX et Matlab.
- Tracer la ligne pour $z = 30$ mm qui correspond aux trois trous proches des images reconstruites (Matlab et MEX). De combien sont espacés ces trous dans la direction x ?
- Faire la différence entre les deux signaux et tracer l'erreur. Est-elle nulle ?

2. **N.B.** Ne pas modifier la fonction `mexFunction`.

3. **N.B.** Pour faire x^2 , ne pas utiliser la fonction `pow`. Utiliser plutôt `x*x`.

4. **N.B.** La fonction `round` n'existe pas en langage C. Utiliser `floor(x+0.5)`.